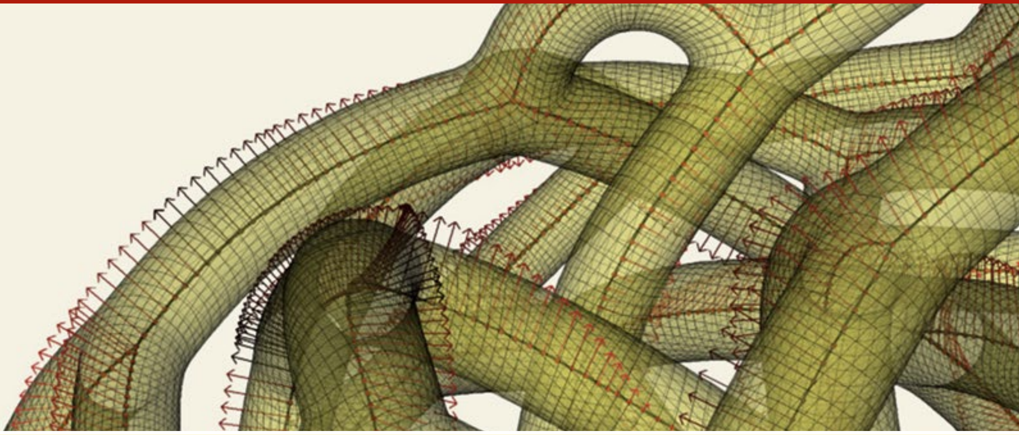
Mathematics and Visualization



Janine Bennett · Fabien Vivodtzev
Valerio Pascucci *Editors*

Topological and Statistical Methods for Complex Data

Tackling Large-Scale, High-Dimensional, and Multivariate Data Spaces

Springer

Mathematics and Visualization

Janine Bennett • Fabien Vivodtzev •
Valerio Pascucci
Editors

Topological and Statistical Methods for Complex Data

Tackling Large-Scale, High-Dimensional, and Multivariate Data Spaces

With 120 Figures, 101 in color

Springer

Editors

Janine Bennett
Sandia National Laboratories
Livermore, CA
USA

Fabien Vivodtzev
CEA CESTA, CS 60001
Le Barp CEDEX
France

Valerio Pascucci
School of Computing and SCI Institute
University of Utah
Salt Lake City, UT
USA

ISSN 1612-3786 ISSN 2197-666X (electronic)
ISBN 978-3-662-44899-1 ISBN 978-3-662-44900-4 (eBook)
DOI 10.1007/978-3-662-44900-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014955717

Mathematics Subject Classification (2010): 54-02, 62-02, 62-07

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The continued improvements in high performance computing and high resolution sensing capabilities are resulting in data of unprecedented size and complexity. Historically topological and statistical techniques have been deployed as independent alternatives in the analysis of a variety of data types. However, the continued increases in size, dimensionality, and number of variables create new challenges that traditional approaches cannot address. New methods that leverage the mutual strengths of both topological and statistical techniques are needed to support the management, analysis and visualization of such complex data.

In an effort to characterize the current challenges and research trends, and to foster collaborations, we organized the Workshop on the Analysis of Large-scale, High-dimensional, and Multivariate Data using Topology and Statistics, held June 12–14 in Le Barp, France. Around 30 researchers from 20 European and American universities, companies, and national research laboratories were in attendance. The program comprised 18 presentations, including a keynote talk by Herbert Edelsbrunner from the Institute of Science and Technology Austria, titled "Persistent Homology: Theory and Practice." A number of interesting challenges were addressed during the workshop, with presentations covering a wide range of topics, including topological techniques for large data, high-dimensional data analysis, computational challenges, multivariate visualization and analysis techniques.

In this book, we present 16 peer-reviewed chapters, divided into 6 parts as the outcome of this workshop. Parts I and II focus on large-scale data, Parts III and IV focus on multivariate data, and Parts V and VI focus on high-dimensional data. The chapters in Part I include recent results in the area of in-situ and distributed analysis. We start with a distributed-memory algorithm for labeling connected components in simulation data (Harrison et al.), followed by a discussion of in-situ visualization in fluid mechanics (Ribes et al.). Part I concludes with a survey of recent discoveries in sublinear algorithms for extreme-scale data analysis (Seshadhri et al.). Part II focuses on the efficient representation of large functions and includes a report on optimal general simplification of scalar fields on surfaces (Tierny et al.), an algorithm for piecewise polynomial monotonic interpolation of 2D gridded data (Allemand-Giorgis et al.), and a technique for shape analysis using real functions (Biasotti et al.). The chapters in Part III focus on structural techniques for multivariate data. This part includes a survey on 3D symmetric tensor fields that highlights what we know and where to go next (Zhang and Zhang), followed by a comparison of Pareto Sets and Jacobi Sets (Huttenberger and Garth), and a report on deformations preserving total curvature (Berres et al.). Part IV focuses on classification and visualization of vector fields, and includes a presentation of Lyapunov time for 2D Lagrangian visualization (Sadlo), followed by a survey of geometric algebra for vector field analysis and visualization (Ausoni and Frey). This part concludes with a report on a technique for computing accurate Morse-Smale complexes from gradient vector fields (Gyulassy et al.). Part V includes chapters focused on the exploration of high-dimensional models, including a presentation of high-dimensional sampling techniques (Ebeida et al.), and a report on the realization of regular maps of large genus (Razafindrazaka and Polthier). Lastly, Part VI presents recent results in the analysis of large, high-dimensional systems, and includes a technique for faster localized solving of systems of equations (Anthony et al.), followed by a system for ensemble analysis of electrical circuit simulations (Crossno et al.).

Fig. 1 Workshop participants, June 2013

In summary, this book brings together recent results from some of the most prominent and recognized leaders in the fields of statistics, topology, and computer science. The book's contents cover both theory and application, providing an overview of important key concepts and the latest research trends. The algorithms detailed in this book are broadly applicable and can be used by application scientists to glean insight from complex data sets.

Acknowledgements

The Workshop on the Analysis of Large-Scale, High-Dimensional, and Multivariate Data Using Topology and Statistics was co-organized by the French Atomic Energy Commission and Alternative Energies (CEA), Sandia National Laboratories, and Kitware, SAS. The organizing committee greatly acknowledges Julien Jomier and Charles Marion for their administrative support in preparation for the workshop, Sophie Vivodtzev for her logistical help, and thanks all of the workshop participants for making it such an enjoyable and informative event. Lastly, the editors would like to thank all of the contributors of this volume for their insightful contributions.

Livermore, CA, USA — Janine Bennett
Le Barp, France — Fabien Vivodtzev
Salt Lake City, UT, USA — Valerio Pascucci
July 2014

Contents

List of Contributors

Léo Allemand-Giorgis Inria - Laboratoire Jean Kuntzmann, University of Grenoble, Grenoble, France

Eleanor Anthony Mathematics Department, University of Mississippi, MS, USA

Chantal Oberson Ausoni Sorbonne Universités, UPMC Univ Paris 06, Institut du Calcul et de la Simulation, Paris, France

Muhammad A. Awad Department of Naval Architecture and Marine Engineering, Alexandria University, Alexandria, Egypt

Janine C. Bennett Sandia National Laboratories, Livermore, CA, USA

Anne Berres University of Kaiserslautern, Kaiserslautern, Germany

Harsh Bhatia SCI Institute, University of Utah, Salt Lake City, UT, USA

Silvia Biasotti CNR-IMATI, Genova, Italy

Ryan Bleile The University of Oregon, Eugene, OR, USA

Georges-Pierre Bonneau Inria - Laboratoire Jean Kuntzmann, University of Grenoble, Grenoble, France

Peer-Timo Bremer Lawrence Livermore National Laboratory, Livermore, CA, USA

Andrea Cerri CNR-IMATI, Genova, Italy

Hank Childs The University of Oregon and Lawrence Berkeley National Laboratory, Eugene, OR, USA

Patricia J. Crossno Sandia National Laboratories, Albuquerque, NM, USA

Andrew A. Davidson Department of Electrical and Computer Engineering, University of California, Davis, CA, USA

Mohamed S. Ebeida Computing Research, Sandia National Laboratories, Albuquerque, NM, USA

Bianca Falcidieno CNR-IMATI, Genova, Italy

Pascal Frey Sorbonne Universités, UPMC Univ Paris 06, Institut du Calcul et de la Simulation, Paris, France

Sorbonne Universités, UPMC Univ Paris 06, Laboratoire Jacques-Louis Lions, Paris, France

Yvan Fournier EDF R&D, Chatou, France

Kelly Gaither The University of Texas at Austin (Texas Advanced Computing Center), Austin, TX, USA

Christoph Garth University of Kaiserslautern, Kaiserslautern, Germany

Sheridan Grant Mathematics Department, Pomona College, Claremont, CA, USA

Peter Gritzmann Fakultät für Mathematik, Technische Universität München, München, Germany

David Günther Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI, Paris, France

Attila Gyulassy SCI Institute, University of Utah, Salt Lake City, UT, USA

Hans Hagen University of Kaiserslautern, Kaiserslautern, Germany

Stefanie Hahmann Inria - Laboratoire Jean Kuntzmann, University of Grenoble, Grenoble, France

Cyrus Harrison Lawrence Livermore National Laboratory, Livermore, CA, USA

Ming-Yu Hsieh Sandia National Laboratories, Albuquerque, NM, USA

Lars Huettenberger University of Kaiserslautern, Kaiserslautern, Germany

Warren L. Hunt Sandia National Laboratories, Albuquerque, NM, USA

Julien Jomier Kitware, Villeurbanne, France

Benjamin Lorendeau EDF R&D, Clamart, France

Ahmed H. Mahmoud Department of Naval Architecture and Marine Engineering, Alexandria University, Alexandria, Egypt

Shawn Martin Sandia National Laboratories, Albuquerque, NM, USA

Scott A. Mitchell Computing Research, Sandia National Laboratories, Albuquerque, NM, USA

John D. Owens Department of Electrical and Computer Engineering, University of California, Davis, CA, USA

Valerio Pascucci School of Computing and SCI Institute, University of Utah, Salt Lake City, UT, USA

Anjul Patney Department of Electrical and Computer Engineering, University of California, Davis, CA, USA

Ali Pinar Sandia National Laboratories, Livermore, CA, USA

Konrad Polthier Freie Universität Berlin, Berlin, Germany

Faniry Razafindrazaka Freie Universität Berlin, Berlin, Germany

Alejandro Ribés EDF R&D, Clamart, France

J. Maurice Rojas Mathematics Department, Texas A&M University, College Station, TX, USA

Filip Sadlo University of Stuttgart, Stuttgart, Germany

C. Seshadhri Sandia National Laboratories, Livermore, CA, USA

Timothy M. Shead Sandia National Laboratories, Albuquerque, NM, USA

Milosz A. Sielicki Sandia National Laboratories, Albuquerque, NM, USA

Michela Spagnuolo CNR-IMATI, Genova, Italy

David Thompson Kitware, Inc., Carrboro, NC, USA

Julien Tierny CNRS LIP6, UMPC, Sorbonne Universites, Paris, France

CNRS LTCI, Telecom ParisTech, 46 Rue Barrault, 75013 Paris, France

Stanley Tzeng Department of Computer Science, University of California, Davis, CA, USA

Fabien Vivodtzev CEA: French Atomic Energy Commission and Alternative Energies, Le Barp, France

Jordan Weiler The University of Oregon, Eugene, OR, USA

Eugene Zhang School of Electrical Engineering and Computer Science, 2111 Kelley Engineering Center, Oregon State University, Corvallis, OR, USA

Yue Zhang School of Electrical Engineering and Computer Science, 3117 Kelley Engineering Center, Oregon State University, Corvallis, OR, USA

Part I
Large-Scale Data Analysis: In-Situ and Distributed Analysis

A Distributed-Memory Algorithm for Connected Components Labeling of Simulation Data

Cyrus Harrison, Jordan Weiler, Ryan Bleile, Kelly Gaither, and Hank Childs

1 Introduction

Parallel scientific simulations running on today's state of the art petascale computing platforms generate massive quantities of high resolution mesh-based data. Scientists often analyze this data by eliminating portions and visualizing what remains, through operations such as isosurfacing, selecting certain materials and discarding the others, isolating hot spots, etc. These approaches can generate complex derived geometry with intricate structures that require further techniques for effective analysis, especially in the context of massive data.

In these instances, representations of the topological structure of a mesh is often helpful (Fig. 1). Specifically, a labeling of the connected components in a mesh provides a simple and intuitive topological characterization of which parts of the

C. Harrison
Lawrence Livermore National Laboratory, Livermore, CA, USA
e-mail: cyrush@llnl.gov

J. Weiler • R. Bleile
The University of Oregon, Eugene, OR, USA
e-mail: jweiler@cs.uoregon.edu; rbleile@cs.uoregon.edu

K. Gaither
The University of Texas at Austin (Texas Advanced Computing Center), Austin, TX, USA
e-mail: kelly@tacc.utexas.edu

H. Childs (✉)
The University of Oregon, Eugene, OR, USA

Lawrence Berkeley National Laboratory, Berkeley, CA, USA
e-mail: hchilds@lbl.gov; hank@cs.uoregon.edu

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_1

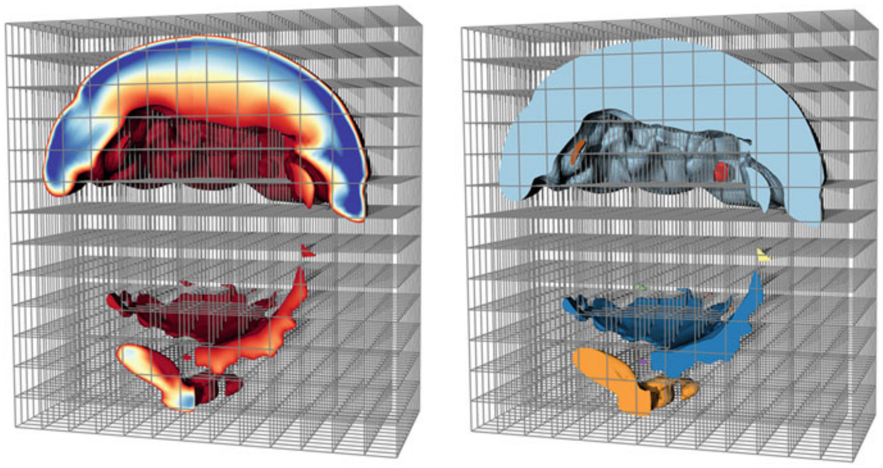


Fig. 1 (*Left*) Sub-volume mesh extracted from a 21 billion cell structured grid decomposed across 2,197 processors. (*Right*) Sub-volume mesh colored by the results from the connected components labeling algorithm described in this chapter

mesh are connected to each other. These unique sub-meshes contain a subset of cells that are directly or indirectly connected via series of cell abutments.

The global nature of connectivity poses a challenge in distributed-memory parallel environments, which are the most common setting for analyzing massive data. This is because massive data sets are typically too large to fit into the memory of a single processor, so pieces of the mesh are distributed across processors. Cells comprising connected sub-meshes may span any of the processors, but the relationships of how cells abut across processors frequently has to be derived. To deal with this problem, sophisticated techniques to resolve connectivity are necessary.

This chapter explores an algorithm that operates on both structured and unstructured meshes and scales well even with very large data, as well as its underlying performance characteristics. The algorithm executes in multiple stages, ultimately constructing a unique label for each connected component and marking each vertex with its corresponding connected component label. This final labeling enables analyses such as: calculation of aggregate quantities for each connected component, feature based filtering of connected components, and calculation of statistics on connected components.

In short, the algorithm provides a useful tool for domain scientists with applications where physical structures, such as individual fragments of a specific material, correspond to the connected components contained in a simulation data set. This chapter presents the algorithm (Sect. 4), results from a weak scaling performance study (Sect. 5), and further analysis of the slowest phase of the algorithm (Sect. 6).

2 Related Work

The majority of research to date in connected components algorithms has been focused on computer vision and graph theory applications. This previous research is useful for contributing high-level ideas, but ultimately the algorithms themselves are not directly applicable to the problem considered here. Computer vision algorithms typically depend on the structured nature of image data, and so cannot be easily applied to unstructured scientific data. Graph theory algorithms are more appropriate, since the cell abutment relationships in an unstructured mesh can be encoded as an undirected graph representation. But this encoding results in a very sparse graph, with the edges having special properties—neighboring cells typically reside on the same processing elements, although not always—that graph theory algorithms are not optimized for. For more discussion of these algorithms, we refer the reader to [11]. That said, previous graph theory research on connected components has used the Union-find algorithm [8], which is also used for the algorithm described in this chapter. Further, the Union-find algorithm and data structures have been used in topology before, for the efficient construction of Contour Trees [5], Reeb Graphs [17], and segmentations [2, 3]. Union-find is discussed further in Sect. 3.1.

The algorithm described in this chapter is intended for distributed-memory parallelism. With this technique, Processing Elements (PEs)—instances of a program—are run on each node, or on each core of a node. By using multiple nodes, the memory available to the program is larger, allowing for processing of data sets so large that they cannot fit into the memory of a single node. Popular end user visualization tools for large data, such as ParaView [1] and VisIt [7], follow this distributed-memory parallelization strategy. Both of these tools instantiate identical visualization modules on each PE, and the PEs are only differentiated by the sub-portion of the larger data set they operate on. The tools rely on the data set being decomposed into pieces (often referred to as domains), and they partition these pieces over their PEs. This approach has been shown to be effective; VisIt performed well on meshes with trillions of cells using tens of thousands of PEs [6]. The algorithm described in this chapter follows the strategy of partitioning data over the PEs and has been implemented as a module inside VisIt. It uses the Visualization ToolKit (VTK) library [15] to represent mesh-based data, as well as its routines for identifying cell abutment within a piece. Finally, we note that other topology algorithms have also been ported to a distributed-memory parallel environment, specifically segmentations [18] and merge trees [13].

3 Algorithm Building Blocks

This section describes three fundamental building blocks used by the algorithm. The first is the serial Union-find algorithm, which efficiently identifies and merges connected components. The second is binary space partitioning trees, which enable efficient computation of mesh intersections across PEs. The third is the concepts of exteriors cells and ghost data, which significantly accelerate the algorithm.

3.1 Union-Find

The Union-find algorithm enables efficient management of partitions. It provides two basic operations: UNION and FIND. The UNION operation creates a new partition by merging two subsets from the current partition. The FIND operation determines which subset of a partition contains a given element.

To efficiently implement these operations, relationships between sets are tracked using a disjoint-set forest data structure. In this representation, each set in a partition points to a root node containing a single representative set used to identify the partition. The UNION operation uses a union-by-rank heuristic to update the root node of both partitions to the representative set from the larger of the two partitions. The FIND operation uses a path-compression heuristic which updates the root node of any traversed set to point to the current partition root. With these optimizations each UNION or FIND operation has an amortized run-time of $O(\alpha(N))$ where N is the number of sets and $\alpha(N)$ is the inverse Ackermann function [16]. $\alpha(N)$ grows so slowly that it is effectively less than four for all practical input sizes. The disjoint-set forest data structure requires $O(N)$ space to hold partition information and the values used to implement the heuristics. The heuristics used to gain efficiency rely heavily on indirect memory addressing and do not lend themselves to a direct distributed-memory parallel implementation.

3.2 Binary Space Partitioning (BSP)

A binary space partitioning (BSP) [10] divides two- or three-dimensional space into a fixed number of pieces. BSPs are used in the connected components labeling algorithm described in this chapter to determine if a component on one PE abuts a component on another PE (meaning they are both actually part of a single, larger component). The BSP is constructed so that there is a one-to-one correspondence between the PEs and the pieces of the BSP tree. Explicitly, if there are N PEs, then the BSP will partition space into N pieces and each PE will be responsible for one piece. The PEs then relocate their cells according to the BSP; each cell is assigned a piece from the BSP based on its partition, and then that cell is sent to the corresponding PE.

It is important that the BSP is balanced, meaning that each piece has approximately the same number of cells. If disproportionately many cells fall within one piece, then its PE may run out of memory when the cells are relocated. As a result, the PEs must examine the cells and coordinate when creating the BSP.

The BSP construction and cell relocation can be very time consuming. More discussion of their complexity can be found at the end of this chapter (Sect. 6).

3.3 Exterior Cells and Ghost Cells

Exterior cells and ghost cells are used by the algorithm to reduce the amount of data needed to coordinate between PEs. Both techniques identify cells that are on the boundary of a PE's piece. Ghost cells identify exactly the cells on the boundary, while exterior cells identify a superset of the boundary cells.

Exterior cells are the cells that lie along the exterior of a volume, which does not necessarily strictly correspond to the exterior of the PE's piece. Consider the example of removing a material: the exterior cells of the remainder will likely have a portion along the PE piece boundary, but it will also likely have a portion along the interior of the piece, where the material interface lies.

"Ghost cells" are the result of placing a redundant layer of cells along the boundary of each domain. Ghost cells are either pre-computed by the simulation code and stored in files or calculated at run-time by the analysis tool. They are typically created to prevent interpolation artifacts at piece boundaries. More discussion of ghost cells can be found in [7] and [12].

Ghost cells are also useful for connected components labeling. They identify the location of the boundary of a piece and provide information about the state of abutting cells in a neighboring piece. Note that the results discussed in this chapter uses ghost cells that are generated at run-time, using the collective pattern described in [7], not the streaming pattern described in [12].

4 Algorithm

The algorithm identifies the global connected components in a mesh using five phases. It first identifies which pieces are at the boundary (Phase 1). It then identifies the connected components local to each PE (Phase 2) and then creates a global labeling across all PEs (Phase 3). It next determines which components span multiple PEs (Phase 4). Finally, it merges the global labels to produce a consistent labeling across all PEs (Phase 5). This final labeling is applied to the mesh to create per-cell labels which map each cell to the corresponding label of the connected component it belongs to. In terms of parallel considerations, Phases 1 and 2 are embarrassing parallel, Phase 3 is a trivial communication, Phase 4 has a large all-to-all communication, followed by embarrassingly parallel work, and Phase 5 has trivial communication following by more embarrassingly parallel work (Fig. 2).

Phase 1: Identify cells at PE boundaries: The goal of this phase is to identify cells that abut the spatial boundary of the data contained on each PE, which

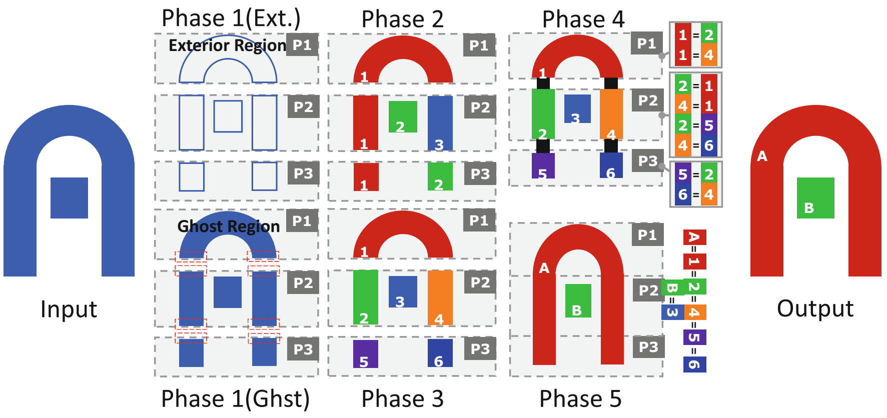


Fig. 2 Example illustrating the five phases of the algorithm on a simple data set decomposed onto three PEs. Phase 1 has two variants, and both variants are shown—"exterior cells" on the *top* and "ghost cells" on the *bottom*

enables reduced communication in Phase 4. We consider two methods for doing this: ghost data and exterior cells. The ghost data option marks the minimum number of cells to be considered, since ghost data always lies along the PE boundary. The exterior cells option marks more cells, since some cells are external to a component, but interior to the PE boundary; these cells cannot be distinguished and thus must be processed unnecessarily.

Ghost cells are not present in all data sets. The algorithm deployed in VisIt uses the ghost data option when ghost data is present, and falls back to the exterior cells option when it is not. However, we point out that the study described in this chapter shows the two variants to have very similar performance.

Phase 1, ghost cells option: Ghost cells are useful because they are always adjacent to boundary cells; finding the cells adjacent to ghost cells is equivalent to finding the list of cells on the boundary. Note that ghost cells cannot be used directly to represent PE boundaries since they themselves lack ghost data. For example, an isosurface operation on a ghost cell lacks the requisite additional ghost data to perform interpolation, and therefore does not have sufficient information to generate the correct contour. For this reason, all ghost cells are removed after the boundary is identified.

In pseudocode:

```
For each cell c:
  boundary[c] = false
  if (not IsGhostCell(c))
    For each neighbor n of c:
       if IsGhostCell(n):
            boundary[c] = true
RemoveGhostCells()
```

Phase 1, exterior cells option: Again, exterior cells are the cells that are on the exterior of the components. Only the cells on the boundary need to be considered in Phase 4, and these cells are a superset of the cells on the boundary. However, they are a subset of all cells and discarding the cells in the interior of the components substantially improves Phase 4 performance.

The benefit of this approach varies based on the data set. If a component has a high surface area to volume ratio, then proportionally less cells will be in the interior and the number of cells discarded is less. Further, the proportion of exterior cells that are not on the boundary compared to those that are on the boundary is data dependent. That said, a factor of 4× to 10× reduction is typical in the number of cells processed in Phase 4 by focusing on exterior cells.

The exterior cells can be calculated by using a standard "external faces" algorithm. For each face, look at the number of cells incident to that face (or each edge in two dimensions). The faces that have one cell incident to it are exterior, and so those cells are marked as exterior.

Phase 2: Identify components within a PE: The purpose of this phase is for each PE to label the connected components for its portion of the data. As mentioned in Sect. 3.1, the Union-find algorithm efficiently constructs a partition through an incremental process. A partition with one subset for each point in the mesh is used to initialize the Union-find data structure. It then traverses the cells in the mesh. For each cell, it identifies the points incident to that cell. Those points are then merged ("unioned") in the Union-find data structure.

In pseudocode:

```
UnionFind uf;
For each point p:
    uf.SetLabel(p, GetUniqueLabel())
For each cell c:
    pointlist = GetPointsIncidentToCell(c)
    p0 = pointlist[0]
    For each point p in pointlist:
        if (uf.Find(p0) != uf.Find(p))
            uf.Union(p0, p)
```

The execution time of this phase is dependent on the number of union operations, the number of find operations, and the complexity of performing a given union or find. The number of finds is equal to the sum over all cells of how many points are incident to that cell. Practically speaking, the number of points per cell will be small, for example eight for a hexahedron. Thus the number of finds is proportional to the number of cells. Further, the number of unions will be less than the number of finds. Finally, although the run-time complexity of the Union-find algorithm is nuanced, each individual union or find is essentially a constant time operation, asymptotically-speaking. Thus the overall execution time of this phase for a given PE is proportional to the number of cells contained on that PE.

Phase 3: Component re-label for cross-PE comparison: At the end of Phase 2, on each PE, the components within that PE's data have been identified. Each of these components has a unique local label and the purpose of Phase 3 is to transform these identifiers into unique global labels. This will allow the algorithm to perform parallel merging in subsequent phases. Phase 3 actually has two separate re-labelings. First, since the Union-find may create non-contiguous identifiers, it transforms the local labels such that the numbering ranges from 0 to N_P, where N_P is the total number of labels on Processing Element P. For later reference, we denote $N = \sum N_P$ as the total number of labels over all PEs. Second, the algorithm constructs a unique labeling across the PEs by adding an offset to each range. It does this by using the PE rank and determining how many total components exist on lower PE ranks. This number is then added to component labels. At the end of this process, PE 0 will have labels from 0 to $N_0 - 1$, PE 1 will have labels from N_0 to $N_0 + N_1 - 1$ and so on. Finally, a new scalar field is placed on the mesh, associating the global component label with each cell.

Phase 4: Merging of labels across PEs:

At this point, when a component spans multiple PEs, each PE's sub-portion has a different label. The goal of Phase 4 is to identify that these sub-portions are actually part of a single component and merge their labels. The algorithm does this by re-distributing the data using a BSP (see Sect. 3.2) and employing a Union-find strategy to locate abutting cells that have different labels. The data communicated involves cells, including their current label from Phase 3, although only the cells that lie on the boundary are needed to locate abutments. The cells identified in Phase 1 are used in the search process, but the cells known not to be on the boundary are excluded, saving about an order of magnitude in the number of cells considered.

The Union-find strategy in Phase 4 has four key distinctions from the strategy described in Phase 2:

- The labeling is now over cells (not points), which is made possible by the scalar field added in Phase 3.
- The algorithm now merges based on cell abutment, as opposed to Phase 2, where cells were merged if it had two points incident. This abutment captures any spatial overlap, be it at a face, a vertex, or one cell "poking" into another.
- Each cell is initialized with the unique global identifier from the scalar field added in Phase 3, as opposed to the arbitrary unique labeling imposed in Phase 2.
- Whenever a union operation is performed, it records the details of that union for later use in establishing the final labeling.

In pseudocode:

```
CreateBSP()
UnionFind uf;
For each cell c:
  uf.SetLabel(c, label[c])
For each cell c:
  For each neighbor n of c:
       if (uf.Find(c) != uf.Find(n))
            uf.Union(n, c)
            RecordMerge(n, c)
```

After the union list is created, the re-distributed data is discarded and each PE returns to operating on its original data.

Phase 5: Final assignment of labels: Phase 5 incorporates the merge information from Phase 4 with the labeling from Phase 3. Recall that in Phase 3 the algorithm constructed a globally unique labeling of per-PE components and denoted N as the total number of labels over all PEs. The final labeling of components is constructed as follows:

- After Phase 4, each PE is aware of the unions it performed, but not aware of unions on other PEs. However, to assign the final labels, each PE must have the complete list of unions. So Phase 5 begins by broadcasting ("all-to-all") each PE's unions to construct a global list.
- Create a Union-find data structure with N entries, each entry having the trivial label.

```
UnionFind uf
For i in 0 to N−1:
   uf.SetLabel(i, i)
```

- Replay all unions from the global union list.

```
For union in GlobalUnionList:
   uf.Union(union.label1 , union.label2)
```

  The Union-find data structure can now be treated as a map. Its "Find" method transforms the labeling we constructed in Phase 3 to a unique label for each connected component.
- Use the "Find" method to transform the labeling from the scalar array created in Phase 3 to create a final labeling of which connected component each cell belongs to.

```
For each cell c:
  val[c] = uf.Find(val[c])
```

- Optionally transform the final labeling so that the labels range from 0 to $N_C - 1$, where N_C is the total number of connected components.

Note that the key to this construction is that every PE is able to construct the same global list by following the same set of instructions. They essentially "replay" the merges from the global union list in identical order, creating an identical state in their Union-find data structure.

5 Performance Study

The efficiency of the algorithm was studied with a performance study that used weak scaling on concurrency levels up to 2,197 cores (and 2,197 PEs) with data set sizes up to 21 billion cells. The study used Lawrence Livermore National Laboratory's

"Edge" machine, a 216 node Linux cluster with each node containing two 2.8 GHz six-core Intel Westmere processors. The system has 96 GB of memory per node (8 GB per core) and 20 TB of aggregate memory.

5.1 Problem Setup

The data input came from a core-collapse supernova simulation produced by the Chimera code [4]. This data set was selected because it contains a scalar entropy field with large components that span many PEs. A data set was generated for each concurrency, using upsampling to ensure each PE would operate on a fixed number of cells. Interval volumes—the volume that lies between two isosurfaces, one with the "minimum" isovalue and one with the "maximum" isovalue—were extracted from the upsampled structured grid to create an unstructured mesh as input to the connected components algorithm.

The following factors were varied:

- Concurrency (12 options): Levels varied from 8 cores (2^3) to 2,197 cores (13^3).
- Data sets (2 options): Data sizes with one million cells per PE and 10 million cells per PE were run. Table 1 outlines the data sizes for the latter case.
- Phase 1 Variant (three options): Both the ghost cell and exterior cells variants of the algorithm were tested, as well as a variant with no identification of cells at PE boundaries (i.e., no Phase 1), since this variant was presented in previous work.

The cross product of tests were run, meaning $12 \times 2 \times 3 = 72$ tests.

Figure 1 shows rendered views of the largest interval volume data set used in the scaling study and its corresponding labeling result.

Table 1 Scaling study data set sizes for the runs with 10 million cells per PE. The study targeted PE counts equal to powers of three to maintain an even spatial distribution after upsampling. The highest power of three PE count available on the test system was $13^3 = 2197$ PEs, so PE counts from 8 to 2,197 and initial mesh sizes from 80 million to 21 billion cells were studied. The interval volume operation creates a new unstructured mesh consisting of portions of approximately 1/8th of the cells from the initial mesh, meaning that each core has, on average, 1.2 million cells

Num cores	Input mesh size	Interval vol. mesh size	Num cores	Input mesh size	Interval vol. mesh size
$2^3 = 8$	80 million	10.8 million	$8^3 = 512$	5.12 billion	621.5 million
$3^3 = 27$	270 million	34.9 million	$9^3 = 729$	7.29 billion	881.0 million
$4^3 = 64$	640 million	80.7 million	$10^3 = 1{,}000$	10 billion	1.20 billion
$5^3 = 125$	1.25 billion	155.3 million	$11^3 = 1{,}331$	13.3 billion	1.59 billion
$6^3 = 216$	2.16 billion	265.7 million	$12^3 = 1{,}728$	17.2 billion	2.06 billion
$7^3 = 343$	3.43 billion	418.7 million	$13^3 = 2{,}197$	21.9 billion	2.62 billion

5.2 Results

Figures 3 and 4 present the timing results from the cross product of tests. As expected, the timings for Phases 2, 3, and 5 are consistent between all variants of the algorithm. At 125 PEs and beyond, the largest subset of the interval volume on a single PE approaches the maximum size, either 1 million or 10 million cells depending on the study. For this reason, weak scaling for Phase 2 is expected. This is confirmed by flat timings for Phase 2 beyond 125 PEs. The ghost cell variant and exterior cell variant perform comparably in Phase 4, and both significantly outperform the variant with no boundary selection. These timings demonstrate the benefit of identifying per-PE spatial boundaries. The small amount of additional

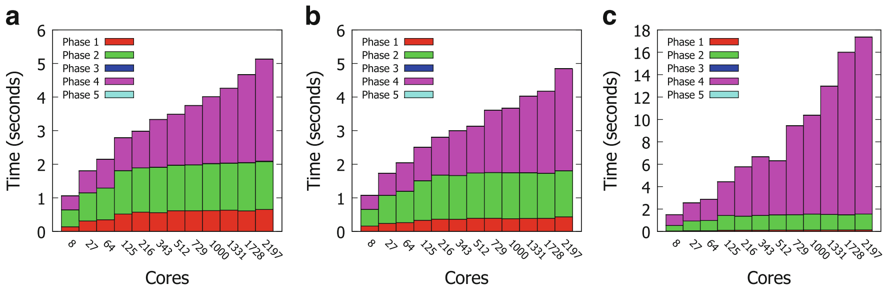


Fig. 3 Scaling study using one million cells per PE. Each figure corresponds to a variant for running Phase 1 and plots the timings for the five phases for each of the 12 concurrency levels for that variant. (**a**) shows the ghost cells variant. (**b**) shows the exterior cells variant. (**c**) shows the variant with no reduction of cells exchanged, which was presented in previous work and is included for comparative purposes. Figures **a** and **b** are very similar in performance and are on similarly scaled axes. Figure **c** performs significantly slower and is on a different scale. The time spent in Phase 1 for the ghost cell and exterior cell variants—which is not present in the third variant—leads to substantial savings in Phase 4. Phases 3 and 5 are negligible across all versions of the algorithm

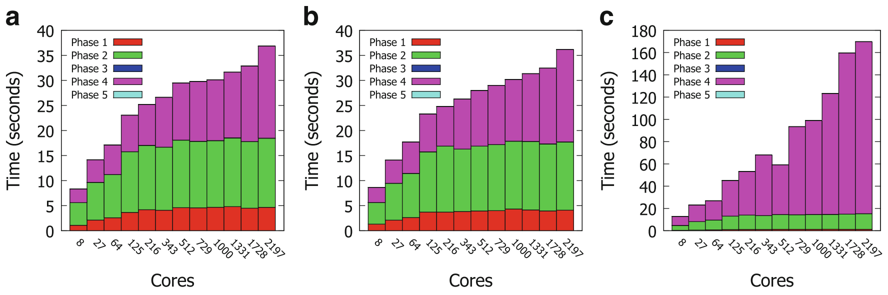


Fig. 4 Scaling study similar to that described in Fig. 3, except using 10 million cells per PE. As expected, the performance is proportional for the unoptimized variant of Phase 1, and nearly proportional for the two optimized variants

Table 2 Cells exchanged in Phase 4 for each variant for the 10 million cell per PE test variant. The total listed for each variant is the percentage of the total number of cells for that concurrency level

Number of cores	Total cells	Ghost cell variant (%)	Exterior cell variant (%)	No optimization (%)
$2^3 = 8$	10.8M	1.2	11.1	100
$3^3 = 27$	34.9M	2.3	9.1	100
$4^3 = 64$	80.8M	2.4	7.7	100
$5^3 = 125$	155M	2.6	6.9	100
$6^3 = 216$	266M	2.6	6.2	100
$7^3 = 343$	419M	2.7	5.7	100
$8^3 = 512$	622M	2.7	5.4	100
$9^3 = 729$	881M	2.7	5.1	100
$10^3 = 1{,}000$	1.2B	2.7	4.9	100
$11^3 = 1{,}331$	1.6B	2.7	4.6	100
$12^3 = 1{,}728$	2.1B	2.7	4.5	100
$13^3 = 2{,}197$	2.6B	2.7	4.3	100

Table 3 Information about the largest component and about the number of global union pairs transmitted in Phase 5. There is a strong correlation the two, and the Pearson correlation coefficient between them is 99.4 %. The percentage of cores spanned by the largest component converges to slightly less than 25 %

Num cores	Num cells in largest comp.	Num cores spanned	Num global union pairs
$2^3 = 8$	10.1 million	4	16
$3^3 = 27$	132.7 million	17	96
$4^3 = 64$	176.7 million	29	185
$5^3 = 125$	146.6 million	58	390
$6^3 = 216$	251.2 million	73	666
$7^3 = 343$	396.4 million	109	1,031
$8^3 = 512$	588.9 million	157	1,455
$9^3 = 729$	835.5 million	198	2,086
$10^3 = 1{,}000$	11.14 billion	254	2,838
$11^3 = 1{,}331$	11.51 billion	315	3,948
$12^3 = 1{,}728$	11.96 billion	389	5,209
$13^3 = 2{,}197$	12.49 billion	476	6,428

preprocessing time required for Phase 1 creates significant reduction in the number of cells transmitted and processed in Phase 4, as shown in Table 2.

Although the amount of data per PE is fixed, the number of connectivity boundaries in the interval volume increases as the number of PEs increases. This is reflected by the linear growth in both the number of union pairs transmitted in Phase 5 and the number of cores spanned by the largest connected component (See Table 3).

6 BSP Generation

Phase 4 is the slowest part of our algorithm, and BSP generation is a significant portion of that time. In this section, we consider the techniques and performance considerations for BSP generation. Section 6.1 describes Recursive Coordinate Bisection (RCB), a technique for generating BSPs. RCB requires a data structure for doing spatial searches; Sect. 6.2 explores the relative advantages of octrees and interval trees.

6.1 Recursive Coordinate Bisection (RCB)

RCB [14] is an algorithm that takes a list of points and a target number of regions and generates a BSP that partitions space such that every region in the BSP contains approximately the same number of points. The list of points is distributed across the PEs, so the RCB algorithm must operate in parallel. Again, in this context, the target number of partitions is the number of PEs, so that each PE can own one region. It is important that each region contains approximately the same number of points, otherwise a PE might receive so many points that it will run out of memory.

RCB starts by choosing a "pivot" to divide space. In the first iteration, the pivot is a plane along the x-axis (e.g., "X = 2"). The pivot should divide space such that half of the point list is on either side of the plane. The algorithm then recurses. It embarks to find a plane in the y-axis for each of the two regions created by the initial split. These planes may be at different y locations. The algorithm continues iterating over the regions, splitting over X, then Y, then Z, then X again, and so on. At each step, it tries to split the region so that half of the points are on each side of the plane (with modifications for non-powers of two). This process is illustrated in Fig. 5.

A key issue for RCB is pivot selection. The pivot selection requires iteration, with each proposed pivot requiring examination of how many points lie on either side. Previous RCB constructions [14] have used randomized algorithms. These

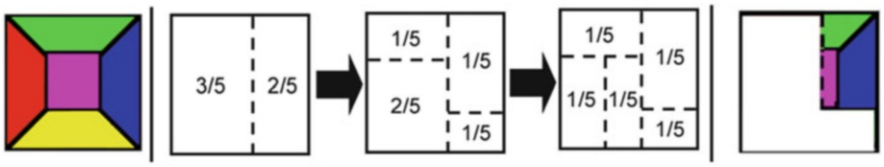


Fig. 5 RCB construction of a BSP-tree in a distributed memory setting. On the *left*, the decomposition of the original mesh. Assume the *red* portions are on PE 1, *blue* on 2, and so on. The iterative strategy starts by dividing in X, then in Y, and continues until every region contains approximately $1/N_{PEs}$ of the data. Each PE is then assigned one region from the partition and the data is communicated so that every PE contains all data for its region. The data for PE 3 is shown on the *far right*

Table 4 Comparison of communication patterns for randomized RCB and the modified "2-Pass" RCB scheme used in this study

Factor	Randomized RCB	2-Pass RCB
Num. passes	N	2
Num. communications per pass	1	1
Num. searches per pass	1	5
Total	N communications,	2 communications,
	N searches	10 searches

Table 5 Breakdown of balance of points in BSP created by "2-Pass" RCB method for run with 2,197 processors

Problem	Maximum	Average	Minimum
2,197 pieces with 2.2B cells	1.3M	1.0M	805K
2,197 pieces with 10.9B cells	6.4M	4.9M	3.8M

The maximums are no more than 30 % bigger than average and the minimums are no more than 25 % smaller than average. These inequities were deemed desirable in the context of reduced parallel communication

algorithms choose a pivot at random, and then restrict the points considered to be those on the majority side of the pivot.

Although it is possible to iterate on pivot location until equal numbers of points lie on either side of the pivot, it is not practical. Each pivot candidate requires parallel coordination. And a pivot that includes the balance by a negligible amount will not substantially improve further processing in Phase 4. So, to make RCB execution go faster, the requirement of a balanced spatial partition was relaxed. As a result, the points in each region were unbalanced. The effective tradeoff is decreased RCB execution time versus increased time later in Phase 4 by the PEs that got more than the average number of points.

The algorithm presented in this chapter used a modified version of RCB (not a randomized version). It chooses five potential pivot points that are evenly spaced through the volume. It then identifies the pair of pivots that contain the ideal pivot and places another five evenly spaced pivots between them. The best choice of those five is the pivot. This method works well in practice, because it minimizes communication, as the five pivot selections can be communicated simultaneously. Table 4 analyzes the communication time, while Table 5 shows the effective balance of the modified "2-Pass" RCB scheme on a large problem.

6.2 *Octrees and Interval Trees*

For each pivot candidate, the algorithm must determine how many points lie on either side of its dividing plane. If there are N PEs, then $N-1$ pivots must be chosen (this can be thought of as the number of interior nodes in a binary tree). Each pivot likely involves some number of candidate pivots (10 in the case of this algorithm). If done poorly, this would involve $(10N - 10) \times O(num\,points)$ comparisons. Placing the points in a search structure can substantially reduce the cost for checking on the quality of a pivot candidate.

There are many data structures for spatial searches [9]; this section considers the tradeoffs between octrees and balanced interval trees, comparing the speed of searches on two data sets meant to model the best and worst case scenario for an interval tree. Since the interval tree is derived by data points and not spatially, ordering of data has a profound effect on the efficiency of the model. (Note that sorting the data points spatially considerably increases overall initialization time.) The octree is spatially created, and will, therefore, always create the same outcome whether the data is sorted or unsorted.

We ran our initial tests against two data sets. Both data sets contained one million points evenly spread across a rectilinear grid with one data set being sorted and the other data set unsorted. The test involved running the octree and interval tree models against five different searches using both types of input data. The first search covered the entire bounding box. Each of the following four searches decreased the bounding size by half of the previous size. Again, this technique was designed to measure the best case search for the octree to the worst case search for the octree, given its early termination criteria. Results are shown in Fig. 6.

Although the first tests were fairly conclusive, additional tests were performed on real world data, namely a data set operated on by one PE from the scaling study. Results are shown in Fig. 7. The results for the tests show that the octree maintains a consistent and very quick search time with low variation, on the order of ten thousandths of a second. The interval tree however performed quite poorly in some cases, on the order of hundredths of a second, and excelled in others, reaching one ten thousandth of a second. The interval tree's quick regions were identified as those which did not contain any data. The interval tree was able to use an early termination scheme to quickly decide if there were no points in the search bounds.

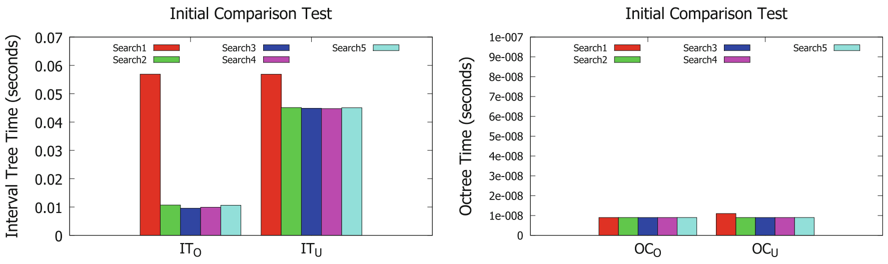
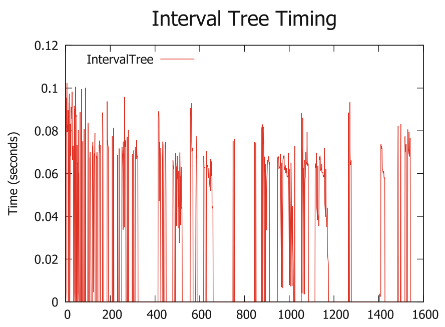


Fig. 6 Interval Tree (IT) tests are on the *left*, Octree (OC) tests are on the *right*, each with ordered (o) and unordered (u) variants. Searches #1–#5 are from largest bounding box to smallest, respectively. The run time for octree was significantly faster (note the axes have different scales), and was sometimes less than the precision of the timing instrumentation



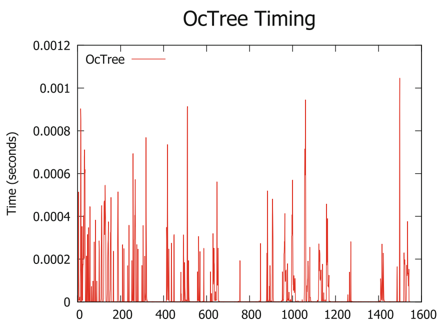


Fig. 7 The tree structures were evaluated using data from one processor of a 128-processor test. Each of the 1,548 searches on that data were recorded and laid out along the X-axis. The Y-axis for each graph is the time (in seconds) for the corresponding search structure to complete the requested search. Note that the scale varies between the graphs, since the octree is significantly faster

But when it could not terminate early, it spent a good deal of time adding up the points that were within the range. The octree, on the other hand, was able to use early termination criteria to deal with these cases. These tests clearly solidified the octree as the better of the two data structures in question for this task, which had a run time similar to the interval tree's best case for every case.

7 Summary

This chapter described a distributed-memory parallel algorithm that identifies and labels the connected components in a domain-decomposed mesh. The labeling produced by the algorithm provides a topological characterization of a data set that enables important analyses. The algorithm is designed to fit well into currently deployed distributed-memory visualization tools, and this was demonstrated in a scaling study.

Acknowledgements This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract DE-AC02-05CH11231, was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, and was also supported in part by the National Science Foundation, grants OCI-0906379 and OCI-0751397.

References

1. Ahrens, J., Geveci, B., Law, C.: Visualization in the ParaView framework. In: Hansen, C., Johnson, C. (eds.) The Visualization Handbook, pp. 162–170. Elsevier Butterworth-Heinemann, Burlington (2005)

2. Bennett, J.C., Krishnamoorthy, V., Liu, S., Grout, R.W., Hawkes, E.R., Chen, J.H., Shepard, J., Pascucci, V., Bremer, P.-T.: Feature-based statistical analysis of combustion simulation data. IEEE Trans. Vis. Comput. Graph. **17**, 1822–1831 (2011)
3. Bremer, P.-T., Weber, G., Tierny, J., Pascucci, V., Day, M., Bell, J.: Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. IEEE Trans. Vis. Comput. Graph. **17**, 1307–1324 (2011)
4. Bruenn, S.W., Mezzacappa, A., Hix, W.R., Blondin, J.M., Marronetti, P., Messer, O.E.B., Dirk, C.J., Yoshida, S.: Mechanisms of core-collapse supernovae and simulation results from the CHIMERA Code. In: CEFALU 2008, Proceedings of the International Conference. AIP Conference Proceedings, pp. 593–601 (2008)
5. Carr, H., Snoeyink, J., Axen, U.: Computing contour trees in all dimensions. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 918–926. Society for Industrial and Applied Mathematics, Philadelphia (2000)
6. Childs, H., Pugmire, D., Ahern, S., Whitlock, B., Howison, M., Prabhat, M., Weber, G., Bethel, E.W.: Extreme scaling of production visualization software on diverse architectures. IEEE Comput. Graph. Appl. **30**, 22–31 (2010)
7. Childs, H., Brugger, E., Whitlock, B., Meredith, J., Ahern, S., Bonnell, K., Miller, M., Weber, G.H., Harrison, C., Pugmire, D., Fogal, T., Garth, C., Sanderson, A., Bethel, E.W., Durant, M., Camp, D., Favre, J.M., Rübel, O., Navrátil, P., Wheeler, M., Selby, P., Vivodtzev, F.: VisIt: an end-user tool For visualizing and analyzing very large data. In: Proceedings of SciDAC 2011, July 2011
8. Cormen, T.H., Stein, C., Rivest, R.L., Leiserson, C.E.: Introduction to Algorithms, 2nd edn. McGraw-Hill Higher Education, New York (2001)
9. Edelsbrunner, H.: Dynamic Data Structures for Orthogonal Intersection Queries. Institut für Informationsverarbeitung, Technische Universität Graz (1980)
10. Fuchs, H., Kedem, Z.M., Naylor, B.F.: On visible surface generation by a priori tree structures. In: Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques, pp. 124–133. ACM, New York (1980)
11. Harrison, C., Childs, H., Gaither, K.P.: Data-parallel mesh connected components labeling and analysis. In: EuroGraphics Symposium on Parallel Graphics and Visualization (EGPGV), pp. 131–140, April 2011
12. Isenburg, M., Lindstrom, P., Childs, H.: Parallel and streaming generation of ghost data for structured grids. IEEE Comput. Graph. Appl. **30**, 32–44 (2010)
13. Morozov, D., Weber, G.: Distributed merge trees. In: Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 93–102 (2013)
14. Nakhimovski, I.: Bucket-based modification of the parallel recursive coordinate bisection algorithm. Linköping Electron. Articles Comput. Inf. Sci. **2**(15), 2–32 (1997)
15. Schroeder, W.J., Martin, K.M., Lorensen, W.E.: The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In: Proceedings of the 7th Conference on Visualization '96 (VIS '96), pp. 93-ff. IEEE Computer Society Press, San Francisco (1996)
16. Tarjan, R.E.: Efficiency of a good but not linear set union algorithm. J. ACM (1975). doi: 10.1145/321879.321884
17. Tierny, J., Gyulassy, A., Simon, E., Pascucci, V.: Loop surgery for volumetric meshes: reeb graphs reduced to contour trees. IEEE Trans. Vis. Comput. Graph. **15**, 1177–1184 (2009)
18. Wang, Y., Yu, H., Ma, K.-L.: Scalable parallel feature extraction and tracking for large time-varying 3D volume data. In: Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), pp. 17–24 (2013)

In-Situ Visualization in Computational Fluid Dynamics Using Open-Source tools: Integration of Catalyst into *Code_Saturne*

Alejandro Ribés, Benjamin Lorendeau, Julien Jomier, and Yvan Fournier

1 Introduction

Computational Fluid Dynamics (CFD) is a fundamental step for the study and optimization of electricity production. Indeed, current power plants use water as a mean of convective heat transfer. Consequently, the simulation and visualization of fluid dynamics phenomena is of great importance for the energy industry. Electricit de France (EDF), one of the largest electricity producer in Europe, has been developing for the past 15 years an open source CFD code named *Code_Saturne*. *Code_Saturne* performs CFD computations on very large models [15]. EDF owns several supercomputers that regularly run this code in order to perform CFD analysis involving large amounts of data. In this context, the post-processing and visualization steps become critical.

EDF also develops, in collaboration with OpenCascade and the French Center of Atomic Research (CEA), an open-source numerical simulation platform called SALOME [12]. This platform provides generic methods for pre- and post-processing of numerical simulations. SALOME is based on an open architecture made of reusable components such as computer-aided design (CAD), meshing, high performance computing (HPC) execution management, multi-physics coupling, data post-processing and visualization. The visualization module of the SALOME

A. Ribés (✉) • B. Lorendeau
EDF R&D, Clamart, France
e-mail: alejandro.ribes@edf.fr; alejandro.ribes@gmail.com; benjamin.lorendeau@gmail.com

J. Jomier
Kitware, Villeurbanne, France
e-mail: julien.jomier@kitware.com

Y. Fournier
EDF R&D, Chatou, France
e-mail: yvan.fournier@edf.fr

platform is currently based on the open-source post-processing platform ParaView. Furthermore, *Code_Saturne* is often used in conjunction with the SALOME platform.

In the past, studies and improvements in scientific simulation have been mainly focused on the solver, due to being the most cycle-consuming part in the simulation process. Thus, visualization has been traditionally run sequentially on a smaller computer and at the very end of the solver computation. At the time, this was easily explained by the small need for both memory and computation resources in most of the visualization cases. Nevertheless, with the increase of our computational capabilities, we tend to use and generate much more data than what we were used to. Thus, as the scale of CFD simulation problems is getting wider, specific issues are emerging related to input/output efficiency. In particular, data generated during the solver computation and used for the visualization are the source of a worrisome overhead. Even worse, some researchers are starting to spend more time writing and reading data than actually running solvers and visualizations [13]. This new trend compels us to design new input/output (I/O) strategies and consider visualization as a part of our high-performance simulation systems.

For some years, in-situ visualisation techniques have been successfully applied in different contexts and mainly by research institutes. In this chapter, we present an overview of the efforts needed to transition a traditional simulation code to an in-situ model in an industrial environment. This is the reason why care have been taken constructing uses cases that are representative of our current visualisation problems.

Most fluid dynamic engineers at EDF R&D are currently visualizing lower temporal and spatial resolution versions of their simulations in order to avoid I/O bottlenecks when large quantities of data are involved. We decided to address the subject of co-processing and in-situ visualization which has been proved to be an effective solution against the current limitations of this problem [11, 14]. Our aim is to provide EDF engineers with an operational research-oriented tool in a mid-term basis. For this, we chose to evaluate Catalyst as an industrial tool for performing in-situ visualization. Catalyst is a library, developed by Kitware, which implements the co-processing for ParaView by defining the visualization process through the ParaView user interface and exploiting VTK's parallel algorithms for the post-processing of data generated by numerical simulation [8].

In this chapter, we report a study upon the effectiveness and scalability of a prototype implementation of the co-processing in an industrial case based on the coupling of *Code_Saturne* with Catalyst. In Sect. 2 we introduce the motivation of this work. In Sect. 3 we discuss related advances on recent visualization in-situ systems. We then introduce, in Sect. 4 *Code_Saturne*, the CFD code developed at EDF R&D. In Sect. 5 we present our integration of Catalyst into *Code_Saturne* and how the system is used by EDF users in the context of fluid dynamic simulations. Section 6 describes our use case and presents results on one of our corporate clusters. Finally, section "Conclusion" presents our analysis of the results and describes our ongoing and future work.

2 Motivation

Most numerical simulation engineers at EDF R&D are currently visualizing lower temporal and spatial resolution versions of their simulations, in order to avoid I/O issues and cumbersome visualisation procedures, when large quantities of data are involved. We believe that other industries dealing with large simulations are having the same problem. This is the reason why we decided to leverage the power of co-processing and in-situ visualization. Our aim is to provide our research-oriented engineers with an operational tool within 2 years. Thus, we have evaluated Catalyst as an industrial tool for performing in-situ visualization.

First of all, it is important to better describe the scope of our industrial visualisation solutions to understand why in-situ processing is needed. In Table 1 we show the results of a simple subjective experiment conducted by one of our engineers. At the end of 2012, she meshed a simple cube at different resolutions and then tried to visualise the results giving a subjective evaluation of how she could work. She used our open-source numerical simulation platform, SALOME, and a standard scientific PC with 8Gb of RAM. Table 1 presents the results of her subjective experiment. The study clearly shows that she started working without an immediate system response for meshes which contain more that 10 millions cells and for 50 million cells the system was not responding at all. At the time that this test was performed, some of our R&D engineers were already running simulations with meshes containing around 200 millions cells and, in June 2014, with meshes reaching 400 millions cells. This implies that copying the simulation results from the simulation cluster to the scientific stations is not practical, first because of the long transfer time and second because, as Table 1 shows, the visualization and post-processing tasks cannot even run. It clearly appears that the visualization and post-processing of large meshes is a serious bottleneck in this industrial context. This motivated the beginning of this work.

A first solution to the post-processing bottleneck consists in the installation of parallel visualisation servers that can deal with the large amount of data generated by the numerical simulations. In general, such a system (in our case a ParaView "pvserver") is installed on a visualisation cluster; the system reads the data from disk and performs the visualisation operations in parallel, while streaming images (or small 3D models) to a remote client hosted on a standard PC. EDF R&D owns

Table 1 Subjective characterization of the reaction time of the SALOME platform for different mesh sizes

Mesh size manipulation experiment					
Number of cells	10 thousands	100 thousands	1 millions	10 millions	50 millions
RAM(%)	<50%	<50%	<50%	100%	Saturated
Reaction time	Immediate	Immediate	2–3 s	Uncomfortable	Not responding

a visualisation cluster as part of its HPC cluster “Ivanhoe”, which will be described later in this chapter. This type of solution implies writing and reading large data in parallel. Even if these operations are performed on a cluster with a fast distributed file system, in-situ processing provides much better performances as large resulting datasets are potentially never generated.

In order to get more insight, we can model the whole time taken by simulation and visualisation tasks as an addition of individual operations. For the traditional a posteriori visualisation approach:

$$t_{posterior} = T_s + T_w + T_r + T_v \tag{1}$$

where T_s is the simulation time, T_w is the time for writing the data, T_r is the time to read the data (either in parallel or sequentially) and T_v is the time to perform visualization operations and probably write visualisation results (like videos, images or graphs). For the in-situ approach:

$$t_{in-situ} = T_s + T_{process} + T_{w-in-situ} + T_{v-in-situ} \tag{2}$$

where T_s is the simulation time (the same as in $t_{posterior}$), $T_{process}$ is the time to perform the visualisation operations in-situ, $T_{w-in-situ}$ the time to store the already processed visualisation results and $T_{v-in-situ}$ the time that the engineer takes to visualize the videos or other pre-processed data. Comparing these two formulas we can see that $t_{posterior} >>> t_{in-situ}$ as, in the case of in-situ, we skip writing and reading large volumes of data, $T_w + T_r >>> T_{process} + T_{w-in-situ}$; but also the visualisation time is reduced $T_v >>> T_{v-in-situ}$ because, in the a posteriori approach, visualising means performing operations on large data while in the in-situ approach only lightweight data is involved. In the rest of this chapter these times will be exemplified. For instance, in the top two images of Fig. 4 one can compare T_w and $T_s + T_w$ for different simulations and the relationship $T_w + T_r >>> T_{process} + T_{w-in-situ}$ becomes clear. These two images demonstrate how quickly I/O times widen, relative to solver times, which is why in-situ techniques are needed.

In conclusion, the whole process of “simulation + visualisation” is faster when performed in-situ, furthermore the volume of the produced data in much smaller. This is the reason that motivated this work.

3 Related Work

The size of generated data has become an important subject in high performance computing, due to the need of a better I/O efficiency in our computing system. To answer this problem, several visualization systems have been created. We can distinguish two main approaches in recent solutions. The first one is to integrate a specific in-situ visualization directly to the simulation code. Such approach proved to be an efficient way to provide co-processing for a given simulation as well as a visualization system as it is the case in the hurricane prediction [6] and earthquake simulation [14] systems. This method has been proven to lead to good performances but is limited to a specific implementation.

The second approach is to provide a general post-processing framework letting the simulation and the visualization code communicate together. EPSN which is a general coupling system, allows for the connection of M simulation nodes to N visualization nodes through a network [7]. This solution is a loosely coupled approach, requiring separate resources and data transfer through the network. This approach presents the advantage of not overloading the nodes used for computation. Thus the visualization code does not interfere with the execution of the simulation. Based on the same approach, a ParaView plug-in named ICARUS [3] has been developed. It differs from EPSN in design by having lower requirements as it only needs the use of a single HDF5 library and file driver extension. Such solutions offer tools for researchers to interact with their simulations by allowing them, not only to monitor their current states but also to modify the parameters of the remaining simulation steps. Those computational steering solutions as well as the RealityGrid project [4] focus on interactivity with simulation whereas our main objective is to provide in-situ visualization operations to researchers while minimizing I/O overhead and disk space use.

Both built upon the well known parallel visualization library VTK, the application frameworks VisIt [5] and ParaView [9] both provide through the possibility to co-process simulation data via libsim [16] and Catalyst [8] respectively. Those in-situ solutions are tightly coupled and while they limit potential interactions with the running simulation, they also highly reduce the need of network data transfer. Thus, they contribute to circumventing the inefficiency of high performance computing I/O systems. Those solutions take their benefits from directly accessing the simulation memory to perform visualization tasks by simply asking for a pointer to the available data. One major drawback of this approach is the necessity to provide a coherent data layout to those libraries. Moreover, as this type of solution often gains from computing pre-determined visualization tasks, it is not well suited for results exploration. As building a steering solution for Code_Saturne is out of the scope of this case study, we do not consider these drawbacks as a limitation.

After evaluating the performance solutions offered by ParaView and VTK, we choose Catalyst as our co-processing library for our case study as it answers EDF's visualization needs while focusing on the reduce of data amount use. Further, Kitware recently included services allowing the interactions with the running simulation, the so-called Live Catalyst.

4 *Code_Saturne*: A Computational Fluid Dynamics Code

Code_Saturne is a computational fluid dynamics software designed to solve the Navier–Stokes equations in the cases of 2D, 2D axisymmetric or 3D flows. Development started in 1997, with a first release in 2000, and the code has been released as free software under a GPL (General Public Licence) licence since 2007. Its main module is designed for the simulation of flows which may be steady or unsteady, laminar or turbulent, incompressible or potentially dilatable, isothermal or not. Scalars and turbulent fluctuations of scalars can be taken into account. The code includes specific modules, referred to as "specific physical models", for the treatment of atmospheric flows, Lagrangian particle tracking, semi-transparent radiative transfer, gas combustion, pulverised coal combustion, electricity effects (Joule effect and electric arcs) and compressible flows. *Code_Saturne* relies on a finite volume discretisation and allows the use of various mesh types which may be hybrid (containing several kinds of elements) and may have structural non-conformities (hanging nodes). The parallelization is based on standard spatial partitioning with ghost cells that facilitate data passing between adjacent cells lying across the boundaries of disconnected parts using the Message Passing Interface. More technical details are presented in [1] and [2], and many resources are available at http://www.code-saturne.org. *Code_Saturne* is also used as a base for the NEPTUNE_CFD code, specialized in multiphase flows, and which uses a different time stepping scheme, but mostly the same volume discretization scheme.

As *Code_Saturne* is used for industrial cases involving complex flows, with turbulence modeling requiring sufficiently fine resolution, large meshes are often needed. In 2000, the largest meshes used for actual studies were around 1.5 million cells; today, they have reached up to 400 million cells. More common studies use meshes about ten times smaller than that. Meshes up to 3.2 billion cells have been tested for a few time steps, to ensure the code's internal mechanisms work well at scale.

Code_Saturne focuses on the solver, and its uses requires external tools for the major part of the meshing and visualisation tasks, though the code itself offers major preprocessing features to make these easier, such as parallel joining of independently-built and read submeshes (whether conforming or not), and user-definable post-processing functions (Fig. 1). Many input mesh and visualisation output formats are supported (including the EDF and CEA MED format, and the standardized CGNS format).

The number of separate executable tools is quite reduced, with a few interactive tools and associated commands designed for data setup.

To make the use of HPC as seamless as possible, without multiplying the number of tools or requiring complex libraries or dependencies, mesh, checkpoint/restart, and post-processing output files are partitioned independently: in addition to the connectivity of each local mesh, which is described with a local numbering, global ids are also maintained, for import and export purposes, and for ghost cell to local cell matching. Multiple ranks participate in reading and writing files using MPI-IO.

Typically, computational resource requirements are primarily determined either by the time to solution, or the size of the model. For time to solution, the number of cores may be selected in order to solve the problem in a given time. In practice, optimal performance is often obtained in a range of 30,000–60,000 cells per rank on a typical HPC cluster (this being the best compromise between communication latency and cache behavior). On machines with very good network/compute

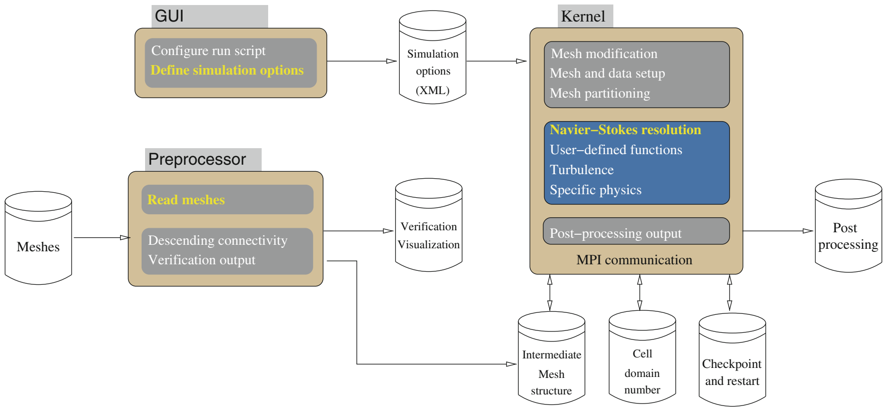


Fig. 1 *Code_Saturne* toolchain components

performance ratios, such as IBM Blue Genes or Cray X series, this range may be a bit wider.

5 Using Catalyst

Catalyst is the coprocessing library of ParaView. It has been designed to be tightly coupled with simulation codes to perform in situ analysis at run time. Catalyst leverages the Visualization Toolkit (VTK) for scalable data analysis and visualization. Furthermore, it can be coupled with the ParaView In Situ Analysis framework to perform run-time visualization of data extracts and steering of the data analysis pipeline. Catalyst provides two sets of tools: one for simulation users and one for simulation developers.

For simulation users, it is possible to create a coprocessing pipeline using two different methods. The first method does not require any knowledge of ParaView and relies on pre-generated scripts. These predefined scripts can be written in C++ or Python and are, usually, expected to run without any configuration options. The second method uses the ParaView interface to generate a coprocessing script from scratch and intuitively adjust its parameters as needed (Fig. 2). This method is similar to using ParaView interactively to setup desired post-processing pipelines. The goal of these pipelines is to extract post-processed information during the simulation run. Ideally one should start with a representative dataset from the simulation. It is also possible to modify directly the generated Python's scripts which have been previously created using ParaView. However, this would require a knowledge of the ParaView Python application programming interface (API).

For simulation developers, Catalyst provides the tools to create an adaptor between the simulation code and the visualization pipeline. The adaptor binds

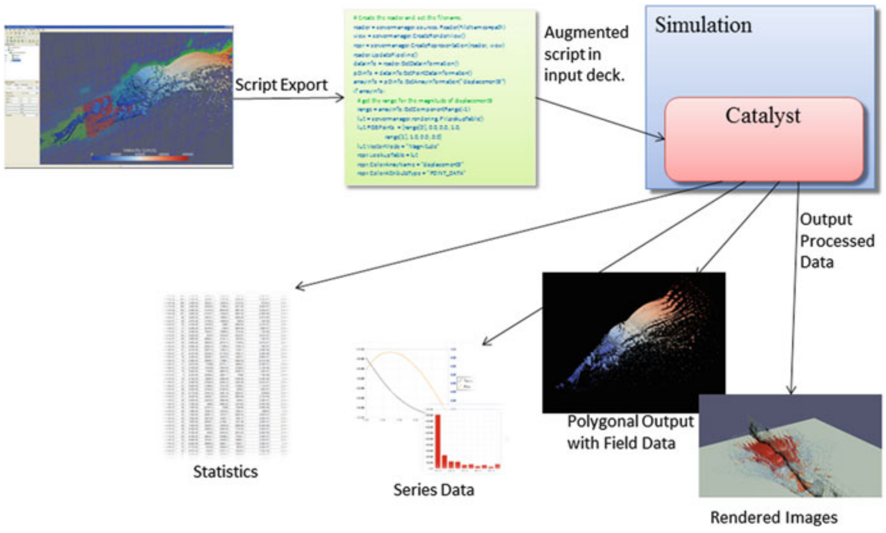


Fig. 2 Overall Catalyst workflow

the simulation code and Catalyst so that both the functions of the simulation code and the general-purpose API of Catalyst can be accessed. As Catalyst itself is independent of the simulation code, only the adaptor has to be developed by the designers of the solver. This flexibility is critical in order to successfully integrate external code into complex simulations usually running with different languages and packages. Catalyst is also easily extensible so that users can deploy new analysis and visualization techniques to existing coprocessing installations. Catalyst provides all the communication and synchronization routine and the pipeline mechanics necessary for coprocessing. Catalyst also provides powerful data processing capabilities through VTK filters as well as many writers and support for compositing and rendering.

The Catalyst library has also been developed with features to address limitations that come with pre-configuring a pipeline, but there may still be some unexpected data in the arbitrary simulation. To address these situations, the pipeline must be adjusted interactively. The Catalyst library can leverage ParaView's client server mechanism to allow an interactive ParaView client to connect to a server running inside an in situ pipeline. ParaView can then read from a Catalyst data source like it reads from a file. This enables construction/modification of a pipeline interactively in the ParaView client via this live data source. Additionally, by enabling the client-server architecture in the Catalyst library, some or all of the data analysis and visualization pipeline can offload, if desired, to a separate machine, e.g., a smaller visualization cluster with specialized graphics hardware.

5.1 Code_Saturne with Catalyst

Using Catalyst with *Code_Saturne* is quite straightforward, and fits quite naturally in the existing architecture. In this section we describe first how we implemented an adaptor for Catalyst and then we present how we configured the post-processing pipeline.

5.1.1 Adaptor Implementation

When embedding Catalyst there is always a non-zero cost in terms of time, memory and number of processors. Naively, one could simply address these issues by requesting a greater number of processors, but in most cases this is not possible nor practical. Therefore, great care must be taken in the implementation of the adaptors. If memory is limited, the adaptor either uses complicated pointer manipulation or uses a smaller region of memory. If memory is not limited, then deep copy of the simulation data structures into a VTK data object doubles the resident memory, and also creates a CPU cost involved with copying the data. The overriding overhead issue in embedding ParaView Catalyst in a simulation code is the memory management in an adaptor translating data structures in the running simulation.

In order to co-process the simulation data, Catalyst must be provided with the data formatted to the VTK data object structure. To accomplish this task, several solutions are possible, depending on the format used for the data of the simulation code. In the case where the format of the simulation code is similar to VTK and, moreover, the simulation data can be shared at any time, then it is possible to feed Catalyst with a direct pointer to the simulation memory. This option is indeed preferred, when possible, as it allows to decrease the memory footprint. Another option is to fully or partially copy the data from the simulation into a VTK object, and then send this object to Catalyst.

As users of *Code_Saturne* are provided with several output formats and as the data structure in our simulation differs from the VTK data object structure, feeding Catalyst with a direct pointer to the simulation memory is not possible. Thus, in this configuration data is copied from the simulation into a VTK data object. In fact, we allocate a vtkDoubleArray data structure to store our data for Catalyst. Furthermore, we provide a pointer of this VTK data structure to *Code_Saturne* so it can transform its simulation data and then fill the VTK data object.

The memory cost increase of our solution can be alleviated by using more machines. The CPU cost of the copy is in a range similar to the one needed when adapting simulation data to a specific output format. This cost is largely affordable comparatively to the time to write data to disk when storing time step specific outputs.

5.1.2 Pipeline Configuration

From the point of view of an engineer performing a fluid mechanics simulation using *Code_Saturne*, the workflow of a co-processing simulation is (1) to define a ParaView pipeline describing what the user wants to study and (2) to run the simulation. Since users are already familiar with fluid mechanics simulations, defining the pipeline for the co-processing remains the main sticking point. Thus this new process should be done in an efficient way and should not become a cumbersome bottleneck. This point is of great importance, especially in an industrial environment like ours.

As we have explained in the previous section, the definition of a Catalyst pipeline can be achieve either programmatically or via the ParaView interface. In our industrial context, the former was considered too complicated and time consuming for the end user, especially when setting camera parameters is needed, as no visualization feedback is provided. Therefore, the Catalyst pipeline has been created using the ParaView user's interface. This solution appears to be much easier as one can interact with ParaView in the same way he/she uses to when visualizing the results a posteriori. This solution is also easier to deploy with ParaView using its companion co-processing plugin.

Indeed, using ParaView to define the co-processing requires a representative input dataset. One could use the available resources on a large cluster in order to setup the pipeline. However we chose to provide a simplified or under-sampled version of the large geometry to define the pipeline. In fact, this strategy is possible in ParaView but some characteristics of the initial geometry must be present in its simplified version; more importantly the name of the data fields must remain the same, as they are part of the definition of the pipeline.

The generation of the co-processing pipeline implies several steps. First of all, the users start with a Computer Aided Design (CAD) version of the geometry which is parameterized. This parametric representation can generate meshes at different resolution levels. In our case, this is performed inside the open-source SALOME [12] platform for numerical simulation.

We then generate two different meshes, one at high resolution (up to 204M hexahedrals in the current use cases) that will be used for the CFD simulation and one with a lower resolution to define the pipeline (700,000 hexahedrals in our use cases). The lowest resolution mesh is fed into *Code_Saturne* to perform a short simulation. This allows ParaView to obtain a representation containing not only the geometry but also the result fields. This is the data that is then used to define the pipeline. The different processing pipelines are presented next.

6 Results

6.1 *Required User Interactions for Co-processing*

Before presenting our results we briefly describe how the user interactions were performed. The following steps were necessary in order to use the developed co-processing technology:

(1) A “light version” of the input mesh is generated as explained in Sect. 5.1.2. As the user possesses a CAD version of the geometry that is parameterized, it is then possible to obtain meshes at different spatial resolutions. A “light mesh” of small size in memory and representative of the CAD geometry is obtained. Figure 3a represents the “light version” of the mesh used in our experiments.
(2) A short simulation (normally just a few seconds on a local machine) on the “light mesh” is run. This simulation allows to define the information about the result fields needed to create a visualisation pipeline in ParaView (e.g. temperature, pressure, velocity). An “augmented light mesh” is therefore created.
(3) The mesh and the fields obtained at the end of step 2 are then read in ParaView and the user defines her/his visualisation pipeline. At the end of this step a simple click in the ParaView interface will create a Python file that programmatically defines the visualisation operations to be performed in-situ.
(4) Finally the real simulation is ran using a full resolution size mesh. The co-processing part of the simulation reads the python script containing the definition of the visualisation pipeline. This step is expected to be time-consuming.

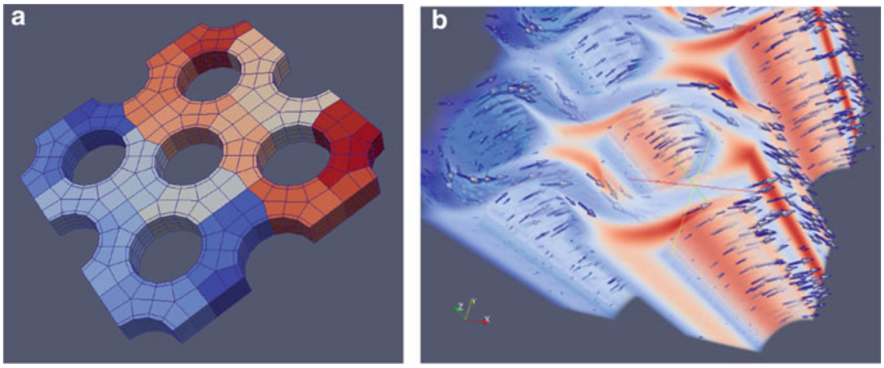


Fig. 3 (**a**) Original geometry of our use case. (**b**) A final coprocessed picture of our simulation

6.2 Use Cases

Our simulations have been run on Ivanoe, an EDF corporate supercomputer, composed of 1,382 nodes, each node having 12 cores for a total of 16,584 cores. In these simulations we associate one MPI process by core and we use from 720 cores up to 3,600 cores. We include two use cases that were run on this supercomputer. The choice of the cases is motivated by two main factors: the size of the mesh and the complexity of the visualization pipeline. We detail next these two impacting factors.

(1) Mesh size. We chose to use two meshes representing the same geometry but at different resolutions, one made of 51M hexahedral elements and another of 204M hexahedrals. As we have already outlined in Sect. 2, in our industrial environment, our simulation engineers are blocked when performing visualisation tasks on their PCs for meshes containing 50 M cells. We chose a 51M elements mesh as it is representative of a typical study at EDF which will induce performance issues for the visualization process. Furthermore, it more than doubles the size used in the results presented in [8] for the PHASTA adaptor. On the other hand, when research oriented simulations are performed at EDF, these simulations currently contain around 200M elements. We choose this size as a typical research oriented or “heavy mesh” simulation data.
(2) Pipeline complexity. We define two pipelines aimed to be representative of two different situations: users performing simple and light visualization operations (e.g. generating slices in a volume) and another using very time-consuming visualization tasks (e.g. performing volume rendering).

In the following we name our uses cases: *CASE_A*, use case using an average mesh size of 51M hexahedrals and a visualization pipeline including volume rendering which aims to be very time-consuming. *CASE_B*, our second use case, contains a light visualization pipeline simply performing some slices but on a large mesh of 204M hexahedrals.

Table 2 summarizes the composition of these use cases. In all our use cases we run a simulation consisting in a fluid with physical properties identical to water passing through the mesh. Then the output is generated at each step, for a total of ten co-processed visualization images.

Table 2 Description of our two use cases

Use cases sum up			
Name	Size	Pipeline	Figures
CASE_A	51M hexahedrals, industrial size case	**Heavy**: volume rendering, celldatatopointdata and glyphs	4a, c, e
CASE_B	204M hexahedrals, research size case	**Light**: nine slices, celldatatopointdata	4b, d, f

6.3 Results

Figure 3b presents an image obtained from one of our in-situ simulations with *CASE_A*. We see the flux of water moving around the vertical cylinders, the glyphs being attached to the velocity vectorial field. The color of the volume rendering represents the turbulent viscosity of the fluid.

We establish first the overhead induced by storing our simulation results in Figs. 4a,b. We observe an average of 18 and 14 % of time used to store results, for *CASE_A* and *CASE_B* respectively. These figures correspond to the comparison of T_w and $T_s + T_w$ in Eq. 1. This overhead tends to increase with the number of processes in use. One can also notice that the overhead is also not stable and subject to important variations with a peak at 26 %. We thus identify the storage process as a bottleneck in everyday CFD studies for its average overhead and its high instability in execution time.

Figure 4c shows two graphs of *CASE_A*: in red the execution time versus the number of cores, in blue the execution time without the co-processing overload. These figures correspond to the comparison of T_s and $T_s + T_{process}$ in Eq. 2. We are satisfied by this overload that is comprised between 20 and 30 % of the total execution time, when we chose complicated task with a high $T_{process}$. Moreover, it looks like this overload is decreasing when the number of cores increases. Figure 4d shows the exact same behavior in the *CASE_B* experiment. Both graphs are difficult to distinguish as the time needed for co-processing is circumscribed between 6 and 10 s, the overload (the difference between T_s and $T_s + T_{process}$) is less than 1 % of the total execution time. We notice that having heavy $T_{process}$ is not very usual in our applications and we consider *CASE_A* as an example of worst case scenario.

We also decided to compare the Catalyst overhead with a non-VTK-based storage strategy that performs no visualization operations. Figures 4e,f, show the comparison of the global execution time with Catalyst co-processing versus the Ensight Gold storage format. This means comparing $T_s + T_w$ and $T_s + T_{process} + T_{w-in-situ}$. Figure 4e presents our implementation results with *CASE_A*. This compares positively for Catalyst as the overhead is approximately 10 % and decreases when the number of cores increases. We notice that for *CASE_A* the heavy $T_{process}$ is already taken into account in the in-situ curve but $T_r + T_v$ is still not performed for the traditional visualisation scheme. This means that this result is very positive and we should not forget that $T_r + T_v$ is very time consuming for this case (and saturates our scientific PCs at EDF R&D).

Figure 4f presents our results for *CASE_B*. Here we can see the potential of Catalyst when lighter and more relevant visualization tasks are processed. Indeed, there is no more overhead as we gain an average of 10 % of execution time while freeing ourselves from storage issues (we evaluate the execution time peak of 3,000 processes as a result of concurrent accesses on our supercomputer storage disks). To emphasize this, Table 3 shows how much data each solution generates, namely a basic storage in Ensight Gold format versus our co-processing implementation using Catalyst. These informations are those of our CASE_B when performing a

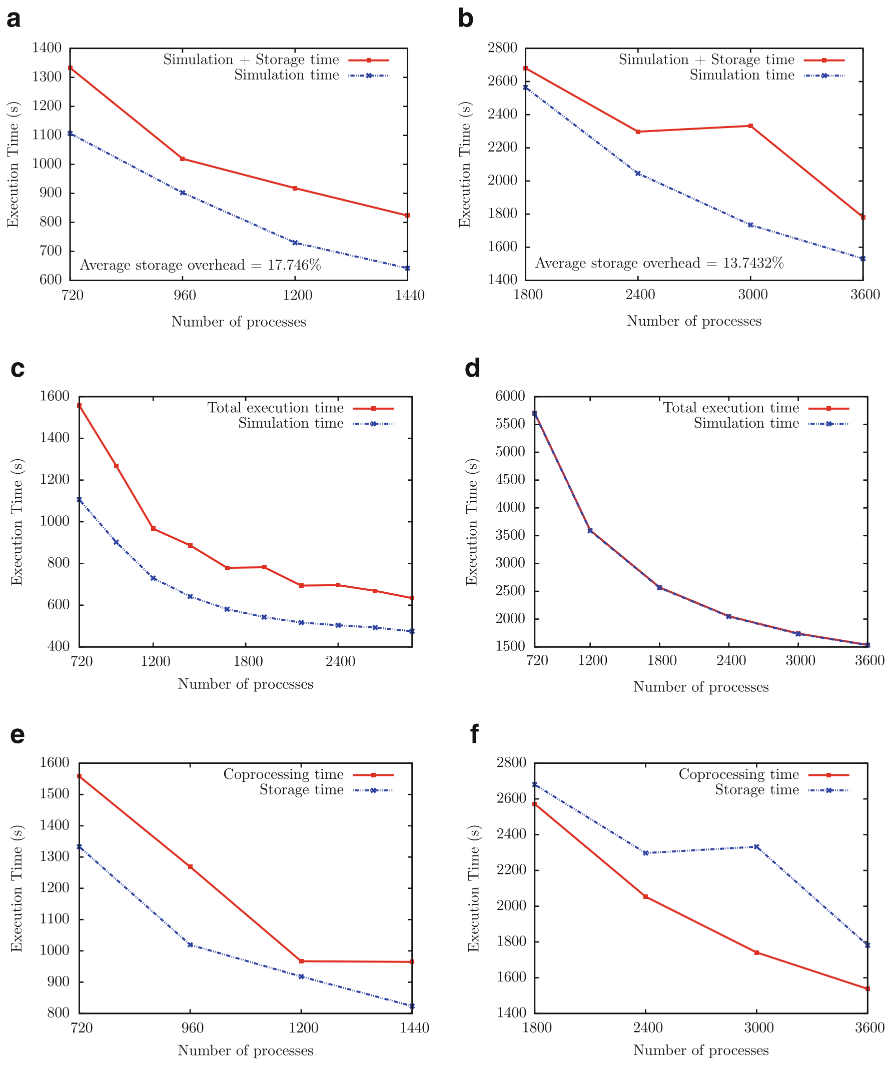


Fig. 4 CASE_A (*left*) and CASE_B (*right*) results. (**a**) Time overhead of storage, (**b**)time overhead of storage, (**c**) execution time with/out coprocessing, (**d**) execution time with/out coprocessing, (**e**) execution time comparison with storage, (**f**) execution time comparison with storage

ten steps simulation. Both size are expected to grow proportionally to the size of the mesh input, and the number of steps. Therefore, we expect the gain provided by the use of co-processing to be increasingly interesting when moving forward in use case size.

Table 3 CASE_B comparison between the size of processed results and simple storage

*Processing size comparison	
Storage	Coprocessing
57Gio	1,3Mio

The simulation was run on ten steps, with ten pictures co-processed

We also studied in [10] the total memory used when running in-situ visualization compared to writing simulation results in Ensight Gold format. We observe that memory consumption is increased by an approximate factor varying from 2 to 3. This can be explained by both our first naive memory management approach and also by a natural increase in memory consumption when visualization operations are to be performed. Concerning the memory management Kitware recently released the so-called "zero-copy VTK" that can be used to avoid naive deep copy of the simulation data structures.

Conclusion
This chapter provides an overview of the efforts needed to transition a traditional simulation code to an in-situ model. We believe that the results of our experiments are bringing new insights to the community, especially to the simulation teams (including simulation code developers and the visualization experts they work with) that are considering the transition to in-situ processing.

The main finding presented in this chapter is that we have successfully integrated Catalyst into *Code_Saturne* (a computational fluid dynamics code developed at EDF R&D). Both Catalyst and *Code_Saturne* are Open Source software and this development can be downloaded, used or tested freely by everyone. After testing the prototype in our corporate supercomputer Ivanhoe, we found Catalyst to be a relevant solution to provide *Code_Saturne* users with visualization co-processing. Catalyst also allowed for a simple and fast implementation of a simulation code adaptor.

The results presented are based on a 51M and a 204M elements mesh, which is above the average size case used by EDF engineers in our industrial environment. We plan to perform simulations on at least 400M elements meshes in the near future, using the same supercomputer. We performed our simulations from 60 up to 300 nodes. This is due to the typical simulation node size in Ivanhoe being around 150 nodes for our engineers. We also plan to work on another of our corporate supercomputers, an IBM BG/Q with 65k cores. In that case, we will test on a much larger number of cores.

The increase of memory usage, described in the results section of [10], indicates that memory optimizations are to be performed before running on

(continued)

the IBM BG/Q. We did not, in this study, perform any delicate memory tweaking in order to reduce the memory consumption. We are currently working on this point, experimenting with the new VTK in-situ data structures implemented recently by Kitware, the so-called "zero copy VTK". This approach aims to facilitate the memory management in the adaptor without the use of complicated pointer manipulation; we expect to reduce memory overhead without much increasing code complexity.

Another ongoing development consists on how we deal with the ghost levels generated by *Code_Saturne*. Indeed, we want to use the same spatial partition of the meshes for *Code_Saturne* and Catalyst, the aim being not to degrade the execution time by not necessary data exchanges among MPI ranks. We currently use ParaView D3 filter (a filter originally performing a redistribution of the data among MPI processes) as a ghost cell handler. However, we asked Kitware for the integration in ParaView/Catalyst of a new filter to perform a direct generation of ghost cells from existing distributed data. This development has been finished in December 2013.

This chapter has been dedicated on how to deal with large data using visual co-processing but we are also testing the computational-steering capabilities of Catalyst, the so-called Live Catalyst. This currently allows the modification of the ParaView pipeline parameters while the numerical simulation is running.

In conclusion, we are mostly satisfied with the integration of Catalyst in *Code_Saturne*. The first version of our integration will be released as part of a new version of this open-source software.

References

1. Archambeau, F., Mechitoua, N., Sakiz, M.: Code saturne: a finite volume code for the computation of turbulent incompressible flows. Industrial applications. Int. J. Finite Volumes **1**, 1–62 (2004)
2. Archambeau, F., Mechitoua, N., Sakiz, M.: Edf, code_saturne Version 3.0 Practical User's Guide (2013)
3. Biddiscombe, J., Soumagne, J., Oger, G., Guibert, D., Piccinali, J.-G.: Parallel computational steering for hpc applications using hdf5 files in distributed shared memory. IEEE Trans. Vis. Comput. Graph. **18**(6), 852–864 (2012)
4. Brooke, J.M., Coveney, P.V., Harting, J., Pickles, S.M., Pinning, A.R., Porter, R.L.: Computational steering in realitygrid. In: Proceedings of the UK E-Science All Hands Meeting, pp. 885–888 (2003)
5. Childs, H., Brugger, E., Bonnell, K., Meredith, J., Miller, M., Whitlock, B., Max, N.: A contract based system for large data visualization. In: IEEE Visualization, (VIS 05), pp. 191–198 (2005)

6. Ellsworth, D., Green, B., Henze, C., Moran, P., Sandstrom, T.: Concurrent visualization in a production supercomputing environment. IEEE Trans. Vis. Comput. Graph. **12**(5), 997–1004 (2006)
7. Esnard, A., Richart, N., Coulaud, O.: A steering environment for online parallel visualization of legacy parallel simulations. In: Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications, (DS-RT'06), pp. 7–14 (2006)
8. Fabian, N., Moreland, K., Thompson, D., Bauer, A.C., Marion, P., Geveci, B., Rasquin, M., Jansen, K.E.: The paraview coprocessing library: a scalable, general purpose in situ visualization library. In: IEEE Symposium on Large Data Analysis and Visualization (LDAV), pp. 89–96 (2011)
9. Law, C.C., Henderson, A., Ahrens, J.: An application architecture for large data visualization: a case study. In: Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics, pp. 125–159 (2001)
10. Lorendeau, B., Fournier, Y., Ribes, A.: In-situ visualization in fluid mechanics using catalyst: a case study for code saturne. In: IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV), pp. 53–57, (2013)
11. Moreland, K.D., Thompson, D., Fabian, N.D., Ice, L.G.: Design issues for performing in situ analysis of simulation data. Technical Report SAND2009-2014, Sandia National Laboratories, pp. 7–18 (2009)
12. Ribes, A., Caremoli, C.: Salome platform component model for numerical simulation. In: 31st Annual International Computer Software and Applications Conference, (COMPSAC 2007), vol. 2, pp. 553–564 (2007)
13. Ross, R.B., Peterka, T., Shen, H.-W., Hong, Y., Ma, K.-L., Yu, H., Moreland, K.: Visualization and parallel i/o at extreme scale. J. Phys. Conf. Ser. **125**(1), 012099 (2008)
14. Tu, T., Yu, H., Ramirez-Guzman, L., Bielak, J., Ghattas, O., Ma, K.-L., O'Hallaron, D.R.: From mesh generation to scientific visualization: an end-to-end approach to parallel supercomputing. In: Proceedings of the ACM/IEEE SC 2006 Conference, pp. 12–12 (2006)
15. Vezolle, P., Heyman, J., D'Amora, B., Braudaway, G., Magerlein, K., Magerlein, J., Fournier, Y.: Accelerating computational fluid dynamics on the ibm blue gene/p supercomputer. In: 22nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 159–166 (2010)
16. Brad Whitlock, J.-M.F., Meredith, J.S.: Parallel in situ coupling of simulation with a fully featured visualization system. In: Proc. of 11th Eurographics Symposium on Parallel Graphics and Visualization (EGPGV'11), pp. 101–109 (2011)

Sublinear Algorithms for Extreme-Scale Data Analysis

C. Seshadhri, Ali Pinar, David Thompson, and Janine C. Bennett

1 Introduction

The need to ingest and glean insight from massive amounts of data is a ubiquitous problem facing the scientific community and the world at large. Increases in the quality and availability of high resolution sensing equipment and computing resources enable ever more enhanced experiments and simulations, but I/O constraints are impeding their impact. The increases in spatial and temporal resolutions are so large that not all data of interest can be written to disk. This makes it impossible to track features with timescales smaller than that of I/O frequency. We need a fundamental shift away from a post-process centric data analyses. Concurrent analysis frameworks are a promising direction wherein raw simulation and/or experimental data is processed as it is computed, decoupling the analysis from I/O. In high performance computing, in-situ [16, 18, 49] and in-transit [1, 5, 46] processing are based on performing analyses as the simulation is running, storing only the results, which are several orders of magnitude smaller than the raw data. This mitigates the effects of limited disk bandwidth and capacity. Operations sharing primary resources of the simulation are considered in-situ, while in-transit processing involves asynchronous data transfers to secondary resources. These work flows pose unique challenges as algorithms must be re-designed to operate within tight memory, communication, and I/O constraints.

C. Seshadhri • A. Pinar • J.C. Bennett (✉)
Sandia National Laboratories, PO Box 969, Livermore, CA 94550, USA
e-mail: scomand@sandia.gov; apinar@sandia.gov; jcbenne@sandia.gov

D. Thompson
Kitware, Inc., 101 East Weaver St, Carrboro, NC 27510, USA
e-mail: david.thompson@kitware.com

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_3

Another realm of importance is *analysis of massive graphs*. In fields as diverse as scientific computing, cybersecurity, and computational social sciences, graphs are used to represent and manipulate data. The sizes of these graphs are becoming extremely large, and our algorithmic abilities are relatively limited. Various graph processing tasks, like subgraph mining, scale superlinearly in the size of the graph, which rapidly becomes infeasible for the practical data sizes in many applications. Architecture advances alone cannot solve this problem, and new algorithms tailored such extreme size problems are required. We believe that *sublinear algorithms* may provide a solution to the analysis challenges for extremely large data. In the rest of this chapter, we will provide a brief introduction to the area of sublinear algorithms, and then discuss adoption of sublinear algorithms in two applications in graph mining and scientific visualization.

2 Sublinear Algorithms

Sublinear algorithms is a recent development in theoretical computer science and discrete mathematics, which addresses the mathematical problem of understanding global features of a data set using limited resources. Often enough, to determine important features of an input, one does not need to actually look at the entire input. The field of sublinear algorithms [19, 29, 30] makes precise the circumstances when this is possible and combines discrete math and algorithmic techniques with a rich set of statistical tools to quantify error and give trade-offs with sample sizes. This confidence measure is necessary for adoption of such techniques by the large-scale scientific computing community, whose scientific results are often used to make high-impact decisions that could have large financial or public policy implications.

Samples chosen by sublinear algorithms are more intricate than statistical random sampling based on standard closed-form distributions. A key insight of sublinear algorithms is that understanding different properties of a data set require different types of samples. It may not even be possible to explicitly describe the samples used by the sublinear algorithm. For example, to estimate the degree information of a graph, we may sample uniform edges [22]; to find low-conductance cuts, we may sample random walks [21]; to count triangles, we may sample paths of short length [35]. The art and creativity in sublinear algorithms is in constructing the algorithm that samples the "right structure".

Although at a high level, any question that can be framed in terms of determining global properties of a large domain is subject to a sublinear analysis, surprisingly, the origins of this field has had nothing to do with "big data" or computational challenges in data analysis. This birth of sublinear algorithms is in computational complexity theory [31], an area quite disjoint from fields like scientific computing or data mining. Hence, there is almost no work in applying these methods to real applications.

2.1 The Formal Setting

Suppose we have a function $f : D \to R$. We are given *oracle access* to this function, meaning, for any $x \in D$, we can query $f(x)$. To make this more concrete, consider the following examples. If $D = [n]^3$ and $R = \mathbb{R}$, this could be the temperature values on a structured grid coming from a jet engine simulation. If $D = [n]$ and $R = \{0, 1\}$, then f represents an n-bit binary string. If $R = \{A, T, G, C\}$, then f could represent a piece of DNA. If $D = [n]^2$, then the function could represent a matrix (or a graph). If D is an arbitrary graph, then this could be data in an unstructured grid. In this way, almost any data analysis input can be cast as a collection of functions over a discrete domain.

We are interested in some *specific* aspect of f. For example, given two strings (which we can think of as a function), we may want to know how common they are. We may wish to know the sizes of regions of high temperature in a jet engine simulation. In a graph, we wish to know how many cliques of size 4 it contains. In the theory of sublinear algorithms, these are abstracted as *properties*, which one can think of as yes/no questions. Let us go back to the examples. Is the edit distance between two strings at least 30 %? Does there exist a high-temperature region spanning the x-axis of the grid? Are there at least three million cliques in the graph?

How do we determine whether f satisfies property $\mathscr{P}$ *without querying all of* f? Naturally, it is not possible to give an exact answer without knowledge of f. To formalize what can be said by querying $o(|D|)$ values of f, we introduce a notion of the *distance to* $\mathscr{P}$. Every function f has a distance to $\mathscr{P}$, denoted ε_f, where $\varepsilon_f = 0$ iff f satisfied $\mathscr{P}$. To provide an exact answer to questions regarding $\mathscr{P}$, we determine whether $\varepsilon_f = 0$ or $\varepsilon_f \neq 0$. However, approximate answers can be given by choosing some error parameter $\varepsilon > 0$ and asking: can we distinguish $\varepsilon_f = 0$ from $\varepsilon_f > \varepsilon$? The theory of sublinear algorithms shows that often, the latter question can be resolved by an algorithm that queries $o(|D|)$ function values.

For example, it may be possible to distinguish in sublinear time if the edit distance is at most 25 % or at least 30 %. It may be possible to distinguish if there does not exist a high temperature region spanning 90 % of the x-axis, or there is one spanning 95 % of the x-axis. The exact nature of the approximation depends heavily on the property at hand, but the theory provides a wide range of mathematical tools to quantify the approximation.

For a sublinear algorithm, there are usually three parameters of interest: the number of queries t, the error ε, and the confidence δ. As described earlier, the error is expressed in terms of the distance to $\mathscr{P}$. The mathematical analysis will show that for a given t, we can estimate the desired answer within error ε with a confidence of $> 1 - \delta$. Conversely, given ε, δ, we can compute the number of queries required. Of course, we are probably not just interested in yes/no questions, but quantifiable properties. This yes/no setting is a mathematical convenience, and the sublinear algorithms can usually provide approximate answers for these problems. But the theory is set up as such questions, and this makes it easier to prove theorems and express guarantees formally.

The actual sublinear algorithm is often quite complex and uses many levels of randomization. It is not as simple as just taking a uniform random sample of the function. For example, in the case of the temperature function, such a sample would give no information about spatial locality, and hence cannot detect large connected regions of high temperature. The samples used by the sublinear algorithm are often correlated in complex ways, and it is not possible to write out the distribution of queries in closed-form. Nonetheless, these algorithms *can* be mathematically analyzed, using the techniques of randomized analysis from theoretical computer science. This involves discrete probability theory, combinatorics, discrete math, and graph theory. There is heavy use of concentration inequalities from probability theory and functional analysis, which gives tail bounds for functions of random variables.

The remaining sections summarize two recent applications that showcase the immense potential of transferring the advances in theoretical computer science into a practical realm. First, we present how sampling algorithms are used for triadic analysis in large-scale graphs. Then we discuss how to choose color maps to make prominent features more distinguishable during visualization of large-scale data.

3 Sublinear Techniques for Triangle Counting

Graphs are used to model infrastructure networks, the World Wide Web, computer traffic, molecular interactions, ecological systems, epidemics, citations, and social interactions, among others. One of the methods for classifying and understanding these graphs is *subgraph mining*. Given a graph G, any subset of its edges (with the corresponding vertices) is called a subgraph. In G, if vertices u, v, and w are all connected to each other, then this triangle is a subgraph of G. It has long been observed in physics, biology, and social sciences, that the frequency of small subgraphs plays a key role in the structure of the data [3, 10, 13, 15, 20, 24, 28, 37, 39, 40, 47, 48]. Computing these measures is an important aspect of understanding and processing these graphs.

In particular, the *triangle* has received special attention in a majority of this work. The triangle is a set of three vertices that are all connected to each other. The history of studying triangles is extremely long and rich, with the first publications on this subject back in the 1970s by social scientists [24]. Diverse fields such as physics, sociology, biology, cybersecurity, computer science have focused on triangle counting [3,10,13,15,17,20,24,48]. From a scientific computing perspective, it is a common problem to run on high performance computing systems. There is much research on getting scalable algorithms for this problem; some focus on enumeration [6, 9, 12, 32, 38] and others on fast counting schemes [2, 26, 33, 42–45].

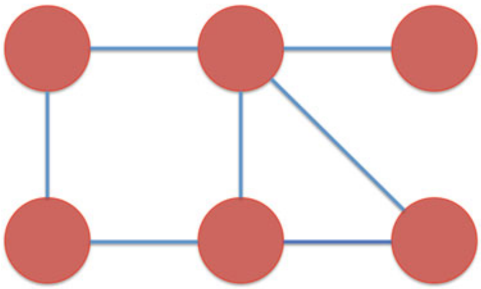


Fig. 1 Example graph with 12 wedges and 1 triangle

Even a moderately sized graph with millions of edges can have many orders of magnitude more triangles [35]. Hence, an enumeration is bound to be expensive due to the size of the output, regardless of how well the algorithm or architecture is engineered. Sublinear subgraph counting is all about using powerful graph sampling methods to answer such questions without enumeration. But naïve sampling schemes such as looking at uniform random vertices or uniform random edges are doomed to fail, since subgraphs can often be concentrated in small regions of the graph (consider a graph having a few dense regions containing all the triangles, but most vertices and edges participate in few of these) (Fig. 1).

3.1 The Nuts and Bolts of Wedge Sampling

We summarize the techniques and results of a successful sampling procedure that works in practice for triangle counting [35], which was based on the concept of *wedge sampling* [33]. A wedge is a path of length 2. Suppose the total number of triangles in G is T, and the total number of wedges is W. A common triangle measure is the *transitivity* τ, the fraction of wedges that participate in triangles. This is exactly $3T/W$ (there is a factor 3 because a triangle contains 3 wedges.)

Let us begin by trying to estimate the transitivity. Denote a wedge to be *closed*, if it participates in a triangle, and *open* otherwise. Let $\mathscr{W}$ be the uniform distribution on wedges. Observe that $\tau = \Pr_{w \sim \mathscr{W}}[w \text{ is closed}]$. Consider the following thought experiment: sample t uniform random wedges $w_1, w_2, \ldots, w_t$. Output the fraction of these wedges that are closed as $\hat{\tau}$, the transitivity estimate. How far is this from the true answer? Define an indicator random variable X_i for the wedge w_i being closed. The X_i's are i.i.d. and $\mathbf{E}[X_i] = \tau$. By the Hoeffding bound [23], for any $\varepsilon \in (0, 1)$,

$$\Pr\left[\left|\sum_i X_i/t - \tau\right| > \varepsilon\right] < 2\exp\left(-\varepsilon^2 t/2\right).$$

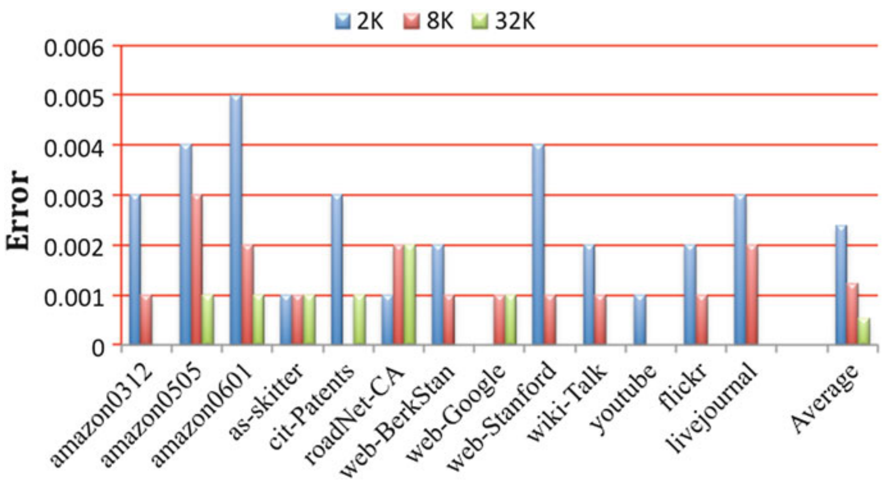


Fig. 2 Absolute errors in transitivity for 2K, 8K, and 32K samples (from [35])

Reverse engineering this bound, set $t = \lceil (2/\varepsilon^2)\ln(2/\delta)\rceil$, for any choice of accuracy ε and confidence δ. This procedure is guaranteed to estimate the transitivity up to additive error ε with probability $1-\delta$. We can explicitly provide a confidence guarantee and quantify the number of samples required. From this, we can also get an estimate on the number of triangles.

At this point, this is still a thought experiment, since we do not know how to sample a uniform random wedge. This can be done with a simple preprocessing on the graph. Let us define the degree of a vertex as the number of its neighbors. It is standard to compute the degrees of all vertices of G. Suppose we sampled vertex v with probability proportional to $\binom{d_v}{2}$ (where d_v is the degree of v). If we choose a uniform random pair of neighbors of v, this can be shown to yield a uniform random wedge. The data structures and sampling can be set up cheaply without much difficulty, and it gives one of the fastest algorithms to estimate the transitivity (and triangle counts). Refer to Fig. 2, which shows the transitivity error on real graphs with the number of samples set to 2K, 8K, and 32K. Observe how the error for 32K samples is less than 0.002 for all these graphs.

Sublinear sampling is incredibly versatile. We can compute different triangle statistics by modifying the sampling, that is by sampling wedges according to other distributions. For instance, by choosing uniform random vertices as centers of wedges, we can compute the local clustering coefficient [35]. Or we can bin vertices and compute the clustering coefficients for wedges centered at vertices in that bin. For instance, Fig. 3 shows the results of an experiment, where the vertices were binned by their degrees to enable computation of degree-wise clustering coefficients, giving a nuanced view of triangle correlations with degrees.

Wedge sampling can also be extended for triangles in directed graphs. For directed graphs, there will be multiple wedge and triangle types, as opposed to a

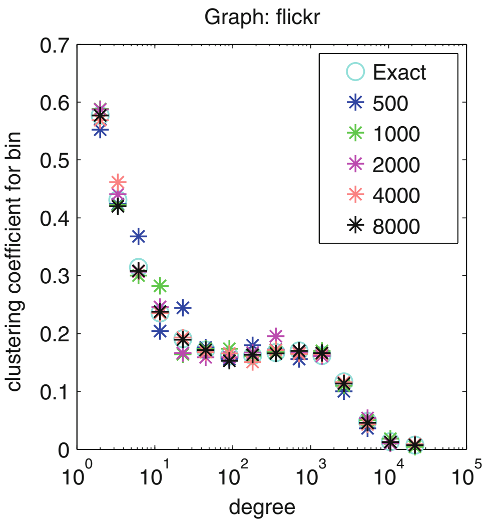


Fig. 3 Getting degree-wise clustering coefficients for the flickr graph, which has ten million edges (from [35])

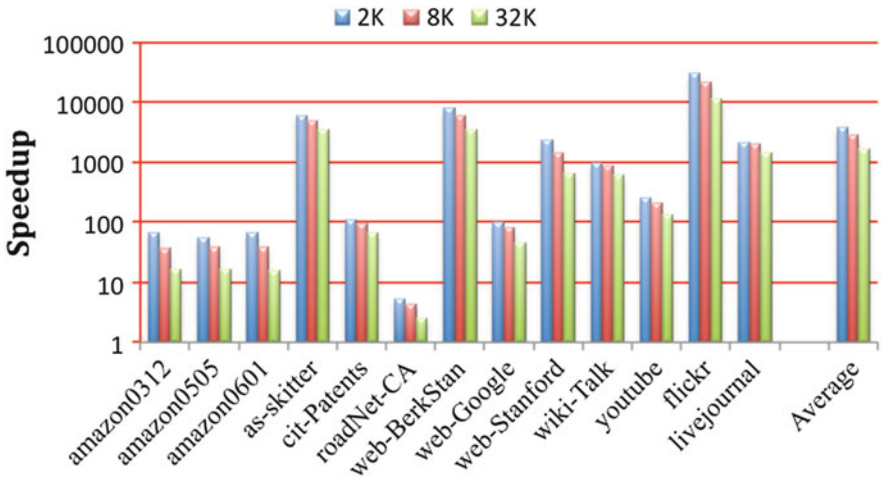


Fig. 4 Improvements in runtime for a wide variety of graphs

single wedge and a single triangle type for undirected graphs [34]. However, by properly defining the distribution from which to choose wedges, we can compute estimates for the numbers of triangles of all types [36].

All of these algorithms beat clever enumerations algorithms by orders of magnitude in running time. In Fig. 4, we show the speedup of our sampling algorithm over a state of the art enumeration algorithm for different sampling rates. For the largest sample size used, the runtime improvement over enumeration is close to a factor of 100 (and improvements increase for larger graphs).

3.2 Wedge Sampling for Massive Graphs

The methods presented in [35] go beyond traditional computing models where the graph is in memory. The wedge sampling algorithms have been successfully parallelized in the Map-Reduce model [25]. Observe that after the graph is preprocessed to set up wedge sampling, each sample can be generated and checked for closure in parallel. The preprocessing step basically requires the degree distribution (and a few extra data structures, all computable from the degrees), which can be obtained in two Map-Reduce passes. Direct implementations require significant communication (called "shuffling") in the Map-Reduce framework. Randomized sampling methods can also be used to significantly reduce this communication cost. The final algorithm generated triangle numbers for the `uk-union` graph [7, 8] (which has four billion edges) in about 30 min. These are the largest published numbers on triangle counting, and demonstrate the scalability of sublinear techniques like wedge sampling.

4 Color Maps

Our aim is to apply these sampling-based techniques to large-scale, physics-based simulations, although the data analysis challenges are significantly different. Our initial investigations in application-independent generation of color maps show promise [41]. The visualization of large-scale data can be difficult because determining a good color map requires information about the distribution of values in the data. The size of the data often makes this process computationally expensive. Using sampling methods, we have devised an algorithm to efficiently generate color maps that support the visualization of features of interest on large-scale data. The color maps relate the prominence of values sampled to the distinguishability of colors in the map. Besides the perceptual issues involved, analysis of the samples is driven by ignorance of whether the data being presented is discrete or continuous in nature. Thus, the sampling process is also driven by a desire to separate prominent discrete values from samples that appear to approximate continuous distributions. The difficulty is that computational representations of sample values are always discretized in some sense and that discretization may occur at the same scales as the physical experiment or numerical simulation generating the dataset.

The approximation we make is that the data may be modeled as a mixture of two distributions: one continuous and one discrete. Values repeated a significant fraction τ of the dataset (such as 0.1 % of all data) are considered to be *prominent values* taken from the discrete distribution; while other values are drawn from a distribution with a continuous cumulative distribution function (CDF), which we wish to estimate. We use the sampling theory to show that the number of samples, t, required to detect prominent values is dependent only on τ and the confidence $(1-\delta)$—the probability that a given sample of the data will contain each prominent value. Thus the number of samples is independent of the dataset size. We also show that the number of samples required to estimate the CDF to within a small, fixed fraction with a high confidence is also independent of dataset size. In the next section, we present the bounds on sample size for both prominent value detection and CDF estimation. After that we illustrate how the samples are used to create color maps on large data.

4.1 Bounding Sample Sizes for Color Maps

The analysis of both algorithms follow from straightforward applications of Chernoff bounds, although we will only discuss the detection of discrete prominent values below. We state the *multiplicative Chernoff bound* (refer to Theorem 1.1 in [14]) for sums of independent random variables.

Theorem 1 (Chernoff Bound) *Let* $X_1, X_2, \ldots, X_k$ *be independent random variables in* $[0, 1]$ *and* $X = \sum_{i=1}^{k} X_i$.

- *(Lower tail) For any* $\varepsilon > 0$,

$$\Pr[X < (1-\varepsilon)\mathbf{E}[X]] \leq \exp(-\varepsilon^2 \mathbf{E}[X]/2).$$

- *(Upper tail) For any* $\varepsilon > 0$,

$$\Pr[X > (1+\varepsilon)\mathbf{E}[X]] \leq \exp(-\varepsilon^2 \mathbf{E}[X]/3).$$

- *(Upper tail) For any* $t > 2e\mathbf{E}[X]$,

$$\Pr[X > t] \leq 2^{-t}.$$

For the first problem, of detecting prominent values, we treat our data as a discrete distribution. For each value r, p_r is the fraction of the dataset where the value r is attained (so $\{p_r\}$ describes a distribution over the range of the dataset). We use $\mathscr{D}$ to denote this distribution and R to denote the support. For any set S (often an interval of the real line), we use $P(S)$ to denote the probability mass of S.

In our theorems, we do not attempt to optimize constants; for convenience, we use c and c' to denote sufficiently large constants. In practice, we have found that $c = 5$ is sufficient for the combustion simulation and other synthetic data we have encountered.

Our aim is to determine values of r such that $p_r > \tau$, where $\tau \in (0,1)$ is a *threshold parameter*. We expect the algorithm below to identify r with probability $1-\delta$.

```
find-prominent($t, \tau$)
Inputs: sample size $t$, threshold $\tau$
Output: the "prominent set" of range elements, $I$.

1. Generate set $S$ of $t$ independent random samples from $\mathscr{D}$
2. Initialize important set $I = \emptyset$.
3. For any element $r \in \mathscr{D}$ that occurs more than $t\tau/2$ times in $S$,
       Add $r$ to $I$.
4. Output $I$.
```

The following theorem states that (up to some approximation), I is indeed the set of frequent elements. The constants in the following are mainly chosen for presentation. (Instead of $p_r < \tau/8$ in the following, we can set it to τ/α, for any $\alpha > 1$, and choose t accordingly.) Throughout our theorems, we use δ for a tunable error parameter that decides the sample size t. Note that it is not an explicit parameter to the algorithms.

Theorem 2 *Set* $s = (c/\tau)\ln(c/(\tau\delta))$. *With probability* $> 1-\delta$ *(over the samples), the output of* `find-prominent`(τ, δ) *satisfies the following.*

If $p_r > \tau$, *then* $r \in I$.
If $p_r < \tau/8$, *then* $r \notin I$.

We show this in [41] by first proving that with probability at least $1-\delta/2$, for all $p_r > \tau, r \in I$. Then we show that with probability at least $1-\delta/2$, for all $p_r < \tau/8$, $r \notin I$. A union bound then completes the proof using the lemma below.

Lemma 1 *Let* $\alpha \in (0,1)$ *and* $s > (c/8\alpha)\ln(c/(8\alpha\delta))$. *Consider a set* R' *such that* $\forall r \in R'$, $0 < p_r \le \alpha$. *With probability* $> 1-\delta/2$, *the following holds. For all* $r \in R'$, *the number of occurrences of* r *in* s *uniform random samples from* $\mathscr{D}$ *is at most* $4s\alpha$.

Again, the constant c is unknown because the Chernoff bounds make no demands of the distribution of $\mathscr{D}$; as the variance of $\mathscr{D}$ diverges, so does c. By fixing c, the performance of the algorithm will degrade when the variance is high.

Similar arguments allow us to bound the number of samples required to estimate the CDF to within a given error with high confidence. Our aim is to construct a series of disjoint intervals (or, blocks) that (almost) equally partition the probability mass. To gain some intuition, consider a positive integer v and a sequence of numbers $y_0, y_1, \ldots, y_v$ where $y_0 = \min_{r\in R} r$, $y_v = \max_{r\in R} r$, and for all $i < v$, $P([y_i, y_{i+1})) = 1/v$. Our algorithm will try to find these intervals (for a parameter v). Of course, such intervals may not even exist, due to the discrete nature of $\mathscr{D}$.

Nonetheless, we will try to find suitable approximations. We assume that there are no values in $\mathscr{D}$ with high probability. This is an acceptable assumption, since we run this procedure after "removing" prominent values from $\mathscr{D}$.

There are two parameters for CDF approximation: the sample size t and a block size b that is used to group samples into intervals. For convenience assume b divides t.

```
find-CDF($t, b$)
Inputs: sample size $t$, block size $b$
Outputs: Intervals $B_1, B_2, \dots$

1. Generate set $S$ of $t$ independent random samples from $\mathscr{D}$.
2. Sort these to get the (ordered) list $\{x_1, x_2, x_3, \dots, x_s\}$.
3. Output the intervals $B_1 = [x_1, x_b)$, $B_2 = [x_{b+1}, x_{2b})$, etc. In general, the
   $i$th interval $B_i$ is $[x_{(i-1)b+1}, x_{ib})$ and there are $t/b$ blocks. The samples in
   this interval form the associated set, so $s(B_i) = |B_i \cap S|$.
```

This main theorem involves some play of parameters, and we express t and b in terms of an auxiliary integer parameter v. Again, the constants chosen here are mainly given for some concreteness and notational convenience. We use the notation $A \in (1 \pm \beta)B$ as a shorthand for $A \in [(1-\beta)B, (1+\beta)B]$. A proof of this theorem can be found in [41].

Theorem 3 *Set $t = cv \ln(cv/\delta)$ and $b = t/v$. Suppose there exists no $r \in R$ such that $p_r > 1/100v$. With probability $> 1 - \delta$, the following holds. For each output interval B, $P(B) \in (1 \pm 1/10)/v$. Furthermore, $P((\min_{r \in R} r, x_1))$ and $P((x_s, \max_{r \in R} r))$ are at most $1/50v$.*

Thus, by evaluating `find-prominent` and `find-CDF` on a small sample t, we obtain a statistical model of the data composed of a discrete distribution I mixed with a continuous distribution modeled by ranges B_j. All that remains is providing a method for choosing colors as the data is rendered.

4.2 Creating Color Maps from Samples

Because our model of the data includes both discrete and continuous components, we can provide a clear distinction between high-probability values and those approximating a continuum. We assign a perceptually distinct color to each entry of I computed by `find-prominent`. If there are more values in I than the human visual can distinguish, multiple values are assigned the same color in a round-robin fashion. For the blocks B_j that are (almost) equiprobable, we assign colors from an *ab*-line in *Lab* color-space such that points on the line assigned to each block are perceptually equidistant. This provides an illustration of the data that is good at showing large-scale, continuous trends.

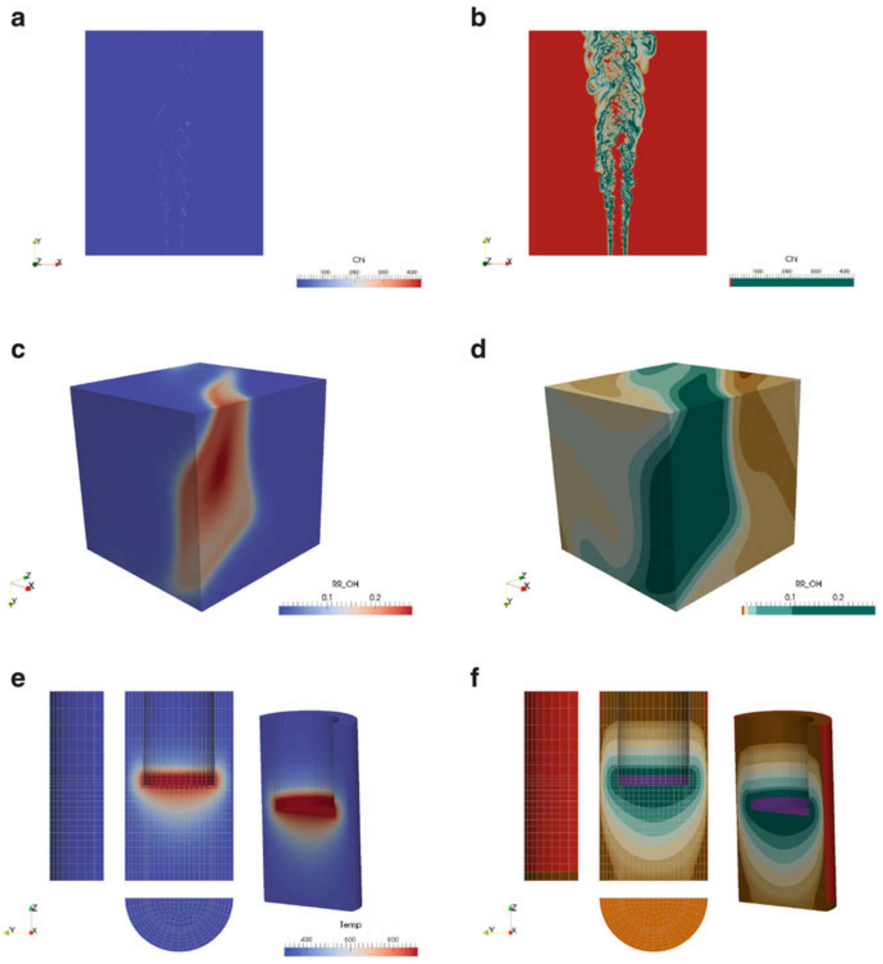


Fig. 5 (**a**) and (**c**) demonstrate default colormap generated by ParaView for two separate combustion data sets and (**e**) demonstrates the default colormap generated by ParaView for temperature inside a rotating disk reactor. (**b**), (**d**), and (**f**) demonstrate sampling-based colormaps for the same datasets. Sampling is able to identify trends and features in the data as well as constant boundary conditions that are otherwise obscured by a traditional linear color map constructed from the full scalar value range

Figure 5 contrasts the sampling-based colormap that our algorithm produces with the default view produced by Paraview for two combustion datasets and for temperature on a rotating disk reactor. As seen in the rotating disk example in Fig. 5e, f, our technique is able to identify constant boundary conditions of temperature that are otherwise obscured by a traditional linear color map. The combustion data as seen in Fig. 5a–d was generated by S3D [11], a turbulent combustion simulation code that performs first principles-based direct numerical simulations in which both

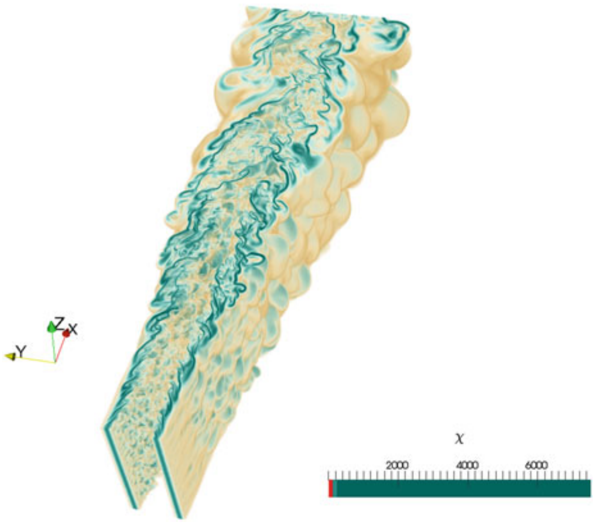


Fig. 6 Sampling-based colormaps could be used to generate high-quality visualizations in a data-driven fashion in-situ, as the simulation is running

turbulence and chemical kinetics introduce spatial and temporal scales spanning typically at least five decades. In the combustion data our approach visually depicts structures in the data that are difficult to see in the default view. These structures cover a relatively small percentage of the overall scalar domain, however these values are observed with relatively high frequency within the data. We note that our approach can be effective at identifying coherent structures in spite of the fact that we are not integrating spatial information into our estimates. Thus, sampling-based colormaps could be leveraged to generate high-quality visualizations in a data-driven fashion in-situ, as a large-scale simulation is running, see Fig. 6.

5 Future Opportunities

There is a rich set of opportunities for which sublinear theory could be leveraged to develop scalable analysis solutions for massive data. Consider, for example, the aggregation of feature-based statistics [4]. Given functions defined over the three-dimensional grid, combustion scientists are interested in the average size of the connected regions with a value greater than some threshold [27]. A sublinear algorithm would ascertain this value without examining the whole domain by choosing a small subset (using randomization) through complex algorithmic schemes. The output would be approximate, but with a quantified time-error trade-off. In scientific applications with an underlying turbulent transport, it is often important to track structures of interest over time as they are advected by the flow. The full analysis can be too computationally expensive at the frequency required by the simulation timescales, and the sketches provided by sublinear algorithms might be utilized as a surrogate for a subset of the time steps. Numerous problems of interest in scientific computing can be cast as finding patterns (such as subsequences or specific substrings) in a large file. For example, the right cache size for compiled code is often estimated through access patterns of registers, which can be abstracted as pattern matching in long strings. Many problems in bioinformatics and cybersecurity involve finding large common subsequences between sets of strings. Sampling methods from sublinear algorithms could be used for fast estimation of these quantities. Finally, many algorithms running on massive inputs work much faster if prior knowledge of the input is incorporated. This is often encoded in terms of algorithm parameters, which are set by domain experts based on their estimates of global properties of the input. Sublinear algorithms can explore a tiny portion of the domain to provide accurate, data-driven estimates of these desired global properties.

In summary, sublinear algorithms demonstrate enormous potential in providing solutions to data analysis challenges in extreme-scale scientific computing environments. This revolutionary approach tells us how to perform computations by looking at small portions of the data, potentially leading to huge reductions in required communication. Initial results in transitioning these technologies from theory to practice show promise for current architectures. However, further study is required to understand both their broad applicability and performance on potential future architectures.

Acknowledgements The authors wish to thank Tamara G. Kolda, Madhav Jha, and Todd Plantenga for many helpful discussions and Dr. Jacqueline Chen for access to the combustion use case examples used in this work. This work is supported by the Laboratory Directed Research and Development (LDRD) program of Sandia National Laboratories and the DARPA GRAPHS program. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. Abbasi, H., Eisenhauer, G., Wolf, M., Schwan, K., Klasky, S.: Just in time: adding value to the IO pipelines of high performance applications with JIT staging. In: Proc. of 20th International Symposium on High Performance Distributed Computing (HPDC'11) (2011)
2. Arifuzzaman, S.M., Khan, M., Marathe, M.: PATRIC: A parallel algorithm for counting triangles and computing clustering coefficients in massive networks. NDSSL Technical Report 12-042, Network Dynamics and Simulation Science Laboratory, Virginia Polytechnic Institute and State University (2012)
3. Becchetti, L., Boldi, P., Castillo, C., Gionis, A.: Efficient semi-streaming algorithms for local triangle counting in massive graphs. In: KDD'08, pp. 16–24 (2008)
4. Bennett, J., Krishnamoorthy, V., Liu, S., Grout, R., Hawkes, E.R., Chen, J.H., Shepherd, J., Pascucci, V., Bremer, P.-T.: Feature-based statistical analysis of combustion simulation data. IEEE Trans. Vis. Comput. Graph. **17**(12), 1822–1831 (2011)
5. Bennett, J.C., Abbasi, H., Bremer, P.-T., Grout, R., Gyulassy, A., Jin, T., Klasky, S., Kolla, H., Parashar, M., Pascucci, V., Pebay, P., Thompson, D., Yu, H., Zhang, F., Chen, J.: Combining

in-situ and in-transit processing to enable extreme-scale scientific analysis. In: Hollingsworth, J. (ed.) SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake Convention Center, Salt Lake City, 10–16 November 2012, pp. 49:1–49:9, pub-IEEE:adr. IEEE Computer Society Press (2012)

6. Berry, J., Fostvedt, L., Nordman, D., Phillips, C., Seshadhri, C., Wilson, A.: Why do simple algorithms for triangle enumeration work in the real world? In: Innovations in Theoretical Computer Science (ITCS) (2014)
7. Boldi, P., Vigna, S.: The web graph framework I: compression techniques. In: Proc. of the Thirteenth International World Wide Web Conference (WWW 2004), Manhattan, , pp. 595–601. ACM Press, New York (2004)
8. Boldi, P., Rosa, M., Santini, M., Vigna, S.: Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In: Proceedings of the 20th International Conference on World Wide Web. ACM Press, Madrid (2011)
9. Buriol, L., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., Sohler, C.: Counting triangles in data streams. In: PODS'06, pp. 253–262 (2006)
10. Burt, R.S.: Structural holes and good ideas. Am. J. Sociol. **110**(2), 349–399 (2004)
11. Chen, J.H., Choudhary, A., de Supinski, B., DeVries, M., Hawkes, E.R., Klasky, S., Liao, W.K., Ma, K.L., Mellor-Crummey, J., Podhorski, N., Sankaran, R., Shende, S., Yoo, C.S.: Terascale direct numerical simulations of turbulent combustion using s3d. Comput. Sci. Discov. **2**, 1–31 (2009)
12. Cohen, J.: Graph twiddling in a MapReduce world. Comput. Sci. Eng. **11**, 29–41 (2009)
13. Coleman, J.S.: Social capital in the creation of human capital. Am. J. Sociol. **94**, S95–S120 (1988)
14. Dubhashi, D., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press, New York (2009)
15. Eckmann, J.-P., Moses, E.: Curvature of co-links uncovers hidden thematic layers in the World Wide Web. PNAS **99**(9), 5825–5829 (2002)
16. Fabian, N., Moreland, K., Thompson, D., Bauer, A.C., Marion, P., Gevecik, B., Rasquin, M., Jansen, K.E.: The paraview coprocessing library: a scalable, general purpose in situ visualization library. In: Proc. of IEEE Symposium on Large Data Analysis and Visualization (LDAV), pp. 89–96 (2011)
17. Fagiolo, G.: Clustering in complex directed networks. Phys. Rev. E **76**, 026107 (2007)
18. Favre, J.M., Whitlock, B., Meredith, J.S.: Parallel in situ coupling of simulation with a fully featured visualization system. In: Proc. of 11th Eurographics Symposium on Parallel Graphics and Visualization (EGPGV'11) (2011)
19. Fischer, E.: The art of uninformed decisions: a primer to property testing. Bull. EATCS **75**, 97–126 (2001)
20. Foucault Welles, B., Van Devender, A., Contractor, N.: Is a friend a friend?: Investigating the structure of friendship networks in virtual worlds. In: CHI-EA'10, pp. 4027–4032 (2010)
21. Goldreich, O., Ron, D.: On testing expansion in bounded-degree graphs. ECCC, TR00-020 (2000)
22. Gonen, M., Ron, D., Shavitt, Y.: Counting stars and other small subgraphs in sublinear time. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 99–116 (2010)
23. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**, 13–30 (1963)
24. Holland, P.W., Leinhardt, S.: A method for detecting structure in sociometric data. Am. J. Sociol. **76**, 492–513 (1970)
25. Kolda, T.G., Pinar, A., Plantenga, T., Seshadhri, C., Task, C.: Counting triangles in massive graphs with mapreduce. Technical Report 1301.5887, arxiv (2013)
26. Kolountzakis, M., Miller, G., Peng, R., Tsourakakis, C.: Efficient triangle counting in large graphs via degree-based vertex partitioning. In: WAW'10 (2010)
27. Mascarenhas, A., Grout, R.W., Bremer, P.-T., Hawkes, E.R., Pascucci, V., Chen, J.H.: Topological feature extraction for comparison of terascale combustion simulation data. In:

Pascucci, V., Tricoche, X., Hagen, H., Tierny, J. (eds.) Topological Methods in Data Analysis and Visualization. Mathematics and Visualization, pp. 229–240. Springer, Berlin/Heidelberg (2011)
28. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: Simple building blocks of complex networks. Science **298**(5594), 824–827 (2011)
29. Ron, D.: Algorithmic and analysis techniques in property testing. Found. Trends Theor. Comput. Sci. **5**(2), 73–205 (2009)
30. Rubinfeld, R.: Sublinear time algorithms. In: International Conference of Mathematicians (2006)
31. Rubinfeld, R., Sudan, M.: Robust characterization of polynomials with applications to program testing. SIAM J. Comput. **25**, 647–668 (1996)
32. Schank, T., Wagner, D.: Approximating clustering coefficient and transitivity. J. Graph Algorithms Appl. **9**, 265–275 (2005)
33. Schank, T., Wagner, D.: Finding, counting and listing all triangles in large graphs, an experimental study. In: Experimental and Efficient Algorithms, pp. 606–609. Springer, Berlin (2005)
34. Seshadhri, C., Pinar, A., Durak, N., Kolda, T.G.: Directed closure measures for networks with reciprocity. arXiv:1302.6220 (2013)
35. Seshadhri, C., Pinar, A., Kolda, T.G.: Triadic measures on graphs: the power of wedge sampling. In: Proceedings of the SIAM Conference on Data Mining (SDM) (2013)
36. Seshadhri, C., Pinar, A., Kolda, T.G.: Wedge sampling for computing clustering coefficients and triangle counts on large graphs. arXiv:1309.3321 (2013)
37. Son, S., Kang, A., Kim, H., Kwon, T., Park, J., Kim, H.: Analysis of context dependence in social interaction networks of a massively multiplayer online role-playing game. PLoS ONE **7**(4), e33918 (2012)
38. Suri, S., Vassilvitskii, S.: Counting triangles and the curse of the last reducer. In: WWW'11, pp. 607–614 (2011)
39. Szell, M., Thurner, S.: Measuring social dynamics in a massive multiplayer online game. Soc. Netw. **32**, 313–329 (2010)
40. Szell, M., Lambiotte, R., Thurner, S.: Multirelational organization of large-scale social networks in an online world. Proc. Natl. Acad. Sci **107**, 13636–13641 (2010)
41. Thompson, D., Bennett, J., Seshadhri, C., Pinar, A.: A provably-robust sampling method for generating colormaps of large data. In: Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV), Atlanta (2013)
42. Tsourakakis, C.E.: Fast counting of triangles in large real networks, without counting: algorithms and laws. In: ICDM 2008, pp. 608–617 (2008)
43. Tsourakakis, C., Drineas, P., Michelakis, E., Koutis, I., Faloutsos, C.: Spectral counting of triangles in power-law networks via element-wise sparsification. In: ASONAM'09, pp. 66–71 (2009)
44. Tsourakakis, C.E., Kang, U., Miller, G.L., Faloutsos, C.: Doulion: counting triangles in massive graphs with a coin. In: KDD '09, pp. 837–846 (2009)
45. Tsourakakis, C., Kolountzakis, M.N., Miller, G.: Triangle sparsifiers. J. Graph Algorithms Appl. **15**, 703–726 (2011)
46. Vishwanath, V., Hereld, M., Papka, M.E.: Toward simulation-time data analysis and i/o acceleration on leadership-class systems. In: Proc. of IEEE Symposium on Large Data Analysis and Visualization (LDAV) (2011)
47. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge (1994)
48. Watts, D., Strogatz, S.: Collective dynamics of 'small-world' networks. Nature **393**, 440–442 (1998)
49. Yu, W., Wang, C., Grout, R.W., Chen, J.H., Ma, K.-L.: In-situ visualization for large-scale combustion simulations. IEEE Comput. Graph. Appl. **30**, 45–57 (2010)

Part II
Large-Scale Data Analysis: Efficient Representation of Large Functions

Optimal General Simplification of Scalar Fields on Surfaces

Julien Tierny, David Günther, and Valerio Pascucci

1 Introduction

As scientific data-sets become more intricate and larger in size, advanced data analysis algorithms are needed for their efficient visualization. For scalar field visualization, topological analysis techniques have shown to be practical solutions in various contexts by enabling the concise and complete capture of the structure of the input data into high-level *topological abstractions* such as contour trees [7, 9, 10], Reeb graphs [23, 27], or Morse-Smale complexes [18, 19]. Moreover, important advances have been made regarding the analysis of topological noise with the formalism of topological persistence [14], which enabled their multi-resolution representations and consequent progressive data explorations. However, the notion of feature is application-dependent. Using persistence to prioritize topological cancellations can be inappropriate for selecting features of interest in many scenarios (depending on the characteristics of the noise). For this reason, users often employ ad-hoc feature identification strategies that combine several

J. Tierny (✉)
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France

CNRS, UMR 7606, LIP6, F-75005, Paris, France

CNRS LTCI, Telecom ParisTech, 46 Rue Barrault, 75013 Paris, France
e-mail: tierny@telecom-paristech.fr

D. Günther
Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI, 46, Rue Barrault, 75013 Paris, France
e-mail: gunther@telecom-paristech.fr

V. Pascucci
School of Computing and SCI Institute, University of Utah, 72 S Central Campus Dr., Salt Lake City, UT 84112, USA
e-mail: pascucci@sci.utah.edu

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_4

criteria to determine which topological cancellations should be considered signal or noise [10]. While established simplification schemes produce multi-resolution representations of the topological abstractions, they do not focus on generating an actual simplification of the underlying scalar field. However, simplifying the field before any analysis can be beneficial in a number of applications. For example, the complexity of an analysis based on the Morse-Smale complex [18, 19] or a topology similarity estimation [5] strongly depends on the number of critical points. In these applications, reducing the number of critical points in a controlled manner can drastically improve run-time performance.

In this chapter, we present a new combinatorial algorithm which generalizes and extends previous work on topological simplification of scalar fields [4, 26]. Given a scalar field f, our algorithm generates a simplified function g that provably admits only critical points from a constrained subset of the singularities of f while strictly minimizing $||f - g||_\infty$ for data-fitting purposes. Bauer et al. [4] presented such an optimal algorithm in the discrete Morse theory setting for the special case of persistence-driven simplifications. In this chapter, we generalize this work to piecewise linear (PL) scalar fields, which are commonly used in visualization software. In contrast to prior work [2, 4, 15], the proposed simplification scheme works with an arbitrary—not necessarily persistence-based—selection of singularities while still minimizing $||f - g||_\infty$. We illustrate this in several experiments which also show empirically the optimality of the proposed algorithm.

1.1 Related Work

The direct simplification of scalar fields given topological constraints is a subject that has only recently received attention. Existing techniques can be classified into two (complementary) categories.

Numerical approaches aim at approximating a desired solution by solving partial differential equations, where a subset of the input singularities are used as topological constraints while smoothness constraints are often used to enforce geometrical quality. The first work in this direction was presented by Bremer et al. [6], where simplified Morse-Smale complexes are used to guide an iterative and localized simplification of the field based on Laplacian smoothing. In the context of geometry processing, approaches have been presented for the computation of smooth Morse functions with a minimal number of critical points [17, 22]. Patanè et al. [24] presented a general framework for the topology-driven simplification of scalar fields based on a combination of least-squares approximation and Tikhonov regularization. Weinkauf et al. [28] improved the work by Bremer et al. [6] with bi-Laplacian optimization resulting in smoother (C^1) output fields. However, one of the biggest challenge of these approaches is the numerical instability in the optimization process. This may create additional critical points in the output preventing it from strictly conforming to the input constraints. Additionally, the overall optimization process might be computationally expensive resulting in extensive running times.

Combinatorial approaches aim at providing a solution with provable correctness that is not prone to numerical instabilities. In a sense, they can be complementary to numerical techniques by fixing possible numerical issues as a post-process. Edelsbrunner et al. introduced the notion of ϵ-simplification [15]. Given a target error bound ϵ, the goal of their algorithm is to produce an output field everywhere at most ϵ-distant from the input such that all the remaining pairs of critical points have persistence greater than ϵ. Their algorithm can be seen as an extension of early work on digital terrain [1, 25] or isosurface processing [8], where the Contour Tree [9] was used to drive a flattening procedure achieving similar bounds. Attali et al. [2] and Bauer et al. [4] presented independently a similar approach for ϵ-simplification computation. By locally reversing the gradient paths in the field, the authors show that multiple persistence pairs can be cancelled with only one procedure. However, these approaches admit several limitations. Their input is a filtration [14] or a discrete Morse function [16]. Since many visualization software require a PL function, the output needs to be converted into the PL setting requiring a subdivision of the input mesh (one new vertex per edge and per face). However, such a subdivision might increase the size of the mesh by an order of magnitude which is not acceptable in many applications. Also, they focus on the special case where the critical points are selected according to topological persistence. On the other hand, Tierny and Pascucci [26] presented a simple and fast algorithm which directly operates on PL functions enabling an arbitrary selection of critical points for removal. However, this approach does not explicitly minimize the norm $||f - g||_\infty$.

1.2 Contributions

Several strategies can be employed to remove a hill from a terrain (i.e. to remove a maximum-saddle pair from a function, see Fig. 1). For example, a hill can be flattened such that the resulting plateau contains an arbitrarily small monotonic slope towards the saddle ([26], Fig. 1b). Alternatively, one can construct a bridge along integral lines emanating from a saddle while enforcing an arbitrarily small monotonic increase along the bridge ([15], Fig. 1c). In the context of persistence-driven simplification, tight upper bounds for the distance $||f - g||_\infty$ have been shown [11]. The algorithm by Bauer et al. [4] achieves these bounds in the discrete Morse theory setting. In this chapter, we make the following new contributions:

- An algorithm that achieves these bounds for the PL setting;
- An algorithm that minimizes the distance $||f - g||_\infty$ in the case of general simplifications (where the critical points to remove are selected arbitrarily).

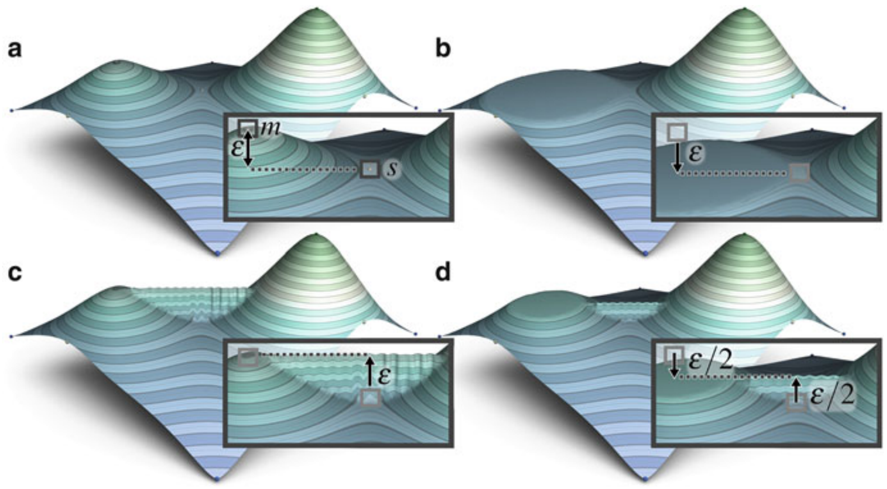


Fig. 1 Removing a maximum-saddle pair (m, s) from a scalar field f such that $|f(m) - f(s)| = \epsilon$ (**a**). A strategy based on *flattening* [26] (**b**) will lower m down to the level of s, yielding $||f - g||_\infty = \epsilon$. A strategy based on *bridging* [15] (**c**) will lift s up to the level of m, yielding $||f - g||_\infty = \epsilon$. A strategy based on a combination of the two [4] (**d**) will lower m halfway down to the level of s while lifting s halfway up to the level of m, yielding a minimized infinity norm $||f - g||_\infty = \epsilon/2$

2 Preliminaries

This section briefly describes our formal setting and presents preliminary results. An introduction to Morse theory can be found in [20].

2.1 Background

The input to our algorithm is a piecewise linear (PL) scalar field $f : \mathscr{S} \rightarrow \mathbb{R}$ defined on an orientable PL 2-manifold $\mathscr{S}$. It has value on the vertices of $\mathscr{S}$ and is linearly interpolated on the simplices of higher dimension. Critical points of PL functions can be classified with simple and inexpensive operations (Fig. 2). The *star* $St(v)$ of a simplex v is the set of simplices σ that contain v as a face. The *link* $Lk(v)$ of a simplex v is the set of simplices in the closure of the star of v that are not also in the star: $Lk(v) = \overline{St(v)} - St(v)$. The *lower link* $Lk^-(v)$ of v is the subset of $Lk(v)$ containing only simplices with all their vertices lower in function value than v: $Lk^-(v) = \{\sigma \in Lk(v) \mid \forall u \in \sigma : f(u) < f(v)\}$. The *upper link* $Lk^+(v)$ is defined by: $Lk^+(v) = \{\sigma \in Lk(v) \mid \forall u \in \sigma : f(u) > f(v)\}$.

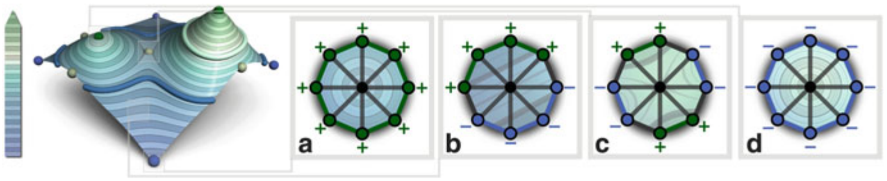


Fig. 2 Scalar field on a terrain (*left*). A level set is shown in *blue*; a contour is shown in *white*. Vertices can be classified according to the connectivity of their lower (*blue*) and upper links (*green*). From left to right: a minimum (**a**), a regular vertex (**b**), a saddle (**c**), a maximum (**d**)

Definition 1 (Critical Point) A vertex v of $\mathcal{S}$ is *regular* if and only if both $Lk^{-}(v)$ and $Lk^{+}(v)$ are simply connected, otherwise v is a critical point of f.

If $Lk^{-}(v)$ is empty, v is a minimum. Otherwise, if $Lk^{+}(v)$ is empty, v is a maximum. If v is neither regular nor a minimum nor a maximum, it is a saddle.

A sufficient condition for this classification is that all the vertices of $\mathcal{S}$ admit distinct f values, which can be obtained easily with symbolic perturbation [12]. To simplify the discussion, we assume that all of the saddles of f are simple (f is then a *Morse* function [20]), and $\mathcal{S}$ is processed on a per connected component basis. The relation between the critical points of the function can be mostly understood through the notions of *Split Tree* and *Join Tree* [9], which respectively describe the evolution of the connected components of the sur- and sub-level sets. Given an isovalue $i \in \mathbb{R}$, the *sub-level set* $L^{-}(i)$ is defined as the pre-image of the open interval $(-\infty, i]$ onto $\mathcal{S}$ through f: $L^{-}(i) = \{p \in \mathcal{S} \mid f(p) \leq i\}$. Symmetrically, the *sur-level set* $L^{+}(i)$ is defined by $L^{+}(i) = \{p \in \mathcal{S} \mid f(p) \geq i\}$. The Split Tree $\mathcal{T}_{+}$ of f is a 1-dimensional simplicial complex obtained by contracting each connected component of the sur-level set to a point. By continuity, two vertices v_a and v_b of $\mathcal{S}$ with $f(v_a) < f(v_b)$ are mapped to adjacent vertices in $\mathcal{T}_{+}$ if and only if for each $v_c \in \mathcal{S}$ such that $f(v_c) \in (f(v_a), f(v_b))$, there holds:

- the connected component of $L^{+}(f(v_a))$ which contains v_a also contains v_b;
- the connected component of $L^{+}(f(v_c))$ which contains v_c does not contain v_b.

By construction, a bijective map $\phi_{+} : \mathcal{S} \to \mathcal{T}_{+}$ exists between the vertices of $\mathcal{S}$ and those of $\mathcal{T}_{+}$. Hence, for conciseness, we will use the same notation for a vertex either in $\mathcal{S}$ or in $\mathcal{T}_{+}$. Maxima of f as well as its global minimum are mapped in $\mathcal{T}_{+}$ to valence-1 vertices, while saddles where k connected components of sur-level sets merge are mapped to valence-$(k+1)$ vertices. All the other vertices are mapped to valence-2 vertices. A *super-arc* (v_a, v_b) [9] is a directed connected path in $\mathcal{T}_{+}$ from v_a to v_b with $f(v_a) > f(v_b)$ such that v_a and v_b are the only non-valence-2 vertices of the path. The Join Tree $\mathcal{T}_{-}$ is defined symmetrically with the sub-level sets of f.

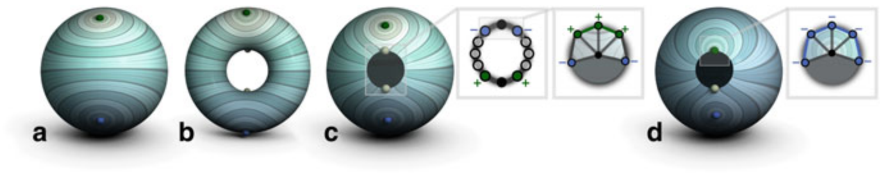


Fig. 3 Non removable critical points: (**a**) a global minimum and a global maximum have to be maintained for the field not to be constant. (**b**) 2 $g_{\mathscr{S}}$ saddles cannot be removed. (**c**), (**d**) Each boundary component has two non-removable global *stratified* extrema, which turn into non-removable saddles (**c**) or (possibly) *exchangeable* extrema (**d**)

2.2 General Simplification of Scalar Fields on Surfaces

Definition 2 (General Topological Simplification) Given a field $f : \mathscr{S} \rightarrow \mathbb{R}$ with its set of critical points $\mathscr{C}_f$, we call a *general simplification* of f a scalar field $g : \mathscr{S} \rightarrow \mathbb{R}$ such that the critical points of g form a sub-set of $\mathscr{C}_f$: $\mathscr{C}_g \subseteq \mathscr{C}_f$.

In other words, a general simplification consists in constructing a close variant of the input field f from which a set of critical points has been removed. We call it *optimal* if it additionally minimizes the infinity norm $||f - g||_\infty$. As described by Tierny and Pascucci [26], critical points can only be removed in extrema-saddle pairs. Hence, the removal of the saddles of f is completely dependent on the removal of its extrema. Note that there are also critical points that can not be removed due to the topology of $\mathscr{S}$ (summarized in Fig. 3). We call them *non-removable* critical points. More details on the influence of the topology of $\mathscr{S}$ on the simplification of critical points can be found in [26].

3 Algorithm

In this section, we present our new algorithm for the computation of optimal general simplifications. Given some input constraints $\mathscr{C}_g^0$ and $\mathscr{C}_g^2$, i.e., the minima and the maxima of g, our algorithm reconstructs a function g which satisfies these topological constraints and minimizes $||f - g||_\infty$. Saddles are implicitly removed by our algorithm due to their dependence on the minima and maxima removal.

To guarantee that the input field admits distinct values on each vertex, symbolic perturbation is used. In addition to its scalar value, each vertex v is associated with an integer *offset* $\mathscr{O}(v)$ initially set to the actual offset of the vertex in memory. When comparing two vertices (e.g., critical point classification), their order is disambiguated by their offset $\mathscr{O}$ if these share the same scalar value. Our new algorithm modifies the scalar values of the vertices (Sects. 3.1–3.3), while the algorithm by Tierny and Pascucci [26] is used in a final pass to update the offsets.

In the following sub-sections, we describe the case $\mathscr{C}_g^0 = \mathscr{C}_f^0$ (only maxima are removed). The removal of the minima is a symmetrical process. We begin with the simple case of the pairwise critical point removal before we go into more complex and general scenarios addressing the removal of multiple critical points. The overall algorithm is summarized at the end of Sect. 3.3.

3.1 Optimal Pairwise Removal

Let C_g^2 be equal to $C_f^2 \setminus \{m\}$ where m is a maximum to remove. As discussed in [26], m can only be removed in pair with a saddle s where the number of connected components in the sur-level set changes (i.e. a valence-3 vertex in $\mathscr{T}_+$). Moreover, a necessary condition for a critical point pair to be cancelled is the existence of a gradient path linking them [21]. In the PL setting, these are connected PL 1-manifolds called *integral lines* [13]. Thus, m can only be removed with a valence-3 vertex $s \in \mathscr{T}_+$ that admits a forward integral line ending in m. Let $S(m)$ be the set of all saddles satisfying these requirements. Since integral lines are connected, m must belong to the connected components of $L^+(f(s))$ which also contains $s \in S(m)$. In $\mathscr{T}_+$, the saddles of $S(m)$ are the valence-3 vertices on the connected path from m down to the global minimum M of f.

To cancel a pair (m, s), one needs to assign a unique target value t to m and s. Since m is the only extremum to remove and m and s are the extremities of a monotonic integral line, we have:

$$||f - g||_\infty = \max(|f(m) - t|, |f(s) - t|) \tag{1}$$

The optimal value t^* which minimizes (1) is $f(m) - |f(m) - f(s)|/2$. Hence, we need to find the saddle $s^* \in S(m)$ that minimizes $|f(m) - f(s)|$. Since the saddles of $S(m)$ lay on a connected path from m to M in $\mathscr{T}_+$, the optimal saddle s^* is the extremity of the only super-arc containing m.[1]

Let $\mathscr{F}$ be the set of vertices of $\mathscr{S}$ mapped by ϕ_+ to the *super-arc* (m, s^*). Let $\mathscr{B}$ be the forward integral lines emanating from s^*. The pair (m, s^*) can then be removed by setting them to the value t^* such that no new critical point is introduced. This can be guaranteed by enforcing monotonicity on $\{\mathscr{F} \cup \mathscr{B}\}$: Our algorithm assigns the target value t^* to any vertex of $\mathscr{F}$ which is higher than t^* and to any vertex of $\mathscr{B}$ which is lower than t^* (see Fig. 4). Thus, given only one maximum to remove, our algorithm produces an optimal general simplification g.

[1]Note that the extremity s of the super-arc (m, s) admits a forward integral line ending in m.

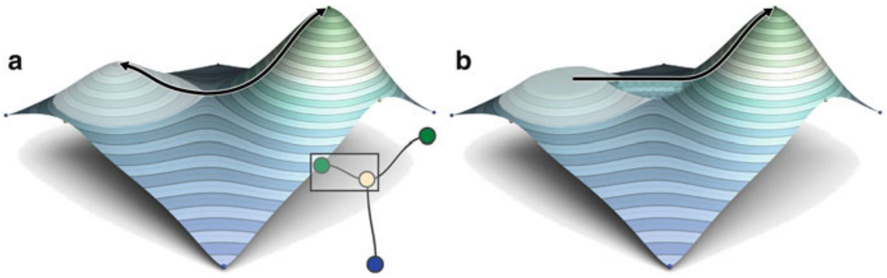


Fig. 4 (**a**) The set of flattening vertices $\mathscr{F}$ is identified from $\mathscr{T}_+$ (*transparent white*). The set of bridging vertices $\mathscr{B}$ is identified through discrete integral line integration (*black curve*). (**b**) The function values of each vertex of $\mathscr{F}$ and $\mathscr{B}$ is updated to produce the simplified function

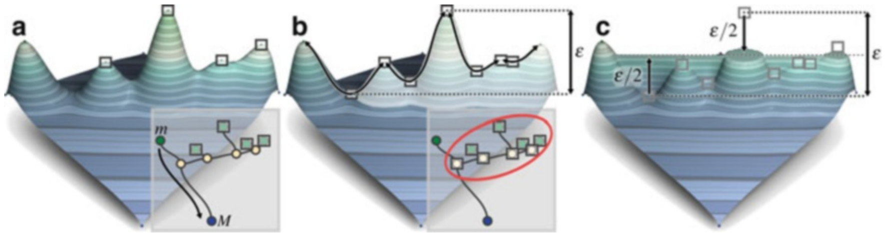


Fig. 5 Optimal simplification of a sub-tree T_k of the split tree $\mathscr{T}_+$. (**a**) A set of maxima corresponding to the leaves of a connected sub-tree T_k is selected. (**b**) The optimal set of saddles to remove can be identified with a simple traversal of $\mathscr{T}_+$. Cancelling the critical points of T_k requires to process a function interval of $\epsilon = |f(m^*) - f(s^*)|$, where m^* and s^* are respectively the highest maximum and the lowest saddle of T_k. The set of candidate vertices $\mathscr{F}$ for flattening is directly identified from T_k. The set of candidate vertices for bridging $\mathscr{B}$ is identified by discrete integral lines emanating from the saddles of T_k. (**c**) Updating the function values of $\mathscr{F}$ and $\mathscr{B}$ yields an infinity norm of $\epsilon/2$: by lifting s^* up by $\epsilon/2$ and by lowering m^* down by $\epsilon/2$

3.2 Optimal Sub-tree Removal

We call a *sub-tree* T_k of $\mathscr{T}_+$ a maximally connected sub-set of $\mathscr{T}_+$ such that: it contains *(a)* k maxima of f to remove and *(b)* k valence-3 vertices of $\mathscr{T}_+$, and that *(c)* for all the valence-3 vertices s_i of T_k except the lowest, there exists no maximum m to maintain such that s_i belongs to the connected path on $\mathscr{T}_+$ from m down to M (Fig. 5a). The optimal simplification of a sub-tree is a generalization of the previous case. By using the pairing strategy described in the previous sub-section, one can process the maxima of T_k in arbitrary order. The maxima are paired with valence-3 vertices of $\mathscr{T}_+$ and the corresponding super-arcs are removed. The resulting paired saddles will always be valence-3 vertices of T_k irrespectively of the order in which the maxima are processed.

Let $\mathscr{F}$ be the pre-image of ϕ_+ restricted to the super-arcs of T_k. Let $\mathscr{B}$ be the forward integral lines emanating from the valence-3 vertices of T_k. By construction

$\{\mathscr{F} \cup \mathscr{B}\}$ is a connected component, see Fig. 5b, from which we aim to remove all the critical points. Similar as in the previous sub-section, these can be cancelled by assigning them a common target value t^* while enforcing monotonicity on $\{\mathscr{F} \cup \mathscr{B}\}$ (no new critical point should be added).

For a given target value t, we have $||f - g||_\infty = max(|f(m^*) - t|, |f(s^*) - t|)$ with m^* and s^* being the highest maximum and the lowest saddle in T_k, respectively. The target value t^* which minimizes $||f - g||_\infty$ is then $t^* = f(s^*) + |f(m^*) - f(s^*)|/2$. Thus, our algorithm assigns the target value t^* to any vertex of $\mathscr{F}$ which is higher than t^* and to any vertex of $\mathscr{B}$ which is lower than t^*.

All the sub-trees T_k of $\mathscr{T}_+$ can be identified with one breadth-first search traversal of $\mathscr{T}_+$ seeded at the maxima to remove, in order of decreasing f value. In this traversal, only the maxima to remove and the valence-3 vertices are admissible. Two connected components (seeded at the maxima to remove) can merge if there exists a super-arc between them. A connected components stops its growth if its number of maxima to remove equals the number of its valence-3 vertices. At the end of the traversal, each remaining component forms a maximally connected sub-tree T_k.

3.3 Optimal Sub-tree Sequence Removal

In this sub-section, we finally describe the optimal simplification of a sequence of sub-trees (corresponding to the most general case, where maxima can be selected for removal arbitrarily).

For a given set of maxima to remove (Fig. 6a), the corresponding maximally connected sub-trees can be identified with the algorithm described in the previous sub-section. Moreover, it is possible to compute their individual optimal target values $\{t_k\}$ that creates optimal simplifications of the sub-trees $\{T_k\}$. To guarantee the monotonicity of the function, special care needs to be given to sub-trees that are adjacent to each other but separated by a super-arc, see Fig. 6b.

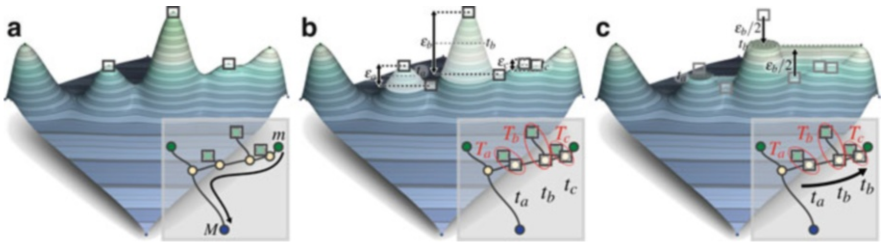


Fig. 6 Optimal simplification of a sequence of sub-trees (**a**). While each sub-tree T_i can be individually simplified at its optimal target value t_i (**b**), monotonicity has to be enforced by selecting for each sub-tree the maximum value among its own target value and its adjacent parent's (**c**)

Let T_0 and T_1 be two sub-trees such that s_0 and s_1 are the lowest valence-3 vertices of T_0 and T_1, respectively. Additionally, let s_0 and s_1 be connected by a super-arc (s_1, s_0) with $f(s_1) > f(s_0)$. Since T_0 and T_1 are adjacent yet distinct maximally connected sub-trees, there exists at least one maximum m to preserve with $f(m) > f(s_1) > f(s_0)$ such that s_0 and s_1 both belong to the directed connected path from m down to the global minimum M of f, see Fig. 6. Hence, in contrast to the previous sub-section, monotonicity should additionally be enforced on the connected path on $\mathscr{T}_+$ from m down to M.

Let $\mathscr{F}$ be the pre-image through ϕ_+ of the super-arcs of T_0 and T_1 and $\mathscr{B}$ the forward integral lines emanating from the valence-3 vertices of T_0 and T_1. Since T_0 and T_1 are adjacent, $\{\mathscr{F} \cup \mathscr{B}\}$ is again a connected component on which g has to be monotonically increasing. Two cases can occur:

1. $t_0 < t_1$: Simplifying the sub-trees T_0 and T_1 at their individual optimal target values t_0 and t_1 yields a monotonically increasing function on $\{\mathscr{F} \cup \mathscr{B}\}$ (An example is given in Fig. 6b for the case $T_0 = T_a$ and $T_1 = T_b$);
2. $t_0 > t_1$: Simplifying the sub-trees T_0 and T_1 at t_0 and t_1 would yield a decreasing function, and hence introduce a new critical point on $\{\mathscr{F} \cup \mathscr{B}\}$. (An example is given in Fig. 6b for the case $T_0 = T_b$ and $T_1 = T_c$). In this case, forcing T_1 to use t_0 as a target value will correctly enforce monotonicity while not changing the distance $||f - g||_\infty$,[2] see Fig. 6c.

Hence, the optimal target value needs to be propagated along adjacent sub-trees to enforce monotonicity. To do so, $\mathscr{T}_+$ is traversed by a breadth-first search with increasing f value. When traversing a vertex s_1, which is the lowest valence-3 vertex of a sub-tree T_1, the algorithm checks for the existence of a super-arc (s_1, s_0) such that s_0 is the lowest valence-3 vertex of a sub-tree T_0. The optimal target value t_1 is updated to enforce monotonicity: $t_1 \leftarrow max(t_0, t_1)$. Note that this monotonicity enforcement among the target values does not change $||f - g||_\infty$. Hence, an optimal simplification is guaranteed. In particular, $||f - g||_\infty$ will be equal to $|f(m^*) - f(s^*)|/2$ with s^* and m^* being the lowest valence-3 and the highest valence-1 vertex of the sub-tree T^* which maximizes $|f(m^*) - f(s^*)|$. In case of persistence-guided simplification, the simplified function g achieves the upper bound $||f - g||_\infty = \epsilon/2$ with $\epsilon = |f(m^*) - f(s^*)|$.

In conclusion, the overall algorithm for optimal simplification can be summarized as follows:

1. Identifying the sub-trees to remove (Sect. 3.2);
2. Enforcing monotonicity on the target values of each sub-tree (Sect. 3.3);
3. Cancelling each sub-tree to its target value with a combination of flattening and bridging (Sects. 3.1 and 3.2).

[2]Since $f(s_0) < f(s_1)$ and $t_0 > t_1$, then $|f(s_0) - g(s_0)| = t_0 - f(s_0) > t_1 - f(s_1) = |f(s_1) - g(s_1)|$. Thus, if T_0 and T_1 are the only sub-trees, $||f - g||_\infty = t_0 - f(s_0)$.

4. Running the algorithm by Tierny and Pascucci [26] as a post-process pass to disambiguate flat plateaus. This last pass is a crucial step which is mandatory to guarantee the topological correctness in the PL sense of the output.

The optimal simplification of the function given some minima to remove is obtained in a completely symmetric fashion by considering the Join Tree $\mathscr{T}_-$.

4 Results and Discussion

In this section, we present results of our algorithm obtained with a C++ implementation on a computer with an i7 CPU (2.93 GHz). In the following, the range of the height functions f exactly spans the interval [0, 100] for all data-sets.

Computational Complexity The construction of the Split and Join Trees takes $O(n\log(n)) + (n+e)\alpha(n+e)$ steps [9], where n and e are respectively the number of vertices and edges in $\mathscr{S}$ and $\alpha(\cdot)$ is an exponentially decreasing function (inverse of the Ackermann function). The different tree traversals required to identify the sub-trees to remove and to propagate the target values take at most $O(n\log(n))$ steps. Updating the function values of the vertices for flattening and bridging takes linear time. The algorithm by Tierny and Pascucci [26] employed to post-process the offset values takes $O(n^2\log(n))$ steps. Hence, it constitutes the bottleneck of our overall algorithm. However, scalability experiments for this latter part [26] exhibits $O(n\log(n))$ performances in practice. Thus, the overall time-complexity of our algorithm is $O(n\log(n))$ in practice. Although our algorithm admits linear memory-complexity, we found that it first becomes memory-limited before being compute-limited due to its practical time efficiency.

Persistence-Driven Simplification is well understood in terms of infinity norm [4, 11]. We start our infinity norm evaluation with this setting to verify that our algorithm meets these expectations. As shown in Fig. 7, our algorithm improves the infinity norm in comparison with an algorithm solely based on flattening [26]. Given a persistence threshold ϵ, the output g generated by our algorithm satisfies $||f-g||_\infty = \frac{\epsilon^*}{2}$ if the most persistent pair selected for removal has persistence $\epsilon^* \leq \epsilon$.

General Simplification aims for arbitrary critical point removal. Figure 8 shows an example where the user interactively selected extrema to remove. Even in this setting, our algorithm improved $||f-g||_\infty$ by a factor of two compared to [26].

Empirical optimality of our algorithm is illustrated in the last part of the experimental section. We provide practical evidence for the minimization of $||f-g||_\infty$. As shown in Sect. 3.1 and in Fig. 1, an extremum-saddle pair can be removed optimally in a localized fashion. Hence, a general simplification can be achieved through a sequence of optimal pairwise removals for a given list of extrema to remove. However, such a general simplification is *not* necessarily optimal as shown in Fig. 9.

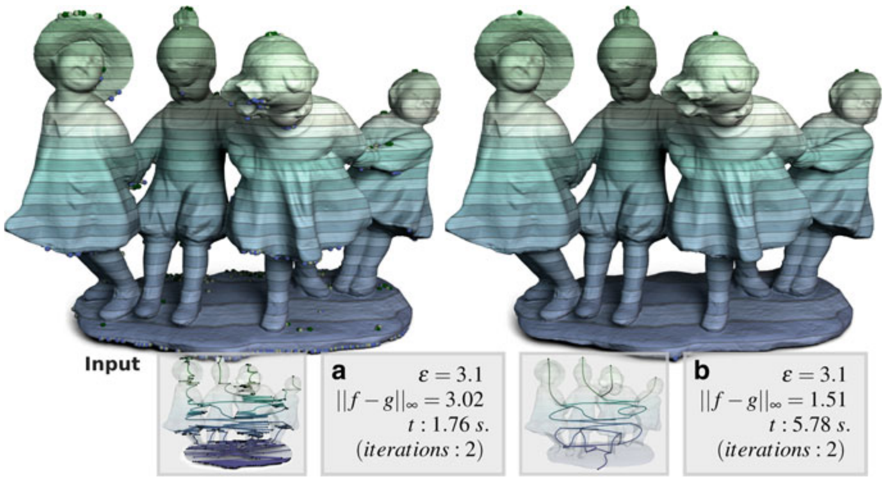


Fig. 7 Comparison of the simplifications obtained with the algorithm proposed in [26] (**a**), ($||f - g||_\infty \leq \epsilon$) and our new algorithm (**b**), ($||f - g||_\infty \leq \epsilon/2$). Critical points are removed based on topological persistence (the persistence threshold is ϵ). The topology of the fields is summarized with the inset Reeb graph for illustration purpose (input surface: 725k vertices)

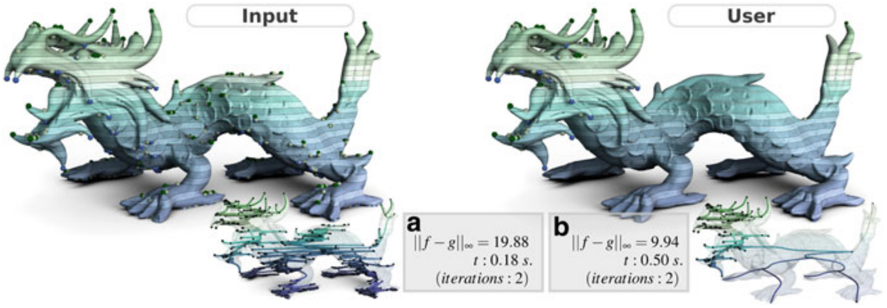


Fig. 8 User driven simplification. The statistics of the simplification are shown in the *grey frames* for the flattening-only algorithm [26] (**a**) and for our new algorithm (**b**). The topology of the fields is summarized with the *inset* Reeb graph for illustration purpose (input surface: 75k vertices)

The distance $||f - g||_\infty$ is even depending on the order of extrema-saddle pair removals. To explore this space of optimal pairwise removal sequences in a Monte-Carlo fashion, we computed 100 sequences of pairwise removals ordered randomly for several data-sets (Figs. 6, 7, and 8) with given sets of extrema to remove. Table 1 shows the minimum, average, and maximum distances for each of the examples. The minimum distance $||f - g||_\infty$ obtained with this Monte-Carlo strategy was never smaller than the distance obtained by our global algorithm. This illustrates that there exists no sequence of optimal pairwise removals that results in a smaller distance $||f - g||_\infty$ than our algorithm. This shows empirically its optimality.

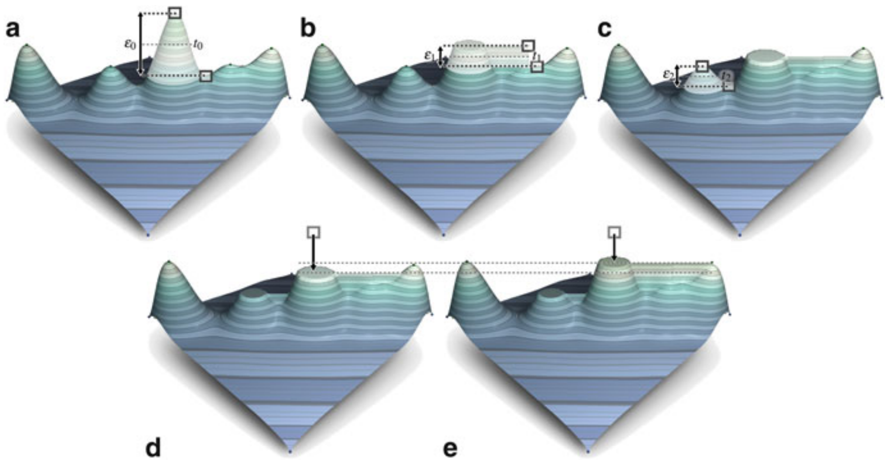


Fig. 9 An arbitrary sequence of pairwise optimal simplifications (**a**–**d**) does *not* necessarily produce an *optimal* simplification. In this example, the global maximum is moved lower (**d**) than it would have been with our global algorithm (**e**). This results in a higher distance with regard to the infinity norm (**d**) ($||f - g||_\infty = 34.83$, (**e**) $||f - g||_\infty = 26.02$)

Table 1 Distance $||f - g||_\infty$ obtained with our algorithm (top) and with sequences of optimal pairwise simplifications (bottom 3) on several data-sets for a given constrained topology

	Terrain (Fig. 6)	Children (Fig. 7)	Dragon (Fig. 8)
Global algorithm	26.02	1.51	9.94
Pairwise sequences (Minimum)	26.02	1.51	9.94
Pairwise sequences (Average)	30.57	1.55	13.20
Pairwise sequences (Maximum)	34.83	1.58	17.04

For simplifications based on pairwise sequences (bottom 3), the order of simplification of the critical point pairs is defined randomly (100 runs per data-sets)

Limitations Although our algorithm achieves the same time complexity as the flattening algorithm [26], our new algorithm is more computationally expensive, in practice (Figs. 7 and 8). This is due the use of the flattening algorithm in the final pass and the necessity of the Join and Split tree computations (which take longer than the flattening algorithm in practice). In the general case, our algorithm may change the value of the maintained critical points after simplification. For instance, if the lowest minimum of $\mathscr{C}_g^0$ is initially higher than the highest maximum of $\mathscr{C}_g^2$, the algorithm will change their values to satisfy the topological constraints. Noisy data may also contain multi-saddles. To account for these, in the sub-tree removal procedure (Sect. 3.2), one must construct a sub-tree T_k by considering k maxima and a set of saddles whose *multiplicity* sums to k. Moreover, a p-multi-saddle must be paired with p maxima to be removed. The rest of the procedure is identical. Alternatively, multi-saddles can be unfolded in a pre-processing step [13]. Finally, our algorithm provides strong guarantees on the topology of the output and on $||f - g||_\infty$ at the expense of geometrical smoothness.

Conclusion
In this chapter, we have presented a new combinatorial algorithm for the optimal general simplification of piecewise linear scalar fields on surfaces. It improves over state-of-the art techniques by minimizing the norm $||f - g||_\infty$ in the PL setting and in the case of general simplifications (where critical points can be removed arbitrarily). Experiments showed the generality and efficiency of the algorithm, and demonstrated in practice the minimization of $||f - g||_\infty$. Such an algorithm can be useful to speed up topology analysis algorithms or to fix numerical instabilities occurring in the solve of numerical problems on surfaces (gradient field integration, scale-space computations, PDEs, etc.). Moreover, our algorithm provides a better data fitting than the flattening algorithm [26] since it minimizes $||f-g||_\infty$. A natural direction for future work is the extension of this approach to volumetric data-sets. However, this problem is NP-hard as recently shown by Attali et al. [3]. This indicates that the design of a practical algorithm with strong topological guarantees is challenging.

Acknowledgements Data-sets are courtesy of AIM@SHAPE. This research is partially funded by the RTRA Digiteo through the *unTopoVis* project (2012-063D). The authors thank Hamish Carr for insightful comments and suggestions.

References

1. Agarwal, P.K., Arge, L., Yi, K.: I/O-efficient batched union-find and its applications to terrain analysis. In: ACM Symposium on Computational Geometry, pp. 167–176 (2006)
2. Attali, D., Glisse, M., Hornus, S., Lazarus, F., Morozov, D.: Persistence-sensitive simplification of functions on surfaces in linear time. In: TopoInVis Workshop (2009)
3. Attali, D., Bauer, U., Devillers, O., Glisse, M., Lieutier, A.: Homological reconstruction and simplification in R3. In: ACM Symposium on Computational Geometry, pp. 117–126 (2013)
4. Bauer, U., Lange, C., Wardetzky, M.: Optimal topological simplification of discrete functions on surfaces. Discrete Comput. Geom. **47**, 347–377 (2012)
5. Bauer, U., Ge, X., Wang, Y.: Measuring distance between Reeb graphs. In: ACM Symposium on Computational Geometry (2014)
6. Bremer, P.-T., Edelsbrunner, H., Hamann, B., Pascucci, V.: A topological hierarchy for functions on triangulated surfaces. IEEE Trans. Vis. Comput. Graph. **10**, 385–396 (2004)
7. Bremer, P.-T., Weber, G., Tierny, J., Pascucci, V., Day, M., Bell, J.: Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. IEEE Trans. Vis. Comput. Graph. **17**, 1307–1324 (2011)
8. Carr, H.: Topological Manipulation of Isosurfaces. Ph.D. thesis, UBC (2004)
9. Carr, H., Snoeyink, J., Ulrike, A.: Computing contour trees in all dimensions. In: Proceedings of Symposium on Discrete Algorithms, pp. 918–926 (2000)

10. Carr, H., Snoeyink, J., van de Panne, M.: Simplifying flexible isosurfaces using local geometric measures. In: Proc. IEEE Vis, pp. 497–504 (2004)
11. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of persistence diagrams. Discrete Comput. Geom. **37**, 103–120 (2007)
12. Edelsbrunner, H., Mucke, E.P.: Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. ACM Trans. Graph. **9**, 66–104 (1990)
13. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical Morse complexes for piecewise linear 2-manifolds. In: ACM Symposium on Computational Geometry, pp. 70–79 (2001)
14. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. Discrete Comput. Geom. **28**, 511–533 (2002)
15. Edelsbrunner, H., Morozov, D., Pascucci, V.: Persistence-sensitive simplification of functions on 2-manifolds. In: ACM Symposium on Computational Geometry, pp. 127–134 (2006)
16. Forman, R.: A user's guide to discrete Morse theory. Adv. Math **134**, 90–145 (1998)
17. Gingold, Y.I., Zorin, D.: Controlled-topology filtering. Comput. Aided Des. **39**, 676–684 (2006)
18. Günther, D., Reininghaus, J., Wagner, H., Hotz, I.: Efficient computation of 3D Morse-Smale complexes and persistent homology using discrete Morse theory. Vis. Comput. **28**, 959–969 (2012)
19. Gyulassy, A., Bremer, P.-T., Hamann, B., Pascucci, P.: A practical approach to Morse-Smale complex computation: scalabity and generality. IEEE Trans. Vis. Comput. Graph. **14**, 1619–1626 (2008)
20. Milnor, J.: Morse Theory. Princeton University Press, Princeton (1963)
21. Milnor, J.W.: Lectures on the H-Cobordism Theorem. Princeton University Press, Princeton (1965)
22. Ni, X., Garland, M., Hart, J.: Fair Morse functions for extracting the topological structure of a surface mesh. ACM Trans. Graph. Proc. ACM SIGGRAPH **23**, 613–622 (2004)
23. Pascucci, V., Scorzelli, G., Bremer, P.T., Mascarenhas, A.: Robust on-line computation of Reeb graphs: simplicity and speed. ACM Trans. Graph. Proc. ACM SIGGRAPH **26**, 58 (2007)
24. Patanè, G., Falcidieno, B.: Computing smooth approximations of scalar functions with constraints. Comput. Graph. **33**, 399–413 (2009)
25. Soille, P.: Optimal removal of spurious pits in digital elevation models. Water Res. Res. **40**, W12509 (2004)
26. Tierny, J., Pascucci, V.: Generalized topological simplification of scalar fields on surfaces. IEEE Trans. Vis. Comput. Graph. **18**, 2005–2013 (2012)
27. Tierny, J., Gyulassy, A., Simon, E., Pascucci, V.: Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. IEEE Trans. Vis. Comput. Graph. **15**, 1177–1184 (2009)
28. Weinkauf, T., Gingold, Y., Sorkine, O.: Topology-based smoothing of 2D scalar fields with C^1-continuity. Comput. Graph. Forum Proc. Euro.Vis. **29**, 1221–1230 (2010)

Piecewise Polynomial Monotonic Interpolation of 2D Gridded Data

Léo Allemand-Giorgis, Georges-Pierre Bonneau, Stefanie Hahmann, and Fabien Vivodtzev

1 Introduction

Preserving meaningful features in scalar fields when simplifying the data set is a requirement in Scientific Visualization for efficiently visualizing and understanding very large data. One class of meaningful features of a scalar function is locations and values of local extrema. Large isolated extrema are indeed salient features and have usually an important meaning. So they have to be preserved in a visualization, whereas nearby local extrema with similar values can be neglected in many application domains. It is even better to remove these spurious extrema in order to enhance the visibility of significant features and thus to improve the understanding of the data set. It is equally important to avoid adding spurious extraneous features when visualizing a data set. In particular if a data set samples a function without local extrema, then the visualization of this discrete data should be free of local extrema as well.

Morse theory [17] is an example of a concept dealing with critical points and their importance. The Morse-Smale (MS) complex, which is based on Morse theory, segments the domain into a set of regions inside which the function is monotonous. A combinatorial simplification of the MS complex has been introduced in [6]. The MS complex is simplified by removing adjacent pairs of critical points in the complex while preserving most significant critical points. Then the original data is however not coherent anymore with the simplified complex, because monotonicity

L. Allemand-Giorgis • G.-P. Bonneau (✉) • S. Hahmann
Inria - Laboratoire Jean Kuntzmann, University of Grenoble, Grenoble, France
e-mail: leo.allemand-giorgis@inria.fr; Georges-Pierre.Bonneau@inria.fr; Stefanie.Hahmann@inria.fr

F. Vivodtzev
CEA: French Atomic Energy Commission and Alternative Energies, Le Barp, France
e-mail: Fabien.Vivodtzev@cea.fr

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_5

got lost inside the MS cells. The new data should be monotonic inside a region. So, techniques which compute monotonic data knowing some values of the function on a grid are needed.

This paper presents a novel approach for computing monotone scalar functions interpolating gridded 2D data sets using smooth piecewise polynomial representations. We built the interpolant in such a way that no local extrema exist in the interior of the function domain if the input data fulfills a simple monotonicity criterion. In contrast to prior related works we do not require the input data to be axial monotonic, meaning that the data does not have to be strictly monotonous along all grid rows and columns, instead we base on a less restrictive monotonicity criterion. In this paper we make the following contributions:

– A monotonicity constraint is used which is more general than the standard axial monotonicity for tensor-product surfaces.
– In concordance with the monotonicity constraint we introduce a modified Sibson split interpolant.
– We derive sufficient conditions on the partial derivatives to ensure monotonicity of the interpolating function.
– We then develop two algorithms to effectively construct a monotone C^1 surface composed of cubic triangular Bézier surfaces.

The main contribution of this paper is a new piecewise polynomial monotonic interpolant. It has been inspired by earlier work in the field of Constrained Shape Design within Computer Aided Geometric Design (CAGD). In this field the objective is to interpolate or approximate a set of points by analytically defined curves or surfaces with a given constraint such as positivity, convexity or monotonicity. Our new interpolant is based on relaxed monotonicity constraints that still ensure that no critical points exist in the interior of the function domain.

The remainder of the present paper is structured as follows. In Sect. 2 we review related works dealing with shape preserving interpolation. Section 3 presents our contributions on piecewise polynomial monotone interpolation. We first introduce our new interpolant. It is based on a relaxed monotonicity constraint. We give sufficient conditions ensuring that no critical point exists in the interior of the domain with our new interpolant. We then develop two algorithms to effectively compute monotone interpolants. Finally we show results in Sect. 4 and conclude the paper in section “Conclusion” with a discussion on future directions of research.

2 Related Works

Shape preserving interpolation is a well studied problem in scientific literature. It consists of computing a curve or surface which interpolates given scalar data while preserving the shape of the underlying data.

The problem we are dealing with in the present paper is closely related to shape preserving interpolation. Indeed, we aim to construct a function free of critical points interpolating a maximum and a minimum function value at opposite vertices of a rectangular domain. Since a monotone surface is sufficient for this purpose, we derive an algorithm which is able to construct a surface that preserves monotonicity along a diagonal direction.

Convexity [3,4,13] and monotonicity are typical shape properties to be preserved. Concerning monotonicity preserving surface fitting most research focussed on monotone bivariate interpolation. In [1, 2, 12] sufficient (and sometimes also necessary) conditions were derived for the monotonicity of piecewise polynomial patches interpolating data given at the grid points of a rectangular mesh. These conditions were transformed into a system of linear inequalities which in turn formed the basis for an algorithm. Since the interpolating functions are determined by a function value and first order derivatives at the grid points, the algorithms compute feasible solutions of the derivatives. All these methods provide surfaces preserving axial monotonicity. Even though our method is similar to [12], we base it on a relaxed monotonicity preservation in diagonal direction only. This is sufficient for our goal to interpolate local extrema without any other critical point inside the function domain.

In [9] it was shown that tensor product Berstein and B-spline bases preserve monotonicity in all directions. In [10] three kinds of monotonicity preservation of systems of bivariate functions on a triangle are studied. We also use the fact that Bernstein polynomials on triangles preserve monotonicity. However, we do not only derive sufficient conditions for monotonicity, in addition we provide an effective algorithm to compute a monotone surface. Our approach splits the rectangular domain into four cubic triangle patches and computes the Bézier control points with a modified Sibson split in order to get a globally C^1-continuous surface.

Let us finally mention that monotonicity preservation has also been investigated for scattered data approximation [19], for subdivision curves [11, 14], convolution of B-splines [18], for rational surfaces [5] and for non-polynomial functions [15].

3 Monotonic Polynomial Interpolation

In this section we propose a novel solution to the following problem.

Problem Given a 2D rectangular grid of scalar values sampled from a monotone function, compute a smooth interpolating function which is monotone as well.

This is a typical problem encountered in *Shape Preserving* interpolation methods [16], where the general goal is to compute an interpolating surface that mimics the shape of the input data.

The method we present in this paper is inspired by the algorithms given in [1, 2, 12], where C^1 monotone increasing spline functions interpolating gridded data are constructed by iteratively adjusting initially estimated gradient values at each grid point. It is a kind of Hermite interpolation, where the gradient values are adjusted to ensure monotonicity. This idea can easily be illustrated on 1D cubic Hermite interpolation of monotone data, where the choice of the derivative (slopes of the tangents) at these data points decide whether the interpolating function is monotonic or not, see Fig. 1.

We use piecewise polynomial functions defined on a triangular subdivision of the gridded domain, where each rectangle is subdivided into four triangles by drawing the main diagonals, as shown in Fig. 2. Each polynomial piece of function $f : T \to \mathbb{R}$ is a cubic triangular Bézier surface (Fig. 3)

$$f(\tau) = \sum_{\substack{i+j+k=3 \\ i,j,k \geq 0}} b_{ijk} B^3_{ijk}(\tau), \qquad \tau \in T$$

defined on a domain triangle $T \subset \mathbb{R}^2$ given by three non-colinear points p_0, p_1, p_2. $\tau = (\tau_0, \tau_1, \tau_2)$ is the triplet of barycentric coordinates of a point in T. The 10 coefficients $b_{ijk} \in \mathbb{R}$ are called the *Bézier ordinates* of f corresponding to the parameter values $(i/3, j/3, k/3)$. $B^3_{ijk}(\tau) = \frac{3!}{i!j!k!}\tau_0^i \tau_1^j \tau_2^k$ are the Berstein polynomials of degree 3. All fundamentals on triangular Bézier surfaces can be found in [7].

The polynomial pieces of our surface are computed by interpolating Hermite data (i.e. function values and partial derivatives) given at the grid points using a modified Sibson split interpolant, as explained in the next section. An advantage of using Hermite data is that C^1-continuity can be ensured more easily. The problem we solve is therefore to find upper bounds on the gradients, which guarantee that the

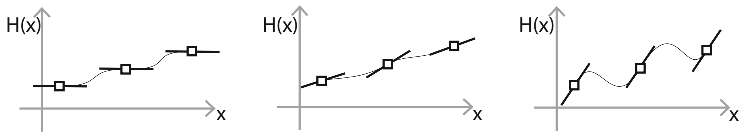


Fig. 1 Cubic Hermite interpolation of monotone increasing data. Different derivatives prescribed at the data points lead to different curves. Setting derivatives equal zero produces a monotone increasing curve (*left*), but critical points are generated. Monotonicity of the function gets lost when the derivative values are too big (*right*)

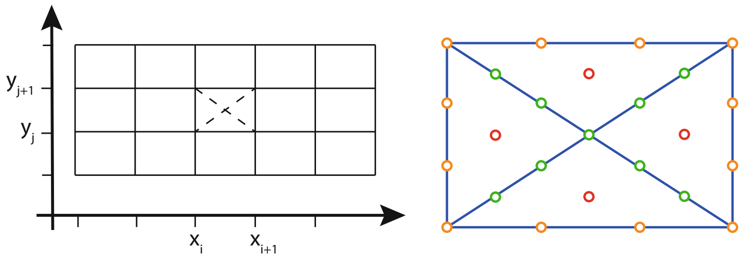


Fig. 2 *Left*: Rectangular function domain D with vertices (x_i, y_j). *Right*: Sibson split and Bézier coefficients of one rectangular patch

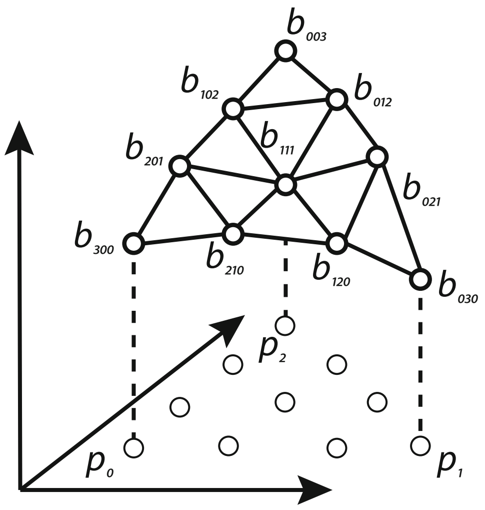


Fig. 3 Triangular Bézier surface patch of degree 3

functions interpolating these Hermite data is monotone, as illustrated in Fig. 1 for the 1D case.

However, instead of solving for functions with monotone increasing values along the grid lines, i.e. along the x- and y-axis as it was done in previous works, we construct our interpolating function to be monotone only in diagonal $(x + y)$-direction. Additionally, we require that the function has to be strictly monotone increasing, i.e. be free of critical points inside its domain. Note that the latter property is not guaranteed by the standard axis aligned monotonicity definition as used in [1, 2, 12].

We begin in Sect. 3.1 by first describing the standard Sibson split method, which subdivides each rectangle into four cubic bivariate polynomials joining with C^1 continuity. Then the modified Sibson split interpolant is presented. Section 3.2 introduces our relaxed monotonicity constraints and explains the difference to the standard monotonicity. Then our new monotonicity preserving interpolant is introduced in Sect. 3.3, and a proof of monotonicity is given. And finally, we derive two algorithms ensuring monotonicity and strict monotonicity respectively for a function interpolating monotone input data in Sects. 3.4 and 3.5.

3.1 Modified Sibson Split Interpolant

Cubic Sibson split patches (SSP) interpolate positional and partial derivative values given at the vertices of a rectangular gridded domain. Each patch is composed of four non-parametric cubic triangular Bézier patches with 25 Bézier ordinates

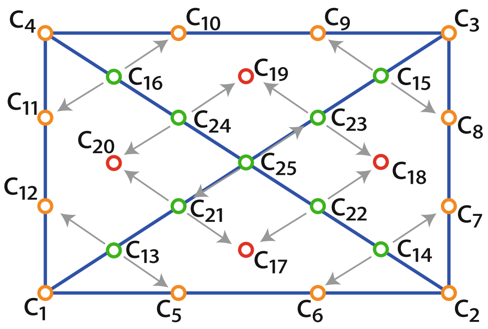


Fig. 4 Cubic (modified) Sibson interpolant. Input data: positional and gradient values located on a rectangular grid. Each grid cell is subdivided into four triangles and yields a C^1 piecewise cubic interpolant. *Orange points* are computed from given data, *red points* from linear cross-boundary derivatives and *green points* from C^1-continuity conditions

as illustrated in Fig. 4. The resulting surface is globally C^1-continuous since two adjacent SSP which share the same boundary curve along the common edge, interpolate the same partial cross-derivatives given at each end-point of the edge and have the same linear cross-boundary derivatives along the edge.

Following [8] the orange control vertices are used to interpolate positional and partial derivative values at the four corners. The green control vertices are constrained by the C^1 continuity between patches. Each green vertex is the mid-point on the straight line between the two vertices indicated by the arrows. The green middle vertex is the midpoint of the plane of its four surrounding vertices. These last conditions are sufficient to ensure C^1 continuity between the four patches, for an arbitrary choice of the red vertices [7]. The red vertices are chosen so that the *normal cross-boundary derivatives* (derivative orthogonal to the boundary) are linear. This leads to the following formulas for these four vertices:

$$\begin{cases} c_{17} = (2c_{13} + 2c_{14} + c_5 + c_6 - c_1 - c_2)\,/4 \\ c_{18} = (2c_{14} + 2c_{15} + c_7 + c_6 - c_1 - c_2)\,/4 \\ c_{19} = (2c_{15} + 2c_{16} + c_9 + c_6 - c_1 - c_2)\,/4 \\ c_{20} = (2c_{16} + 2c_{13} + c_{11} + c_6 - c_1 - c_2)\,/4. \end{cases} \tag{1}$$

The detailed formulas for all other coefficients are given in the Appendix.

Modified Sibson Interpolant

Instead of having linear normal cross-boundary derivatives, we require the *directional cross-boundary derivatives* $\frac{\partial f}{\partial(x+y)}(x, y)$ to be linear. This results in the following formulas for the Bézier ordinates marked in red in Fig. 4 and which replace the formulas (1) :

$$\begin{cases} c_{17} = (-c_1 + 2c_5 - c_6 + c_{13} + c_{14})/2 \\ c_{18} = (-c_3 + 2c_8 - c_7 + c_{14} + c_{15})/2 \\ c_{19} = (-c_3 + 2c_9 - c_{10} + c_{15} + c_{16})/2 \\ c_{20} = (-c_1 + 2c_{12} - c_{11} + c_{13} + c_{16})/2. \end{cases} \tag{2}$$

All other coefficients remain the same.

Lemma 1 *The modified Sibson Interpolant (2) is C^1 continuous for given position and gradient values at the corners of a rectangular domain.*

Proof Formulas (2) correspond to linear interpolation of $\frac{\partial f}{\partial(x+y)}(x, y)$ along the outer boundaries of each SSP. Since the SSPs share the same partial derivatives at their corners, the directional derivatives in the direction $x + y$ must be continuous across the domain D. Furthermore directional derivatives along the boundary curves of the SSPs are trivially continuous since the SSPs are C^0 continuous. It follows that the interpolant is C^1 across the domain. □

3.2 Relaxed Monotonicity

Monotonicity Constraints Let D be a rectangular domain in $\mathbb{R}^2$, subdivided into rectangles $D_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ with $1 \le i < n_x$ and $1 \le j < n_y$ and $h_i^x = x_{i+1} - x_i$, $h_j^y = y_{j+1} - y_j$.
The data set $\{(x_i, y_j, z_{ij})\}_{i,j=1}^{n_x,n_y}$ is called *diagonal monotone increasing* if

$$z_{ij} < z_{i+1,j+1} \tag{3}$$

for all $1 \le i < n_x$ and $1 \le j < n_y$.
The problem we seek to solve is to compute a C^1-continuous function $f : D \to \mathbb{R}$ such that f interpolates a diagonal monotone increasing data set, i.e.

$$f(x_i, y_j) = z_{ij}$$

for all $1 \le i \le n_x$ and $1 \le j \le n_y$, and f is *monotone increasing in* $(x + y)$-*direction*, i.e.

$$\frac{\partial f}{\partial(x + y)}(x, y) \ge 0 \tag{4}$$

for all (x, y) in D.

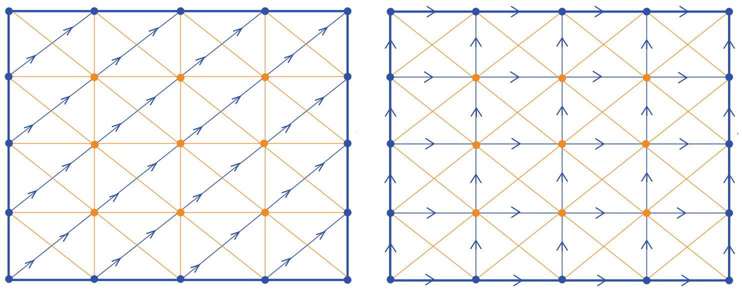


Fig. 5 *Left*: Diagonal monotone increasing data. The function values z_{ij} at the grid points are increasing along the grid diagonals (*blue arrows*). *Right*: Axis-aligned monotone increasing data. The functions values z_{ij} at the grid points are increasing along the x- and the y-axis (*blue arrows*)

Remark A function is called *strictly monotone increasing* if (4) is replaced by

$$\frac{\partial f}{\partial (x+y)}(x,y) > 0. \tag{5}$$

In contrast to the usually used axis-aligned monotonicity as in [12] our *diagonal monotonicity* (3) of the input data does not require that the given function values increase along rows and columns, but only along the grid diagonals, see Fig. 5. Note that this is a more general framework since if the data is increasing along the rows and columns then it is also increasing along diagonals. It is thus possible to deal with horizontal or vertical grid lines that are not monotone increasing.

3.3 *Sufficient Monotonicity Conditions*

We now introduce a new monotonicity preserving interpolation scheme for gridded data, that only assumes the input values to increase along diagonals, as illustrated in Fig. 5-left.

Let us assume without loss of generality that all D_{ij} are squares of the same size $h := h_i^x = h_j^y$. The next theorem gives sufficient conditions on the partial derivatives so that the modified SSP interpolant is (strictly) monotonic.

Theorem 1 *The modified SSP given by* (2) *which interpolates positional values* z_{kl} *and partial derivatives* z_{kl}^x, z_{kl}^y *on* D_{ij} *(*$k = i, i+1$, $l = j, j+1$*) satisfies* (5) *providing the following conditions hold:*

$$z_{ij}^x + z_{ij}^y > 0\,, \tag{6}$$

$$z_{i+1j}^x + z_{i+1j}^y > 0\,, \tag{7}$$

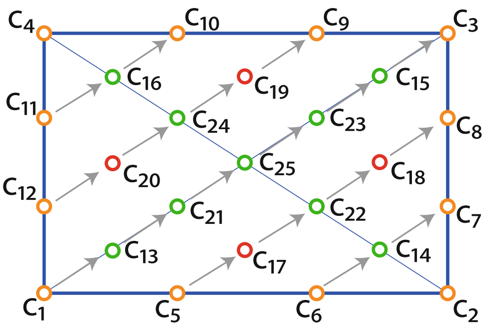


Fig. 6 If for every arrow the coefficient at its basis is strictly smaller than the coefficients at its tip then the four cubic triangular Bézier patches satisfy the diagonal monotonicity constraint (5), i.e. they are strictly increasing in the diagonal direction $x + y$

$$z^x_{ij+1} + z^y_{ij+1} > 0\ , \tag{8}$$

$$z^x_{i+1j+1} + z^y_{i+1j+1} > 0, \tag{9}$$

$$\frac{12}{h}(z_{i+1j+1} - z_{ij}) > 5z^x_{ij} + z^y_{ij} + 2z^x_{i+1j} + 2z^y_{i+1j} + z^x_{i+1j+1} + 5z^y_{i+1j+1}\ , \tag{10}$$

$$\frac{12}{h}(z_{i+1j+1} - z_{ij}) > z^x_{ij} + 5z^y_{ij} + 2z^x_{ij+1} + 2z^y_{ij+1} + 5z^x_{i+1j+1} + z^y_{i+1j+1}\ . \tag{11}$$

Proof In essence the proof of Theorem 1 follows from three arguments. The first is that conditions (6)–(9) imply the monotonicity constraint (5) at the four corners of the SSP. The second argument comes from the modified Sibson-Split interpolation formulas (2), which imply that (5) is fulfilled along the four outer boundary edges of the SSP. Eventually, conditions (10) and (11) enable to propagate the monotonicity constraint (5) inside the SSP.

From [7] we know that condition (5) is satisfied by the modified SSP provided that for all couples of coefficients ($c_j \to c_k$), as shown in Fig. 6, the relation $c_j < c_k$ holds. This leads to 18 sufficient conditions on the 25 coefficients c_i.

Hypotheses (6)–(9) imply $c_{13} - c_1 > 0,\ c_{14} - c_6 = c_7 - c_{14} > 0,\ c_3 - c_{15} > 0,\ c_{16} - c_{11} = c_{10} - c_{16} > 0$ (see the Appendix for the values of the control-points). Adding (6) and (7) leads to $2(c_{17} - c_5) = (c_{13} - c_1) + (c_{14} - c_6) > 0$. Analogously we have $c_8 - c_{18} > 0,\ c_9 - c_{19} > 0$, and $c_{20} - c_{12} > 0$. Furthermore $2(c_{21} - c_{13}) = (c_{20} - c_{12}) + (c_{17} - c_5) > 0$, and analogously we have $c_{15} - c_{23} > 0$. $c_{22} - c_{17}$ can be computed from the interpolation conditions using (2) and the control-point formulas given in the Appendix. This leads to

$$c_{22} - c_{17} = \frac{1}{2}\Big[(z_{i+1j+1} - z_{ij}) - \frac{h}{12}(5z^x_{ij} + z^y_{ij} + 2z^x_{i+1j}$$
$$+2z^y_{i+1j} + z^x_{i+1j+1} + 5z^y_{i+1j+1})\Big]$$

Therefore condition (10) implies $c_{22} - c_{17} > 0$. Analogously condition (11) implies $c_{24} - c_{20} > 0$. From the Appendix we have $2 * (c_{25} - c_{21}) = (c_{22} - c_{17}) + (c_{24} - c_{20})$ and therefore $c_{25} - c_{21} > 0$. The remaining three conditions follow from the Appendix: $c_{18} - c_{22} = c_{22} - c_{17} > 0$, $c_{23} - c_{25} = c_{25} - c_{21} > 0$, $c_{19} - c_{24} = c_{24} - c_{20} > 0$. $\square$

Remark (Theorem 1') Replacing the strict inequality signs by $\geq$ in (6)–(11) yields an analogous theorem for the weaker monotonicity constraint (4). The proof is analogous as well.

3.4 Algorithm 1

Assume now we are given as input gridded values z_{ij} which satisfy (3). Our goal is to compute partial derivatives z_{ij}^x and z_{ij}^y such that the sufficient conditions given in Theorem 1 are satisfied. Let us call them *admissible* partial derivatives. Using these admissible partial derivatives together with the input values z_{ij} for the modified SSP interpolant makes the resulting function being monotonic.

We present two algorithms for computing admissible partial derivatives, one for the weaker Theorem 1' and one for Theorem 1. Also the strategies followed by the algorithms are different. While the first algorithm modifies some given estimations of the partial derivatives, the second algorithm computes admissible partial derivatives only from the input function values.

The first part of Algorithm 1 corresponds to the conditions (6)–(9) of Theorem 1. If the sum of the partial derivatives $z_{kl}^x + z_{kl}^y$ is negative in any corner of the SSP, we apply an orthogonal projection as illustrated in Fig. 7. This procedure results in a gradient that is closest to the original gradient. The modified partial derivatives are thus given by $((z_{kl}^x - z_{kl}^y)/2, (z_{kl}^y - z_{kl}^x)/2)$.

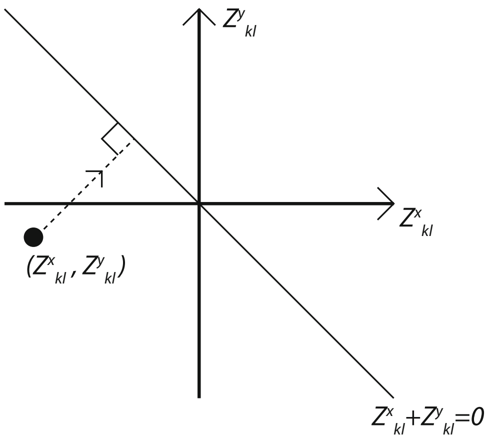


Fig. 7 Projection of the partial derivatives (z_{kl}^x, z_{kl}^y) on $z_{kl}^x + z_{kl}^y = 0$ when their sum is negative.

Algorithm 1 Modify the input partial derivatives for one SSP in order to satisfy the sufficient monotonicity conditions given by Theorem 1

```
Require: $z_{kl}$ with $z_{ij} < z_{i+1j+1}$, and partial derivatives $z^x_{kl}, z^y_{kl}$ $(k = i, i+1, l = j, j+1)$
  {First part of the algorithm}
  for $k \in \{i, i+1\}$ do
    for $l \in \{j, j+1\}$ do
      {Verification of the four conditions (6)-(9)}
      if $z^x_{kl} + z^y_{kl} < 0$ then
        $old_x \leftarrow z^x_{kl}$
        $old_y \leftarrow z^y_{kl}$
        {Orthogonal projection of the point $(z^x_{kl}, z^y_{kl})$ onto $z^x_{kl} + z^y_{kl} = 0$}
        $z^x_{kl} \leftarrow \frac{old_x - old_y}{2}$
        $z^y_{kl} \leftarrow \frac{old_y - old_x}{2}$
      end if
    end for
  end for
  {Second part of the algorithm}
  $\Delta z \leftarrow \frac{12}{h}(z_{i+1j+1} - z_{ij})$
  $S_2 \leftarrow 5|z^x_{ij}| + |z^y_{ij}| + 2|z^x_{i+1j}| + 2|z^y_{i+1j}| + |z^x_{i+1j+1}| + 5|z^y_{i+1j+1}|$
  $S_3 \leftarrow |z^x_{ij}| + 5|z^y_{ij}| + 2|z^x_{ij+1}| + 2|z^y_{ij+1}| + 5|z^x_{i+1j+1}| + |z^y_{i+1j+1}|$
  $S_{max} \leftarrow Maximum(S_2, S_3)$
  {Verification of the last two conditions (10),(11)}
  if $\Delta z < S_{max}$ then
    $c \leftarrow \frac{\Delta z}{S_{max}}$
    $z^x_{kl} \leftarrow c\, z^x_{kl}$ for $k = i, i+1, l = j, j+1$
    $z^y_{kl} \leftarrow c\, z^y_{kl}$ for $k = i, i+1, l = j, j+1$
  end if
```

The second part of Algorithm 1 decreases the partial derivative values so that conditions (10) and (11) of the Theorem 1 are satisfied. Absolute values are required since the partial derivatives as input may have a negative value.

3.5 *Algorithm 2*

The previous algorithm needs estimated gradient values as input. We now propose a second algorithm which directly computes admissable gradient values for the strict monotonicity of Theorem 1.

This second algorithm ensures that the right hand sides in conditions (10) and (11) are always smaller than $\lambda \frac{12}{h}\left(z_{i+1j+1} - z_{ij}\right)$, where $\lambda \in]0, 1[$ is a user-specified constant. It also ensures that all partial derivatives are positive, so that conditions (6) to (9) are satisfied as well.

Proof (Algorithm 2) Because $z_{i+1j+1} > z_{ij}$, all K_{ij} are positive. Therefore all $KMin_{ij}$ are positive, and all partial derivatives z^x_{ij} and z^y_{ij} computed by Algorithm 2 are also positive. It follows that conditions (6)–(9) are fulfilled. It remains to prove conditions (10) and (11).

Let S be the right part of the condition (10):

$$\begin{aligned} S &= 5z^x_{ij} + z^y_{ij} + 2z^x_{i+1j} + 2z^y_{i+1j} + z^x_{i+1j+1} + 5z^y_{i+1j+1}, \\ &= 6\frac{\lambda}{2}KMin_{ij} + 4\frac{\lambda}{2}KMin_{i+1j} + 6\frac{\lambda}{2}KMin_{i+1j+1}. \end{aligned}$$

Algorithm 2 Compute partial derivatives for all SSPs in order to fulfill the sufficient monotonicity conditions given by Theorem 1

```
Require: $z_{ij}$ for $1 \le i \le n_x$ and $1 \le j \le n_y$, such that $z_{ij} < z_{i+1j+1}$
Require: a constant $\lambda$ with $0 < \lambda < 1$
  for $i = 1 \cdots n_x - 1$ do
    for $j = 1 \cdots n_y - 1$ do
      $K_{ij} \leftarrow \frac{3}{2h}\left(z_{i+1j+1} - z_{ij}\right)$
    end for
  end for
  $KMin_{11} \leftarrow K_{11}$
  $KMin_{n_x n_y} \leftarrow K_{n_x-1,n_y-1}$
  $KMin_{n_x,1} \leftarrow K_{n_x-1,1}$
  $KMin_{1,n_y} \leftarrow K_{1,n_y-1}$
  {traverse the interior of domain}
  for $i = 1 \cdots n_x - 2$ do
    for $j = 1 \cdots n_y - 2$ do
      $KMin_{i+1,j+1} \leftarrow min\left(K_{ij}, K_{i+1j}, K_{ij+1}, K_{i+1j+1}\right)$
    end for
  end for
  {traverse left and right domain boundaries}
  for $j = 1 \cdots n_y - 2$ do
    $KMin_{1,j+1} \leftarrow min\left(K_{1j}, K_{1,j+1}\right)$
    $KMin_{n_x,j+1} \leftarrow min\left(K_{n_x-1j}, K_{n_x-1,j+1}\right)$
  end for
  {traverse bottom and top domain boundaries}
  for $i = 1 \cdots n_x - 2$ do
    $KMin_{i+1,1} \leftarrow min\left(K_{i1}, K_{i+1,1}\right)$
    $KMin_{i+1,n_y} \leftarrow min\left(K_{i,n_y-1}, K_{i+1,n_y-1}\right)$
  end for
  {initialize partial derivatives from KMin array}
  for $i = 1 \cdots n_x$ do
    for $j = 1 \cdots n_y$ do
      $z^x_{ij} \leftarrow \lambda\frac{1}{2}KMin_{ij}$
      $z^y_{ij} \leftarrow z^x_{ij}$
    end for
  end for
```

Algorithm (2) computes the coefficients $KMin_{ij}$ such that:

$$KMin_{ij},\ KMin_{i+1j},\ KMin_{i+1j+1} \leq \frac{3}{2h}(z_{i+1j+1} - z_{ij}).$$

And therefore:

$$S \leq \lambda \frac{12}{h}(z_{i+1j+1} - z_{ij}),$$

which means that condition (10) is verified.

Condition (11) can be proven analogously. □

The parameter λ can be used to control the shape of the interpolant. A value of λ close to 1 tends to increase the partial derivatives at the corners of the SSP, and to decrease the partial derivatives in the interior of the SSP. On the other hand, a value of λ close to 0 implies almost 0 partial derivatives at the corners of the SSP. In our examples we have used $\lambda = \frac{2}{3}$.

4 Results

The first and simple example shown in Fig. 8 corresponds to a 4×2 grid of scalar values that satisfy the relaxed monotonicity constraints (3). The interpolated values are increasing along the rows, but not along all columns. They are diagonal monotone increasing but not axis-aligned monotone. This simple data set can thus not be handled by previous works on monotonicity preserving interpolation. In contrast, our relaxed monotonicity constraints enable to built an interpolant that

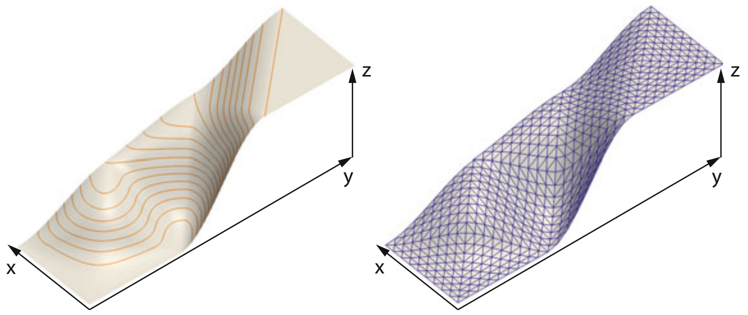


Fig. 8 A 4×2 grid of scalar values is interpolated by our C^1 cubic interpolant. This simple example can not be handled by prior works on monotonicity preserving interpolation since some of the rows have increasing and other have decreasing values. Nevertheless our interpolant does not exhibit critical points as it can be seen from the isolines in the *left*. The isoparametric lines of the 12 cubic Bézier patches are shown on the *right*

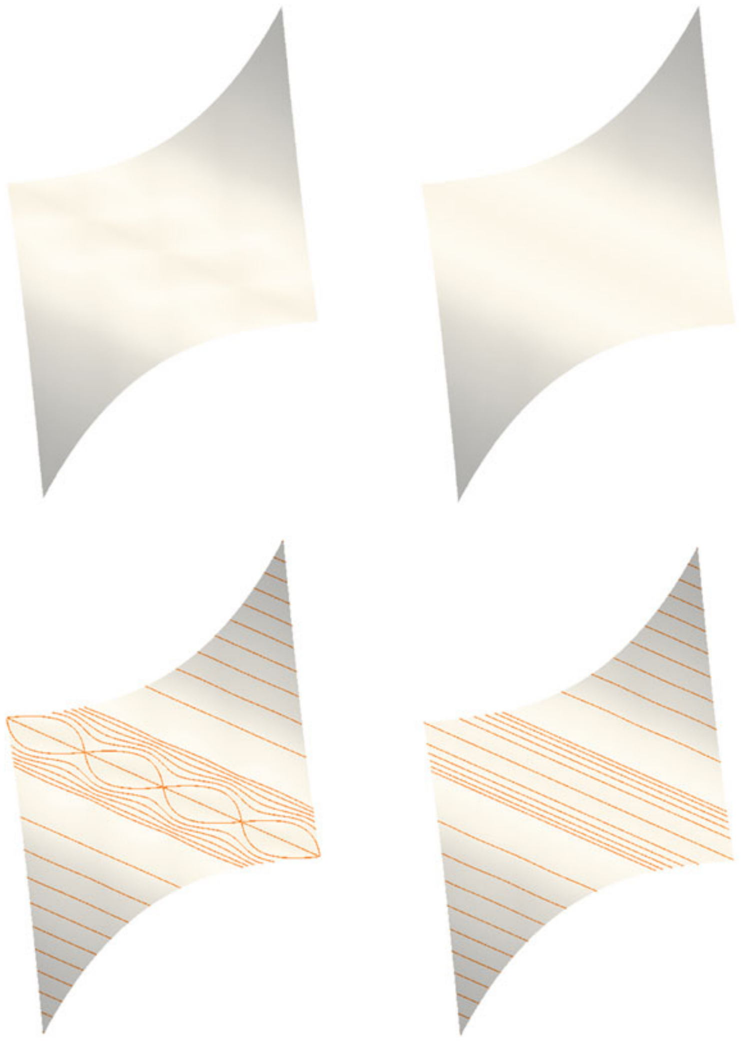


Fig. 9 Interpolation of a grid sampled from an analytical function. The *left part* shows the result when the exact partial derivatives of the function are used for the interpolant. The isolines on the *bottom left* clearly indicate critical points. In the *right part* these critical points are removed by modifying the partial derivatives with our Algorithm 1 (Sect. 3.4)

is free of critical points as it can be seen from the isolines shown in the left side of Fig. 8. On the right side, the same interpolant is shown together with the isoparametric lines. Since the interpolant is composed of triangular Bézier patches, there are three isoparametric directions.

The second example in Fig. 9 illustrates the ability to use estimates of gradient values thanks to Algorithm 1 presented in Sect. 3.4. In this example a grid of

5×5 sampled values in the domain $[-0.4, +0.4]^2$ for the function $(x + y)^3$ is interpolated. We give as input to Algorithm 1 the exact partial derivatives sampled from the function. We show our modified Sibson-Split interpolant on the left side of Fig. 9 without correction of the partial derivatives, and on the right side with the corrected partial derivatives computed by Algorithm 1. As it can be seen from the isolines at the bottom of Fig. 9, the interpolant with the exact partial derivatives exhibits critical points, which are removed when the corrected partial derivatives computed by Algorithm 1 are used for the interpolant.

The third example shows the interpolation of a grid of size 10×10. The values at the grid vertices are computed randomly but satisfy the diagonal monotonicity constraint (3). For this example Algorithm 2, described in Sect. 3.5, is applied to compute admissible values for the partial derivatives at the grid vertices. This data set can not be handled by previous works since neither the rows nor the columns are monotone increasing. Nevertheless, our new method is able to produce a monotone C^1 interpolant free of critical points as it can be seen from the isolines in Fig. 10. The inset on the top right shows a closeup view with the isoparametric lines of the Bézier patches.

The last example in Fig. 11 illustrates the ability of *piecewise* monotone interpolants to interpolate a grid of function values, where local minima, maxima and saddles are prescribed. In contrast to the previous settings, where a globally monotone function performs shape preserving interpolation, we now apply the

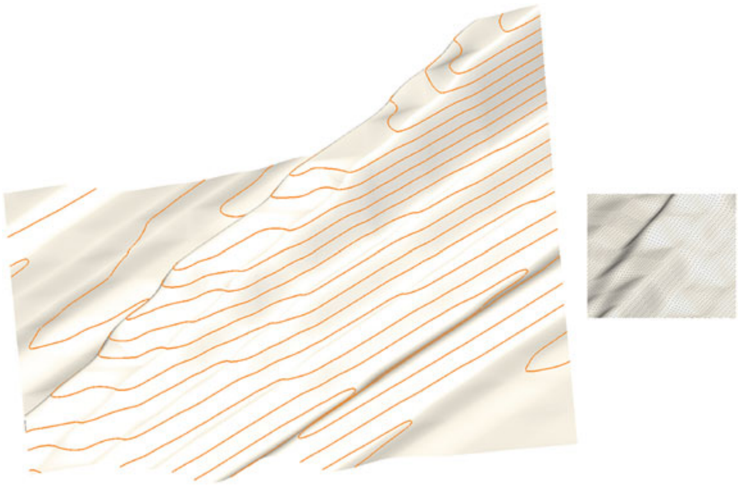


Fig. 10 Interpolation of a grid of 10×10 function values. The grid values are randomly chosen such that our relaxed monotonicity constraint (3) is fulfilled. Since neither the rows nor the columns are increasing, this grid can not be handled by previous works on constraint monotonic interpolation. Our new method can produce an interpolant free of critical points in the interior of the definition domain, as shown by the isolines. The partial derivatives are computed from Algorithm 2, described in Sect. 3.5. The *inset* on the *top right* shows a closeup view with the isoparametric lines

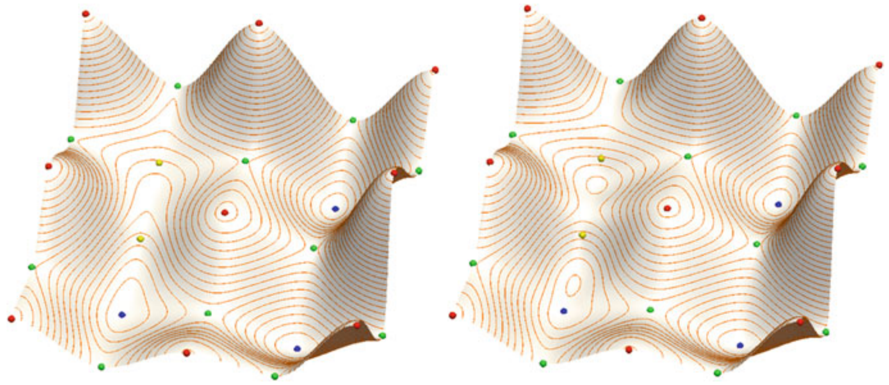


Fig. 11 Local maxima (*red*), minima (*blue*), saddles (*green*) and regular (*yellow*) vertices are interpolated by a C^1 piecewise cubic interpolant. *Left*: no unwanted local extrema exist in the interior of the cubic patches. *Right*: partial derivatives too large in size are chosen for the *yellow* regular vertices implying that additional unwanted local extrema appear inside the cubic polynomial patches

monotonicity property only locally. We compute a surface which is piecewise monotone (inside each grid cell) without generating any extraneous critical points except at the grid vertices where critical points are prescribed. The red vertices are local maxima, the blue vertices are local minima and the green vertices are saddles. At all these prescribed critical points the partial derivatives are fixed to be 0. There are only two grid vertices in yellow, which are regular vertices and where the choice of the partial derivatives is determinant for the monotonicity of the neighbouring patches. Indeed, this example also allows to illustrate the influence of the gradient values on the shape of a function computed by a Sibson interpolant. Figure 11 shows two results with different gradient values at the yellow vertices, reproducing thus the phenomena described in Fig. 1-right for the 2D case. In fact, on can observe that the choice of gradient values too large in size produces extra unwanted critical points, see Fig. 11-right, whereas Fig. 11-left shows the resulting interpolant with properly chosen partial derivatives the yellow vertices. In both surfaces the local extrema are exactly interpolated, and the shape of the interpolant is smooth around the yellow regular vertices. All the patches are individually monotone increasing and join with C^1-continuity.

Finally, let us provide some statistics: All examples have been computed in less then 1 ms. Even though we only show small size examples to appreciate geometric and topological properties of the resulting functions, the method runs in real time also for very large data sets. We generated data sets with 10^6 grid cells and computed valid gradient values in 3 ms. Moreover, the computation of Bézier surfaces can be performed in parallel. Indeed, once the derivatives known at the corners of the grid cell, the modified Sibson interpolant can be computed independently from its neighbours.

Conclusion

In this paper we propose a new method to interpolate a 2D grid of scalar values by a C^1 piecewise cubic function with no critical points and no local extrema in the interior of the domain. In comparison with prior related works, we do not require the values to increase along all rows and columns of the grid. Instead, we introduce a relaxed monotonicity constraint in which the values are required to increase monotonously only along diagonals of the grid cells. We also introduce a modified Sibson-split interpolant which is coherent with this relaxed monotonicity constraint. We give sufficient conditions on the partial derivatives at the grid vertices such that the Sibson-split interpolant is free of local extrema. And we propose two algorithms to actually compute admissible partial derivatives satisfying these sufficient conditions. The first algorithm takes as input estimated values of the partial derivatives and modifies them in order to ensure monotonicity. The second algorithm computes partial derivatives without requiring any initial guess. As shown in Sect. 4, such a C^1 piecewise cubic monotonic interpolant can also be used to interpolate a grid of prescribed local minima, maxima and saddles.

Our method is a step towards reconstructing function from MS complexes, even though it can't be applied directly in its present form. Indeed, the current work is limited to regular grids. Therefore, we are currently investigating the generalization of our results to monotone interpolation of function values defined on a triangular mesh instead of a grid. This would enable us to extend the example shown in Fig. 11 to an arbitrary setting of local extrema. Furthermore, we plan to apply our method to the reconstruction of monotonic functions within MS cells. Another direction of future research is to extend the present approach to C^2 continuity using quintic triangular Bézier patches.

Appendix—Sibson Split Interpolant

Let D be a rectangular domain in $\mathbb{R}^2$, regularly subdivided into rectangles $D_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$, $1 \leq i < n_x$, $1 \leq j < n_y$ and the following ordinates z_{ij} and gradients z^x_{ij}, z^y_{ij} given at the grid points. Let $h^x = x_{i+1} - x_i$, $h^y = y_{j+1} - y_j$. Each rectangle is splitted into four sub-triangles by drawing both diagonals.

The Sibson split (cf. [7]) is a cubic C^1-continuous function $f : D \rightarrow \mathbb{R}$ interpolating the input data with

$$f(x_i, y_i) = z_{ij}, \qquad f_x(x_i, y_i) = z^x_{ij}, \qquad f_y(x_i, y_i) = z^y_{ij},$$

where each patch defined on D_{ij} is composed of four cubic polynomials with in total 25 Bézier coefficients (see Fig. 4), computed uniquely as follows:

$$
\begin{array}{lll}
c_1 = z_{ij} & c_2 = z_{i+1,j} & c_3 = z_{i+1,j+1} \\
c_4 = z_{i,j+1} & c_5 = c_1 + \frac{h^x}{3} z^x_{ij} & c_6 = c_2 - \frac{h^x}{3} z^x_{i+1,j} \\
c_9 = c_3 - \frac{h^x}{3} z^x_{i+1,j+1} & c_{10} = c_4 + \frac{h^x}{3} z^x_{i,j+1} & c_{12} = c_1 + \frac{h^y}{3} z^y_{ij} \\
c_7 = c_2 + \frac{h^y}{3} z^y_{i+1,j} & c_8 = c_3 - \frac{h^y}{3} z^y_{i+1,j+1} & c_{11} = c_4 - \frac{h^y}{3} z^x_{i,j+1} \\
c_{13} = \frac{1}{2}(c_5 + c_{12}) & c_{14} = \frac{1}{2}(c_6 + c_7) & c_{15} = \frac{1}{2}(c_8 + c_9) \\
c_{16} = \frac{1}{2}(c_{11} + c_{10}) & c_{21} = \frac{1}{2}(c_{20} + c_{17}) & c_{22} = \frac{1}{2}(c_{17} + c_{18}) \\
c_{23} = \frac{1}{2}(c_{18} + c_{19}) & c_{24} = \frac{1}{2}(c_{19} + c_{20}) & c_{25} = \frac{1}{2}(c_{21} + c_{23}) = \frac{1}{2}(c_{22} + c_{24}).
\end{array}
$$

References

1. Beatson, R.K.T., Ziegler, Z.: Monotonicity preserving surface interpolation. SIAM J. Numer. Anal. **22**(2), 401–411 (1985)
2. Carlson, R.E., Fritsch, F.N.: An algorithm for monotone piecewise bicubic interpolation. SIAM J. Numer. Anal. **26**(1), 230–238 (1989)
3. Carnicer, J.M., Floater, M.S., Peña, J.M.: Linear convexity conditions for rectangular and triangular Bernstein-Bézier surfaces. Comput. Aided Geom. Des. **15**(1), 27–38 (1997)
4. Cavaretta Jr., A.S., Sharma, A.: Variation diminishing properties and convexity for the tensor product bernstein operator. In: Yadav, B.S., Singh, D. (eds.) Functional Analysis and Operator Theory. Lecture Notes in Mathematics, vol. 1511, pp. 18–32. Springer, Berlin/Heidelberg (1992)
5. Delgado, J., Peòa, J.M.: Are rational Bézier surfaces monotonicity preserving? Comput. Aided Geom. Des. **24**(5), 303–306 (2007)
6. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical Morse–Smale complexes for piecewise linear 2-manifolds. Discrete Comput. Geom. **30**(1), 87–107 (2003)
7. Farin, G.E.: Triangular Bernstein-Bézier patches. Comput. Aided Geom. Des. **3**(2), 83–127 (1986)
8. Farin, G.E.: Curves and Surfaces for Computer-Aided Geometric Design: A Practical Code, 4th edn. Academic, Orlando (1996)
9. Floater, M.S., Peña, J.M.: Tensor-product monotonicity preservation. Adv. Comput. Math. **9**(3–4), 353–362 (1998)
10. Floater, M.S., Peña, J.M.: Monotonicity preservation on triangles. Math. Comput. **69**(232), 1505–1519 (2000)
11. Floater, M., Beccari, C., Cashman, T., Romani, L.: A smoothness criterion for monotonicity-preserving subdivision. Adv. Comput. Math. **39**(1), 193–204 (2013)

12. Han, L., Schumaker, L.L.: Fitting monotone surfaces to scattered data using c1 piecewise cubics. SIAM J. Numer. Anal. **34**(2), 569–585 (1997)
13. Jüttler, B.: Surface fitting using convex tensor-product splines. J. Comput. Appl. Math. **84**(1), 23–44 (1997)
14. Kuijt, F., van Damme, R.: Monotonicity preserving interpolatory subdivision schemes. J. Comput. Appl. Math. **101**(1–2), 203–229 (1999)
15. Mainar, E., Peña, J.M.: Monotonicity preserving representations of non-polynomial surfaces. J. Comput. Appl. Math. **233**(9), 2161–2169 (2010)
16. McAllister, D.F., Passow, E., Roulier, J.A.: Algorithms for computing shape preserving spline interpolations to data. Math. Comput. **31**(139), 717–725 (1977)
17. Smale, S.: On gradient dynamical systems. Ann. Math. **74**(1), 199–206 (1961)
18. Strom, K.: On convolutions of b-splines. J. Comput. Appl. Math. **55**(1), 1–29 (1994)
19. Willemans, K., Dierckx, P.: Smoothing scattered data with a monotone powell-sabin spline surface. Numer. Algorithms **12**(1), 215–231 (1996)

Shape Analysis and Description Using Real Functions

Silvia Biasotti, Andrea Cerri, Michela Spagnuolo, and Bianca Falcidieno

1 Introduction

Since many years researchers work on different aspects of shape-based modelling, where by shape we mean a one-, two- or higher-dimensional connected, compact space having a visual appearance. In this context, geometric modelling techniques are at a quite mature stage in the sense that the most fundamental problems related to the modelling, manipulation and visualization of static and dynamic 3D shapes are well understood and solved. However, several issues are still open and nowadays even more critical due to the larger use of data in digital form in a variety of applied and scientific domains. Among these, in the mid-1990s the shape similarity problem (and related issues about how to code uncertainty due to incompleteness or noise affecting original data [1]) started to be approached on the basis of the geometric, structural and semantic properties of the digital shapes [16].

3D Shape Analysis studies how to derive shape information from objects stored in the form of geometric models, extracting from low-level properties (points, triangles, voxels) high-level knowledge about the geometry, structure and semantic meaning of 3D objects. It deals, for example, with evaluating the similarity among 3D objects and the semantic annotation of objects and their parts; the best-known representative of this class of activities is probably the classification of shapes (or taxonomy), which takes place in a variety of contexts: botanics, mathematics, and zoology to cite just a few. The idea underlying classification is to group objects, which exhibit similar characteristics, even if they may differ in some quantitative parameter. Classification requires tools, which permit the identification of these *similar characteristics* independently of their specific embedding in the object.

S. Biasotti • A. Cerri • M. Spagnuolo • B. Falcidieno (✉)
CNR-IMATI, Via De Marini 6, Genova, Italy
e-mail: silvia.biasotti@ge.imati.cnr.it; andrea.cerri@ge.imati.cnr.it; michela.spagnuolo@ge.imati.cnr.it; bianca.falcidieno@ge.imati.cnr.it; bianca.falcidieno@ge.imati.cnr.it

Though young as a discipline on its own, 3D Shape Analysis shows solid mathematical foundations. As anticipated by D'Arcy Thompson in his seminal work, mathematical formalism and tools are needed in shape analysis and classification for their abstraction and synthesis power: *"we must learn from the mathematician to eliminate and discard; to keep in mind the type and leave the single case, with all its accidents, alone; and to find in this sacrifice of what matters little and conservation of what matters much one of the peculiar excellences of the method of mathematics"* [26].

Though until recently regarded as a niche of research, 3D Shape Analysis is just now maturing to a key discipline: it is getting many people involved, and is penetrating many applied domains, spanning from Entertainment to Life Sciences, from Cultural Heritage to Industrial Design, from Manufacturing to Urban Planning. In all these contexts, mathematics provides shape descriptors, which can be successfully applied to shape analysis. Special emphasis is given to issues related to the definition of abstraction tools for deriving high-level descriptions of complex shape models [5, 17, 22, 23].

Our efforts have been focusing on computational topology that is a branch of computational mathematics aimed at the solution of problems related to topological issues, without disregarding the feasibility or the computational complexity of the problem.

Here, we present a shape analysis and comparison pipeline based on a collection of functions defined of an object (Sect. 2). These functions are supposed to represent meaningful shape properties and drive the extraction of a geometric/topological shape descriptor. In this contribution, we focus on Reeb graphs and persistence diagrams (Sects. 2.1.1 and 2.1.2). In order to use sets of functions for capturing different shape aspects, we describe a clustering approach that minimizes the redundancy in the final shape description (Sect. 2.1.3). We also take advantage of persistence spaces, which can be used as shape descriptors in the case of vector-valued functions (Sect. 2.2.1). We finally show the proposed pipeline in action, dealing with the retrieval of textured 3D shapes (Sect. 3).

2 Computational Topology Methods for Shape Description

Characterizing a shape means building a computational description (a *shape descriptor*) that preserves the most representative features of the shape, usually a few basic types, along with their relationships and their invariants [16]. In this context, Mathematics plays a key role to handle with the complexity of digital shapes.

2.1 Shape Descriptions Based on Real-Valued Functions

Differential topology, and specifically Morse theory, provides a suitable setting to formalize several problems related to shape analysis and description, and concretely approach them through the algorithmic resources offered by computational topology. In this scenario, methods have been derived to analyze the shape of an object according to the properties of real functions defined on it. The added value of these approaches is that different functions can be used according to the properties and invariants that one wishes to capture, thus providing a flexible shape description framework. Examples of popular functions used in object retrieval are distance functions, geodesic-based functions, Laplace eigenfunctions and distribution maps [6, 7]. The resulting descriptors compactly store the geometric/topological information related to the shape; here we focus on Reeb graphs and persistence diagrams.

2.1.1 Reeb Graphs

The notion of a Reeb graph dates back to 1946, when George Reeb [22] defined the quotient space of a smooth manifold S with respect to a simple, Morse function $f : S \to \mathbb{R}$ in terms of equivalence relation "$\sim$" that identifies the points belonging to the same connected component of level sets of f:

$(p, f(p)) \sim (q, f(q))$ if and only if $f(p) = f(q)$ and p, q are in the same connected component of $f^{-1}(f(p))$.

The quotient space defined by Reeb is what is currently called Reeb graph and stores the evolution and the arrangement of the level sets of a real function defined over the shape. It is able to convey both geometrical and topological information, since the topological analysis is driven by the properties expressed by f.

Since its first application in Computer Graphics by Shinagawa and Kunii [23] in 1991, Reeb graphs have been used to solve different problems related to shape matching, morphing and coding. To deal with shape matching, it is necessary to enrich the pure topological Reeb graph with local information, i.e. geometric attributes that represent the characteristics of the shape parts that correspond to nodes and edges of the graph. We adopt here the *Extended Reeb Graph (ERG)* description, which has been first defined in [5]. The ERG represents a surface through a finite set of level sets of f, and can be used to describe triangle meshes representing surfaces, with or without boundary [4].

If we consider the interval $[f_m, f_M]$ containing the image of the continuous function $f : S \to \mathbb{R}$ and a set of real values $\{f_i : f_i \in [f_m, f_M]\}$, an *extended Reeb* equivalence between two points $p, q \in S$ is defined by:

1. $f(p)$ and $f(q)$ belong to the same element of $t \in I$;
2. p and q belong to the same connected component of $f^{-1}(t)$, $t \in I$;

where I is the partition of $[f_m, f_M]$ induced by the values $\{f_i\}$. It follows that all points satisfying the above conditions 1. and 2. for an element $t \in I$ are Reeb-equivalent in the extended sense and they may therefore collapse into the same point of the quotient space. The quotient space obtained from such a relation is a discrete space, called *Extended Reeb (ER)* quotient space [4]. The *ER* quotient space may be represented as a traditional graph named the *Extended Reeb Graph (ERG)*, by following the adjacency relations among the elements of $f^{-1}(t)$, $t \in I$.

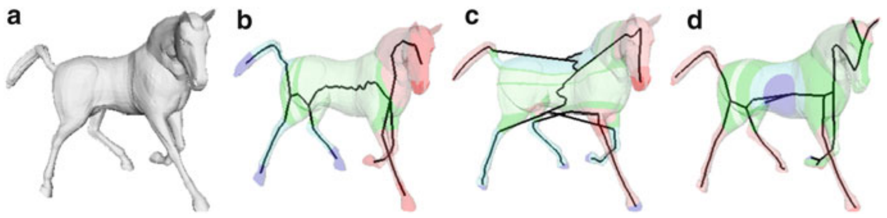


Fig. 1 A model (**a**) and the corresponding ERGs with respect to two eigenfunctions of the Laplace-Beltrami operator (**b** and **c**) and the distance from the center of mass (**d**)

From the practical point of view, we start the extraction of the ERG by considering a uniform sampling of $[f_m, f_M]$, then we possibly refine the level set using the adaptive, bisection criterion proposed in [4]. Beside the combinatorial representation of the graph, the ERG is embedded in $\mathbb{R}^3$ by associating each node with the position of the barycentre of the corresponding part and visualized almost everywhere as an approximation of the centerline skeleton of S, see Fig. 1b–d. This picture shows how different functions can give insights on the shape from various perspectives.

2.1.2 Persistence Diagrams

Persistence diagrams are one of the main tools introduced in the context of *persistence* [14], and can be used to globally describe the shape of an object according to a topological perspective.

The main idea is to explore the evolution of the lower level sets of a function $f : X \to \mathbb{R}$ defined on a topological space X. The function f measures properties which are considered relevant for the analysis of the shape; the role of X is to represent the object under study, and can vary from an iconic view of the model to its surface or volume.

Focusing on the occurrence of important topological events, it is possible to obtain a global shape description. We refer, e.g., to the birth and the death of connected components, tunnels and voids, which are encoded in a *0th-*, *1st-* and *2nd-persistence diagram*, respectively. For example, the 0th persistence diagram of $f : X \to \mathbb{R}$ keeps track of the connected components which remain disconnected passing from a lower level set X_u of X to another lower level set X_v with $u < v$, being a lower level set for f defined as $X_u = \{x \in X : f(x) \leq u\}$, for $u \in \mathbb{R}$.

Figure 2 shows the 0th persistence diagrams for the “height” function defined on two surface models. As displayed in the picture, a persistence diagram can be seen as a collection of points lying in the half-plane $\{(u, v) \in \mathbb{R}^2 : u < v\}$, (red dots in Fig. 2b, c). Each point (u, v) in the diagram describes the *lifespan* of a connected component: the u-coordinate denotes the value of f at which a new component appears in the associated lower level set X_u; similarly, the v-coordinate denotes the value of f at which that component merges with an existing one. The distance from

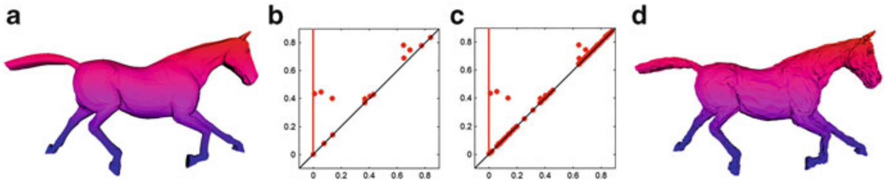


Fig. 2 A model (**a**) and a noisy version of it (**d**), both associated with the height function (color coded from *blue* for low values to *red* for high values), and the corresponding 0th persistence diagrams (**b** and **c**)

the diagonal $u = v$ represents the component lifespan, which in turn mirrors the importance of the topological feature that component represents: points far from the diagonal describe important, long-lived features, whereas points close to the diagonal describe local information such as smaller details and noise. Consider for instance the three red points in Fig 2b which are farthest from the diagonal: Together with the red line, they mirror the existence of four meaningful features: the legs of the horse (Fig 2a) born at the four minima of the height function. In particular, the red vertical line can be interpreted as a point at infinity, and represents a connected component that will never die, i.e. its u-component corresponds to the smallest value for the height function, and its v-component is equal to $+\infty$.

Persistence diagrams are stable under the so-called *bottleneck distance*: small changes in the function f induces only small changes in the corresponding diagrams, with respect to the aforementioned distance [13]. An intuition of this fact is given by Fig. 2b, c: a slight perturbation in the original data results in a minor movement for points of the persistence diagram which are far from the diagonal. In other words, relevant topological features do not change that much. In particular, the stability property allows for replacing the comparison of two shapes with the one of corresponding persistence diagrams.

2.1.3 Selection of Representative Functions

Although the generalization to the case of multi-variate functions allows us to capture properties of shapes that are intrinsically multidimensional (see discussions in Sect. 2.2), here we depict the more general case in which sets of scalar functions, not necessarily related each other, concur to provide shape's information under different viewpoints.

Selecting the (hopefully small) number of functions that better describe the shape of an object is an open problem. Being able to reduce the number of functions would help to avoid redundancy in the shape description and to associate to the object a small set of descriptors, e.g. Reeb graphs. Moreover, such a reduction must preserve as much as possible the descriptive power of the descriptors.

In this scenario, we present the clustering approach proposed in [9] to select a subset $F' \subseteq F$ of scalar functions that qualitatively define a sort of basis of the shape properties. The approach groups the functions in a completely unsupervised manner according to a distance defined on F. The results obtained in [9] show that for a given class of shapes it is possible to identify a (small) number of functions that are mutually independent.

For a set $F = \{f_1, \ldots, f_n\}$ of n functions defined on a triangle mesh T representing a shape, the distance $\mathscr{I}(f_i, f_j)$ between two functions f_i and f_j is given by:

$$\mathscr{I}(f_i, f_j) := \frac{1}{area(T)} \sum_{t \in T} \left| < \frac{\nabla^t f_i}{\|\nabla^t f_i\|}, \frac{\nabla^t f_j}{\|\nabla^t f_j\|} > \right|,$$

with $\nabla^t f_i$, $\nabla^t f_j$ representing the gradient of f_i and f_j over the triangle $t \in T$. Intuitively, we are assuming that a relevant discrepancy in the distribution of the gradients (and therefore of the level sets) of two functions f_i and f_j implies that they are significantly different. From the practical point of view, we derive the function distances directly as a weighted sum over the triangles of the dot products of the function gradients. In this way, gradients are well defined and the distance evaluation is linear in the number of triangles of T. Then, a matrix M with entries $M_{ij} := 1 - \mathscr{I}(f_i, f_j)$ is used to store the distances between all the possible couples of functions and to determine the clusters, i.e., groups of functions that qualitatively perform similarly. Therefore, the computation of M costs $O(n^2\tau)$ operations, where n represents the cardinality of F and τ is the number of triangles of T.

We represent each cluster as an n-dimensional vector $\mathbf{x} = (x_i)$, whose components are real numbers expressing the level of participation of the function f_i in the cluster. A small value for x_i mean that f_i is weakly associated to the cluster, whereas a high value means that the f_i is strongly associated to it. The cohesiveness of the elements of a cluster is expressed in terms of the quadratic form $Q = \mathbf{x}^T M \mathbf{x}$ so that the (pairwise) clustering problem is reduced to that of finding a vector x maximizing Q; in practice the clustering problem is solved by the following quadratic program (see the formal proof in [21]):

$$\text{maximize } \mathbf{x}^T M \mathbf{x},\ x \in \Delta^n, \tag{1}$$

where Δ^n is the standard simplex of $\mathbb{R}^n$.

Equation (1) is (locally) solved through the so-called first-order replicator equation [28], adopting the iterative model (2):

$$x_i(k+1) = x_i(k) \frac{(M\mathbf{x})_i}{\mathbf{x}(k)^T M \mathbf{x}(k)}; \tag{2}$$

where k represents the iteration step. Once a cluster is identified, the corresponding rows and columns are removed from the matrix M. In summary:

1. select the components of $\mathbf{x}$ such that $x_i \geq \epsilon$, where ϵ is a cohesiveness threshold fixed by the user;

2. remove the functions already clustered by deleting the corresponding rows and columns in the matrix M;
3. reiterate on the remaining functions.

In the practical implementation of the method, the complexity of the cluster detection is $O(n \log n)$ where n is the cardinality of the set F.

2.2 *Shape Descriptions Based on Vector-Valued Functions*

A common scenario in applications is when two or more properties naturally concur to define and analyze the shape of an object. This is generally the case of the analysis and comparison of time-varying CT scans in medical imaging: for example, studying features like gradient magnitude and curvature is important to detect anomalies related to the cycles of the heart process. Sometimes it is desirable to study properties of a shape that are intrinsically multidimensional, such as the coordinates of an object in a 2D or 3D image (e.g. for tracking applications) [3], or photometric properties, which are usually taken into account in digital image segmentation. Further examples can be easily addressed in the context of computational biology and scientific computing for the simulation of natural phenomena. All these cases motivate the introduction of vector-valued functions for shape analysis and description.

2.2.1 Persistence Spaces

In the context of persistence, the use of vector-valued functions leads to a generalization of the notion of persistence diagram to that of persistence space [10]. For example, for a function $f = (f_1, \dots, f_n) : X \to \mathbb{R}^n$ and for any $u = (u_1, \dots, u_n), v = (v_1, \dots, v_n) \in \mathbb{R}^n$, with $u_i < v_i$ for all $i = 1, \dots, n$, the 0th persistence space associated with f keeps track of the connected components which remain disconnected passing from the lower level set X_u of X to the lower level set X_v, being a lower level set for f defined as $X_u = \{x \in X : f_i(x) \le u_i, i = 1, \dots, n\}$.

We refer to [10, 11] for more details on the notion of persistence space, as well as for a list of analogies with that of persistence diagram. For the present contribution, it is probably sufficient to emphasize that, similarly to the case of persistence diagrams, also for persistence spaces it is possible to prove stability under function perturbations, with respect to either the Hasdorff distance [11] or a generalization of the bottleneck distance [10]. In Sect. 3.2.1 we describe a computational framework for the practical usage of persistence spaces as textured shape descriptors.

3 An Application to the Retrieval of Textured 3D Images

In order to show the capabilities of the proposed shape descriptions, we test them in the context of textured 3D shape retrieval.

Nowadays, many sensors are able to acquire not only the 3D shape of an object but also its texture, and multiple-view stereo techniques enable the recovery of both geometric and photometric information directly from images. Photometric features contain rich information about the visual appearance of real objects [25], and play an important role in many shape analysis applications [27]. Indeed, the combination of shape and colour information has been studied for visual object recognition [24] and image processing [19]. In a shape retrieval context, methods dealing with photometric properties of shapes have been introduced, e.g. in [18, 20, 30].

Here we propose a shape description and retrieval work-flow able to handle both geometric and photometric information. The geometric information is based on a collection of Reeb graphs built on a set of real-valued functions, while photometric information is handled within the persistence setting, namely using persistence diagrams and spaces. In this way, a similarity measure between different shapes can be assessed, comparing them through suitable distances defined on the associated shape descriptors. Experiments are performed on the SHREC'13 dataset, a collection of 240 textured shapes, in the form of triangle meshes [12].

3.1 Geometric Description

In our framework we are interested in functions that somewhat are related to shape properties. In principle, this set of functions is very large; the space of the eigenfunctions of the Laplace-Beltrami operator or heat-kernel functions are possible examples as well as distance-based or geodesic-based functions [9]. We start from a set of 76 functions that in our idea reflects either intrinsic or extrinsic shape features (see [8] for a complete description of these functions), then we cluster them according to the method presented in Sect. 2.1.3. In this way, 12 functions have been selected out of the starting 76, each one being a representative of a cluster, see Fig. 3 for an example.

For each model in the dataset, such functions have been used to compute as many Reeb graphs, thus providing a battery of geometric shape descriptors. We note that the insertion in the loop of new functions would influence the type of geometric invariance captured by the description without modifying the global framework.

For a function f, two Reeb graphs G_1, G_2 corresponding to two shapes S_1, S_2 are compared by measuring pattern vectors built on the weighted Laplacian matrix L associated to the graphs, as proposed in Wilson et al. [29]. More in detail, the Laplacian of a graph G with weighted edges is given by: $L = D - A$, where D is the diagonal degree matrix whose non-zero elements D_{ii} are the number of edges

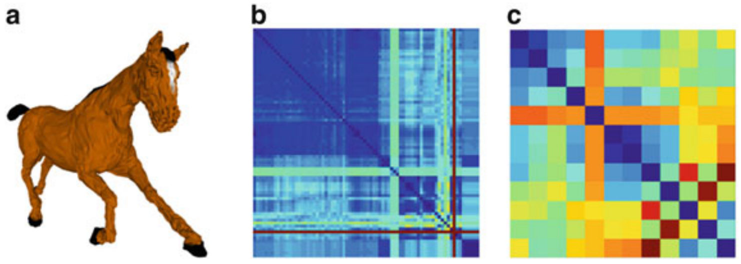


Fig. 3 A 3D textured model for the SHREC'13 dataset (**a**), the corresponding matrix M with 76 functions (**b**), and a reduced version with 12 functions (**c**). The distances range from *blue* (zero) to *red* (1); *blue regions* indicate functions that are strongly similar

which exit from the individual nodes v_i, and the matrix A is the weighted adjacency matrix defined as:

$$A_{ij} = \begin{cases} \mathscr{W}(e), & \text{if } e = (v_i, v_j) \in G; \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathscr{W}(e)$ is the weight associated to e. The matrix L is semidefinite positive. Denoting $\mathbf{e}_i$ and λ_i the ith eigenvector and eigenvalue of L, the matrix L may be expressed by the formula:

$$L = \Phi\Phi^t,$$

where $\Phi = (\Phi_{ij})_{i,j=1,\dots,n} = (\sqrt{\lambda_1}\mathbf{e}_1, \dots, \sqrt{\lambda_n}\mathbf{e}_n)$. Then, the set of elementary symmetric polynomials $S_j(e_1, \dots, e_n) = \sum_{i_1<\dots<i_j} e_{i_1}e_{i_2}\cdots e_{i_j}$, $j = 1, \dots, n$, is used to define the *feature vector* $\mathbf{B}$ of the graph in terms of the matrix:

$$\mathbf{B} = (f_{11}, \dots, f_{1n}, \dots, f_{n1}, \dots, f_{nn})^t,$$

where $f_{ij} = \mathrm{sign}(S_j(\Phi_{1i}, \dots, \Phi_{ni})) \cdot \ln(1 + |S_j(\Phi_{1i}, \dots, \Phi_{ni})|)$ are elements of a matrix $F = (f_{ij})_{i,j=1,\dots,n}$.

Then, the distance between the two graphs G_1, G_2 is defined by $d(G_1, G_2) = \|\mathbf{B}_1 - \mathbf{B}_2\|$. The final geometric distance between S_1 and S_2 is the sum of the 12 normalized distances between Reeb graphs, i.e. $D(S_1, S_2) = \sum d_{i=1}^{n}(d_i(G_1, G_2))$.

3.2 Photometric Description

Photometric properties can be represented in different colour spaces, such as the RGB and HSV spaces. We opt here for the CIELab colour space [15]. Defining measures in the CIELab colour space is justified by physiological studies, showing

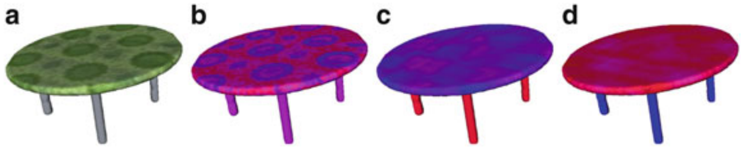


Fig. 4 A textured 3D model (**a**) and a representation of the associated L-, a- , and b-channels (**b–d**). All representations are color-coded from low (*blue*) to high (*red*) channel values

that it well represents how the human eye perceive colours. Moreover, in the CIELab space tones and colours are held separately: the L channel is used to specify the luminosity or the black and white tones, whereas the a channel specifies the colour as either a green or a magenta hue and the b channel specifies the colour as either a blue or a yellow hue, see Fig. 4 for an example.

We exploit such colour representation in the way we include photometric information in the persistence framework. Indeed, we consider the a, b coordinates as jointly defining a bivariate function over a given shape, whereas L is used separately as a scalar function. In this way, colour and intensity are treated separately. More precisely, we associate each considered shape S with two functions $f_L : S \to \mathbb{R}$ and $f_{a,b} : S \to \mathbb{R}^2$, the former taking each point $p \in S$ to the L-channel value at p, the latter to the pair given by the a- and the b-channel values at p, respectively. The values of f_L, $f_{a,b}$ are then normalized to range in the interval [0,1]. Lastly, S is associated with the 0th persistence diagram of f_L, say $\mathrm{Dgm}(f_L)$, and the 0th persistence space of $f_{a,b}$, denoted by $\mathrm{Spc}(f_{a,b})$.

For two shapes S_1 and S_2, their photometric distance is the normalized sum of the bottleneck distance $d_B\,(\mathrm{Dgm}_1(f_L), \mathrm{Dgm}_2(f_L))$ between the corresponding persistence diagrams and the Hausdorff distance $d_H\,(\mathrm{Spc}_1(f_{a,b}), \mathrm{Spc}_2(f_{a,b}))$ between the corresponding persistence spaces. Indeed, these distances allow for a stable comparison between the aforementioned persistence-based descriptors, thus implying robustness to noise in concrete applications.

3.2.1 Computational Framework

For a shape S represented as a triangle mesh with k vertices, the computation of the 0th persistence diagram $\mathrm{Dgm}(f_L)$ can be achieved by the use of a union find data structure, and is at most proportional to $k\alpha(k)$, where α is the notoriously slow growing inverse of the Ackermann function. For further details we refer to [14, 31].

As for a persistence space, we remark that, practically, it is not possible to compute it *exactly*. Indeed, differently from persistence diagrams, in general persistence spaces are not discrete collection of points in the real plane [10]. For instance, the 0th persistence space of a bivariate function like $f_{a,b}$ is an arrangement of continuous structures living in $\mathbb{R}^4$. Being continuous, this implies that concretely we can only get approximations of a 0th persistence space of $f_{a,b}$. Following [8], this can be done through a collection of suitable 0th persistence diagrams computed as follows:

1. let P be the set of points in the triangle mesh representing a shape S, and let M be the greatest value between $\max_{p\in P} a(p)$ and $\max_{p\in P} b(p)$;
2. being k the cardinality of P, fix $\log k$ equally distributed real values i in the interval $(0,1)$, and as many equally distributed real values j in the interval $[-M, M]$;
3. for every i and j, compute the 0th persistence diagram for the function $F_{ij} : P \to \mathbb{R}$ taking each $p \in P$ to
$$F_{ij}(p) = \max\left\{\frac{a(p)-j}{i}, \frac{b(p)+j}{1-i}\right\};$$
4. for every point (u, v) in the 0th persistence diagram of F_{ij}, project it back to a point $(u_1, u_2, v_1, v_2) \in \mathbb{R}^4$ through the equations
$$(u_1, u_2) = (i, 1-i)u + (j, -j), \quad (v_1, v_2) = (i, 1-i)v + (j, -j).$$

The set of all points in $\mathbb{R}^4$ obtained according to the above procedure gives us an approximation for the 0th persistence space of $f_{a,b}$. Being based on the computation of $\log^2 k$ persistence diagrams, approximating a persistence space in such a way is at most proportional to $k\alpha(k)\log^2(k)$.

3.3 Experimental Results

We tested the proposed framework in a retrieval experiment. The 240 models of the SHREC'13 dataset can be roughly grouped in ten subsets (four legs animals, ants, birds, glasses etc.), each one containing 24 elements obtained as follows. Six null models (two base meshes endowed with three different textures) are deformed according to four transformations. Deformations occur at different strength levels for the classes; nevertheless, the same texture may be shared by models belonging to different subsets, see Fig. 5 for some examples.

In our experiments, models are associated with the geometric and the photometric shape descriptions detailed in the previous sections. The final similarity score between two shapes is the sum of the geometric and photometric distances, which are beforehand normalized to range in the interval [0,1]. Then, each model is used in turn as a query against the remaining part of the database, with the goal of retrieving the most similar objects. In doing this, we assume that two objects belong to the same class if they share both geometric and texture information. We compared our method **(OurMtd)** with four ones among the best of [12]: a method based on Scale Invariant Heat Kernels **(A2)**, the Color-weighted Histograms of Area Projection Transform **(Gi)**, a method based on 2D multi-view and bag-of-features

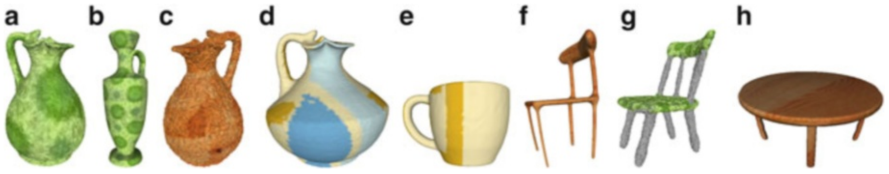
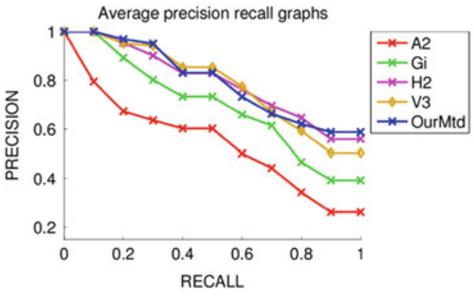


Fig. 5 Models in the SHREC'13 benchmark belonging to different subsets: four amphoras, a cup, two chairs and a table. Models in the same subset are characterized by geometric (**a**, **b**, **d**), topological (**f**, **g**) or texture (**a**, **b**) deformations. Different textures are used for models in the same subset (**a**, **c**, **d**). However, a similar texture may be applied to models in different subsets [(**c**, **h**), (**d**, **e**)]



Run	NN	FT	ST
A2	0.508	0.561	0.730
Gi	0.788	0.658	0.748
G2	0.898	0.733	**0.893**
V3	0.909	0.733	0.862
OurMtd	**0.943**	**0.741**	0.867

Fig. 6 *Left*: Precision-recall curves for all methods. *Right*: Retrieval performances on the whole dataset. For each evaluation measure, best results are in *bold text*

approach **(G2)**, and an approach merging a shape description based on geodesic distance matrices with RGB histograms **(V3)**. We used Average precision-recall curves, Nearest Neighbor (NN), First Tier (FT) and Second Tier (ST) as evaluation measures.

Figure 6 (left) shows the performances of all methods in terms of average precision-recall curves. Recall indicates the ratio of the models in a given class returned within the top matches; precision indicates the ratio of the top matches which are members of the correct class [2]. The final curve is averaged over all the models. The larger the area below the curve, the better the performance under examination.

Figure 6 (right), columns 2–4, reports the performances for all methods according to the NN, FT and ST measures. These evaluation measures aim at checking the fraction of models in the query's class also appearing within the top k retrievals, where k can be 1 (NN), the size of the query's class (FT), or the double size of the query's class (ST). The final score, ranging between 0 and 1, is an average over all the models.

Conclusion

Combining shape and photometric properties offers several advantages: the information conveyed can be encoded in compact and informative geometric/topological descriptions; the resulting framework is flexible because of the translation of the shapes properties in terms of functions; it is robust to noise and invariant with respect to the groups of transformations the functions are invariant.

The experiments we have shown are performed over a synthetic dataset combining geometric and photometric information. However, our framework can be easily tuned to more specific, real-world datasets, for instance repositories of archaeological artefacts or protein models: in that case the selection of geometric functions will reflect the nature of the new collection and also the contribution of the photometric description might change according to the different users needs. In addition, we highlight that the method is not limited to geometric and photometric information: indeed we are able to include and combine any property, which can be defined by real or vector-valued function (e.g., an electrostatic potential in the case of protein models).

Finally, we think that this work paves the road to the automatic selection of properties (i.e., functions) that are informative for specific classes of shapes and goes towards the selection of intra-class shape invariants. In practice, the method evolves with the insertion of new models (it allows for the inclusion of other properties in the processing pipeline), it is modular (it is possible to select informative subsets of properties) and adaptive (the combination rule of the description can be changed, and even learned using a learning set extracted from the original database). This fact naturally leads to the integration of the framework proposed here with semantic annotation and statistic techniques, such as semi-supervised learning strategies.

Acknowledgements This work is partially funded by the EU project IQmulus under grant agreement no. FP7-ICT-2011-318787 and the CNR Flagship project INTEROMICS, InterOmics PB05, research unit WP 15.

References

1. Anile, M.A., Falcidieno, B., Gallo, G., Spagnuolo, M., Spinello, S.: Modeling uncertain data with fuzzy B-splines. Fuzzy Sets Syst. **113**(3), 397–410 (2000)
2. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Boston (1999)
3. Bhatia, H., Norgard, G., Pascucci, V., Bremer, P.-T.: The Helmholtz-Hodge Decomposition - A Survey. IEEE Trans. Vis. Comput. Graph. **19**(8), 1386–1404 (2013)

4. Biasotti, S.: Topological coding of surfaces with boundary using Reeb graphs. Comput. Graph. Geom. **7**(1), 31–45 (2005)
5. Biasotti, S., Falcidieno, B., Spagnuolo, M.: Extended Reeb Graphs for surface understanding and description. In: Borgefors, G., di Baja, G.S. (eds.) Discrete Geometry for Computer Imagery. Lecture Notes in Computer Science, vol. 1953, pp. 185–197. Springer, Berlin (2000)
6. Biasotti, S., Falcidieno, B., Frosini, P., Giorgi, D., Landi, C., Patanè, G., Spagnuolo, M.: 3D shape description and matching based on properties of real functions. In: EG 2007 Tutorial, pp. 949–998 (2007)
7. Biasotti, S., De Floriani, L., Falcidieno, B., Frosini, P., Giorgi, D., Landi, C., Papaleo, L., Spagnuolo, M.: Describing shapes by geometrical-topological properties of real functions. ACM Comput. Surv. **40**(4), 1–87 (2008)
8. Biasotti, S., Cerri, A., Giorgi, D., Spagnuolo, M.: PHOG: photometric and geometric functions for textured shape retrieval. Comput. Graph. Forum **32**(5), 13–22 (2013)
9. Biasotti, S., Spagnuolo, M., Falcidieno, B.: Grouping real functions defined on 3D surfaces. Comput. Graph. **37**(6), 608–619 (2013)
10. Cerri, A., Landi, C.: The persistence space in multidimensional persistent homology. In: Gonzalez-Diaz, R., Jimenez, M.-J., Medrano, B. (eds.) Discrete Geometry for Computer Imagery. Lecture Notes in Computer Science, vol. 7749, pp. 180–191. Springer, Berlin (2013)
11. Cerri, A., Landi, C.: Persistence space of vector-valued continuous functions. Technical Report IMATI no. 4/13 (2013)
12. Cerri, A., Biasotti, S., Abdelrahman, M., Angulo, J., Berger, K., Chevallier, L., El-Melegy, M.T., Farag, A.A., Lefebvre, F., Giachetti, A., Guermoud, H., Liu, Y.-J., Velasco-Forero, S., Vigouroux, J.-R., Xu, C.-X., Zhang, J.-B.: Shrec'13 track: retrieval on textured 3d models. In: Proceedings of Eurographics Workshop on 3D Object Retrieval, EG 3DOR, pp. 73–80 (2013)
13. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of persistence diagrams. Discrete Comput. Geom. **37**(1), 103–120 (2007)
14. Edelsbrunner, H., Harer, J.: Computational Topology: An Introduction. American Mathematical Society, Providence, 2010.
15. Fairchild, M.D.: Color Appearance Models, 2nd edn. Wiley, Chichester, (2005)
16. Falcidieno, B., Spagnuolo, M.: A shape abstraction paradigm for modeling geometry and semantics. In: Proceedings of the Computer Graphics International 1998, CGI '98, pp. 646–656. IEEE Computer Society, Washington (1998)
17. Harvey, W., Rübel, O., Pascucci, V., Bremer, P.-T., Wang, Y.: Enhanced topology-sensitive clustering by reeb graph shattering. In: Topological Methods in Data Analysis and Visualization II, pp. 77–90. Springer, Berlin (2012)
18. Kanezaki, A., Harada, T., Kuniyoshi, Y.: Partial matching of real textured 3D objects using color cubic higher-order local auto-correlation features. Vis. Comput. **26**(10), 1269–1281 (2010)
19. Kimmel, R., Malladi, R., Sochen, N.: Images as embedded maps and minimal surfaces: movies, color, texture, and volumetric medical images. Int. J. Comput. Vis. **39**(2), 111–129 (2000)
20. Liu, Y.-J., Zheng, Y.-F., Lv, L., Xuan, Y.-M., Fu, X.-L.: 3D model retrieval based on color + geometry signatures. Vis. Comput. **28**(1), 75–86 (2012)
21. Pavan, M., Pelillo, M.: Dominant sets and pairwise clustering. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 167–172 (2007)
22. Reeb, G.: Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences **222**, 847–849 (1946)
23. Shinagawa, Y., Kunii, T.: Constructing a Reeb graph automatically from cross sections. IEEE Comput. Graph. Appl. **11**, 44–51 (1991)
24. Slater, D., Healey, G.: Combining color and geometric information for the illumination invariant recognition of 3-d objects. In: Proceedings of IEEE ICCV, pp. 563–568 (1995)
25. Tanaka, J., Weiskopf, D., Williams, P.: The role of color in high-level vision. Trends Cogn. Sci. **5**, 211–215 (2001)
26. Thompson, D.W.: On Growth and Form. Cambridge University Press, Cambridge (1942)

27. Thorstensen, N., Keriven, R.: Non-rigid shape matching using geometry and photometry. In: Proceedings of 9th Asian Conference on Computer Vision, pp. 644–654. Springer, Berlin (2010)
28. Weibull, J.: Evolutionary Game Theory. MIT Press, Cambridge (1995)
29. Wilson, R.C., Hancock, E.R., Luo, B.: Pattern vectors from algebraic graph theory. IEEE Trans. Pattern Anal. Mach. Intell. **7**(27), 1112–1124 (2005)
30. Zaharescu, A., Boyer, E., Horaud, R.: Keypoints and local descriptors of scalar functions on 2D manifolds. Int. J. Comput. Vis. **100**(1), 78–98 (2012)
31. Zomorodian, A., Carlsson, G.: Computing persistent homology. Discrete Comput. Geom. **33**(2), 249–274 (2005)

Part III
Multi-Variate Data Analysis: Structural Techniques

3D Symmetric Tensor Fields: What We Know and Where To Go Next

Eugene Zhang and Yue Zhang

1 Introduction

This chapter discusses the state of research in the visualization of second-order, symmetric tensor fields. Such tensor fields appear in a wide range of applications, such as the metric curvature tensor in differential geometry, the stress and strain tensor in solid and fluid mechanics, and the diffusion tensor in medical imaging. While the word *tensor* can be more general, we will use the word to mean symmetric, second-order tensors in the remainder of this chapter and therefore omit the terms symmetric and second-order.

There have been many recent advances in the analysis and visualization of tensor fields, made by the geometry processing, scientific visualization, image processing, and medical imaging communities. We believe this is a time for reflection, when we look back at our achievements and identify future directions. In this book chapter, we will review facts about tensor fields that we think are the most important, i.e., that should appear in an introductory textbook on tensor fields. Furthermore, we make an attempt to point out problems that we believe are the most pressing in advancing the research of tensor field analysis and visualization. We hope that our effort can lead to more discussions and exchange of ideas.

E. Zhang (✉)
School of Electrical Engineering and Computer Science, 2111 Kelley Engineering Center, Oregon State University, Corvallis, OR 97331, USA
e-mail: zhange@eecs.oregonstate.edu

Y. Zhang
School of Electrical Engineering and Computer Science, 3117 Kelley Engineering Center, Oregon State University, Corvallis, OR 97331, USA
e-mail: zhangyue@eecs.oregonstate.edu

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_7

Section 2 reviews past research in symmetric tensor field visualization. In Sect. 3 we review what we believe are the most important results about tensor fields. In Sect. 4 we discuss open problems that we feel are the most pressing before concluding in section "Conclusion".

2 Previous Work

There has been much work on the topic of 2D and 3D tensor fields for medical imaging and scientific visualization. We refer the readers to the recent survey by Kratz et al. [1]. Here we only refer to the research most relevant to this chapter.

Most of the earlier research on symmetric tensor field analysis and visualization focused on the *diffusion tensor*, a semi-positive-definite tensor extracted from brain imaging. The main focus is two-folds. First, fibers following the eigenvectors of the diffusion tensor are computed. Second, appropriate glyphs are designed to help the user understand the diffusion tensor. This has led to various measures for the anisotropy in the diffusion tensor, such as the *relative anisotropy* and the *fractional anisotropy* [2]. Unfortunately these measures do not distinguish between the linear and planar types of tensors. Westin et al. [3] overcome this by modeling the anisotropy using three coefficients that measure the linearity, planarity, and sphericalness of a tensor, respectively. The aforementioned measures are designed for semi-positive-definite tensors, such as the diffusion tensor. We refer interested readers to the book [4] and the survey by Zhang et al. [5] on research related to this area.

There have been a number of approaches to visualize 2D and 3D symmetric tensor fields. Delmarcelle and Hesselink [6] introduce the notion of hyperstreamlines for the visualization of 2D and 3D symmetric tensor fields. Zheng and Pang [7] visualize hyperstreamlines by adapting the well-known Line Integral Convolution (LIC) method of Cabral and Leedom [8] to symmetric tensor fields which they term *HyperLIC* [7]. Zheng and Pang also deform an object to demonstrate the deformation tensor [9]. These visualization techniques have been later used for geomechanics data sets [10]. One of the fundamental differences between the diffusion tensor and the other symmetric tensors from mechanics (stress, strain, symmetric part of the velocity gradient tensor) is that the former is semi-positive definite (no negative eigenvalues) while the latter can have both semi-positive and negative eigenvalues. Schultz and Kindlmann [11] extend ellipsoidal glyphs that are traditionally used for semi-positive-definite tensors to *superquadric glyphs* which can be used for general symmetric tensors. In our work, we focus on the analysis of traceless tensors which lead to a different glyph design.

Delmarcelle and Hesselink [6, 12] introduce the topology of 2D symmetric tensor fields as well as conduct some preliminary studies on 3D symmetric tensors in the context of flow analysis. Hesselink et al. later extend this work to 3D symmetric tensor fields [13] and study the degeneracies in such fields. Zheng and

Pang [14] point out that triple degeneracy, i.e., a tensor with three equal eigenvalues, cannot be extracted in a numerically stable fashion. They further show that double degeneracies, i.e., only two equal eigenvalues, form lines in the domain. In this work and subsequent research [15], they provide a number of degenerate curve extraction methods based on the analysis of the discriminant function of the tensor field. Furthermore, Zheng et al. [16] point out that near degenerate curves the tensor field exhibits 2D degenerate patterns and define separating surfaces which are extensions of separatrices from 2D symmetric tensor field topology. Tricoche et al. [17] convert the problem of extracting degenerate curves in a 3D tensor field to that of finding the ridge and valley lines of an invariant of the tensor field, thus leading to a more robust extraction algorithm. Tricoche and Scheuermann [18] introduce a topological simplification operation which removes two degenerate points with opposite tensor indexes from the field. Zhang et al. [19] propose an algorithm to perform this *pair cancellation* operation by converting the tensor field to a vector field and reusing similar operations in vector field topological simplification [20].

More recently, Wagner et al. [21] describe a viewpoint that applies to tensors of any type, including symmetric second-order tensors. The main idea is based on the notion of harmonic field analysis, an application of spectral analysis.

3 Important Facts About Tensor Fields

In this section we review the most relevant background on tensors and tensor fields.

3.1 Tensors

A K-dimensional (symmetric) tensor $\mathbf{T}$ has K real-valued *eigenvalues*: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_K$. When all the eigenvalues are non-negative, the tensor is referred to as *semi-positive-definite*. The largest and smallest eigenvalues are referred to as the *major eigenvalue* and *minor eigenvalue*, respectively. When $K = 3$, the middle eigenvalue is referred to as the *medium eigenvalue*. An eigenvector belonging to the major eigenvalue is referred to as a *major eigenvector*. Medium and minor eigenvectors can be defined similarly. Eigenvectors belonging to different eigenvalues are mutually perpendicular.

The trace of a tensor $\mathbf{T} = (\mathbf{T}_{ij})$ is $trace(\mathbf{T}) = \sum_{i=1}^{K} \lambda_i$. $\mathbf{T}$ can be uniquely decomposed as $\mathbf{D} + \mathbf{A}$ where $\mathbf{D} = \frac{trace(\mathbf{T})}{K}\mathbb{I}$ ($\mathbb{I}$ is the K-dimensional identity matrix) and $\mathbf{A} = \mathbf{T} - \mathbf{D}$. The *deviator* $\mathbf{A}$ is a *traceless* tensor, i.e., $trace(\mathbf{A}) = 0$. Note that $\mathbf{T}$ and $\mathbf{A}$ have the same set of eigenvectors. Consequently, the anisotropy in a tensor field can be defined in terms of its deviator tensor field. Another nice property of the set of traceless tensors is that it is closed under matrix addition and scalar multiplication, making it a linear subspace of the set of tensors.

The magnitude of a tensor $\mathbf{T}$ is $||\mathbf{T}|| = \sqrt{\sum_{1 \le i,j \le K} T_{ij}^2} = \sqrt{\sum_i^K \lambda_i^2}$, while the determinant is $\prod_{i=1}^K \lambda_i$. When $K = 3$, there is an additional quantity, $minor(\mathbf{T}) = \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3$, which is not to be confused with the term *minor eigenvalue* of $\mathbf{T}$. For a traceless tensor $\mathbf{T}$, when $K = 2$ we have $\frac{||\mathbf{T}||^2}{2} = -|\mathbf{T}|$ while when $K = 3$ we have $\frac{||\mathbf{T}||^2}{2} = -minor(\mathbf{T})$. These facts demonstrate two things. First, the formulas of these tensor invariants become much simpler when considering traceless tensors without losing anisotropic information (difference in the eigenvalues and eigenvector information) in the tensor. Second, for 3D tensors, the trace of the tensor, the magnitude and determinant of the deviator of the tensor uniquely determine the three eigenvalues, yet with a more meaningful description than considering the three eigenvalues independently.

A tensor is *degenerate* when there are repeating eigenvalues. In this case, there exists at least one eigenvalue whose corresponding eigenvectors form a higher-dimensional space than a line. When $K = 2$ a degenerate tensor must be a multiple of the identity matrix. When $K = 3$, there are two types of degenerate tenors, corresponding to three repeating eigenvalues (*triple degenerate*) and two repeating eigenvalues (*double degenerate*), respectively. The *discriminant* of a tensor [14] is defined as $\prod_{1 \le i < j \le K}(\lambda_i - \lambda_j)^2$. A tensor is degenerate if and only if its discriminant is zero.

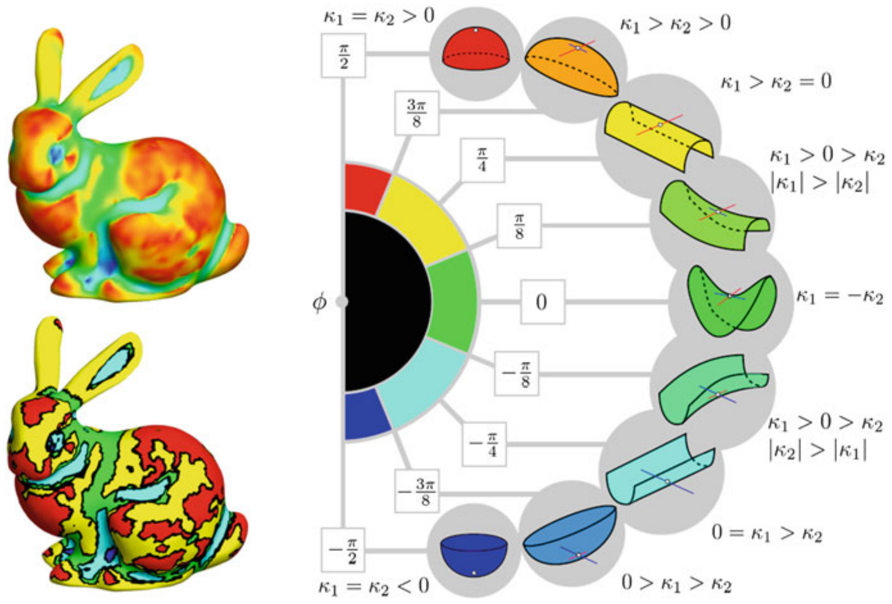
In 2D, the aforementioned trace-deviator decomposition can turn any tensor into the sum of a degenerate tensor (isotropic) and a non-degenerate tensor (anisotropic). As an example, consider curvature tensor of two-dimensional manifolds embedded in 3D. It has the form $K = U' \begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix} U$ where $\kappa_1 \ge \kappa_2$ are the *principal curvatures* and the columns of U are the corresponding *principal curvature directions*.

The isotropic-deviator decomposition amounts to [22]

$$\begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix} = \frac{\sqrt{\kappa_1^2 + \kappa_2^2}}{\sqrt{2}} \left[\sin\phi \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \cos\phi \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \right] \tag{1}$$

where $\kappa_1^2 + \kappa_2^2 = ||K||^2$ is the *total curvature* and $\phi = \arctan(\frac{\kappa_1+\kappa_2}{\kappa_1-\kappa_2})$ measures the relative strength between the isotropic and anisotropic components in the curvature tensor. Figure 1 shows the canonical shapes corresponding to various representative ϕ values and how ϕ can be used as a classification of surface geometry on the bunny surface.

For $3D$ tensors, the complexity increases. A traceless tensor can still be degenerate, i.e., two repeating eigenvalues. Moreover, there are now two types of double degenerate tensors. The first type is *linear*, where $\lambda_1 > \lambda_2 = \lambda_3$. In this case, $\lambda_2 = \lambda_3$ is the *repeated eigenvalue*, while λ_1 (major eigenvalue) is the *non-repeated eigenvalue*. The second type is *planar*, where $\lambda_1 = \lambda_2 > \lambda_3$. For planar tensors, the non-repeated eigenvalue is the minor eigenvalue.



$\kappa_1 = \kappa_2 > 0$
$\kappa_1 > \kappa_2 > 0$
$\kappa_1 > \kappa_2 = 0$
$\kappa_1 > 0 > \kappa_2$
$|\kappa_1| > |\kappa_2|$
$\kappa_1 = -\kappa_2$
$\kappa_1 > 0 > \kappa_2$
$|\kappa_2| > |\kappa_1|$
$0 = \kappa_1 > \kappa_2$
$0 > \kappa_1 > \kappa_2$
$\kappa_1 = \kappa_2 < 0$
ϕ
$\frac{\pi}{2}$
$\frac{3\pi}{8}$
$\frac{\pi}{4}$
$\frac{\pi}{8}$
0
$-\frac{\pi}{8}$
$-\frac{\pi}{4}$
$-\frac{3\pi}{8}$
$-\frac{\pi}{2}$

Fig. 1 Surface classification scheme based on the shape index $\phi \in [\pi/2, \pi/2]$ is color mapped to the [*BLUE*, *RED*] arc in HSV color space: *Left top*: continuous mapping. *Bottom*: binned classification. The legend (*right*) shows surfaces patches which are locally similar to points with given values. This figure is a courtesy of [22], ©2012 IEEE

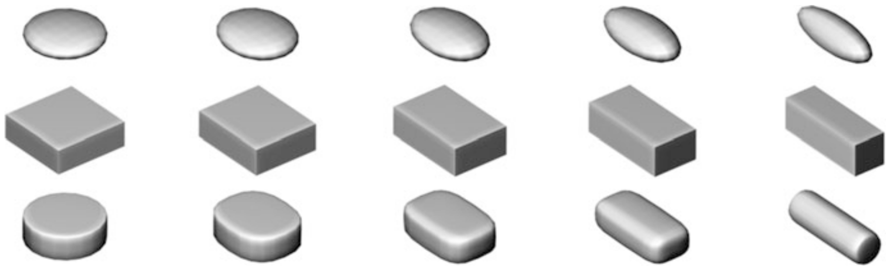


Fig. 2 Visualization of the impact of tensor mode using various types glyphs: ellipsoids (*top*), boxes (*middle*), and superquadrics (*bottom*). The *leftmost* and *rightmost columns* correspond to planar and linear degenerate tensors, respectively. Notice how planar degenerate tensors transition gradually towards linear degenerate tensors through non-degenerate tensors (*middle columns*)

A traceless tensor **T** can be characterized by its *mode* [17], defined as $mode(\mathbf{T}) = \frac{3\sqrt{6}|\mathbf{T}|}{||\mathbf{T}||^3}$. The mode achieves its maximum value 1 when **T** is a linear degenerate tensor and its minimal values -1 when **T** is a planar degenerate tensor. Figure 2 shows the transition from planar degenerate tensors (left column) to linear degenerate tensors (right column) through non-degenerate tensors (middle columns).

If all the eigenvalues of a tensor **T** are positive, the tensor is *positive-definite*. Examples of semi-positive-definitions include the diffusion tensor from medical imaging and the metric tensor from differential geometry. Tensors that are indefinite include the curvature tensor from differential geometry and the stress and strain tensors from solid and fluid mechanics. For semi-positive-definite tensors, their anisotropy can be measured by the so-called *fractional anisotropy*:

$$FA(\mathbf{T}) = \sqrt{\frac{1}{2}}\frac{\sqrt{(\lambda_1-\lambda_2)^2+(\lambda_2-\lambda_3)^2+(\lambda_3-\lambda_1)^2}}{||T||} \tag{2}$$

The fractional anisotropy achieves the minimum value 0 when **T** has three equal eigenvalues. It achieves the maximal value 1 when **T** is degenerate.

3.2 Tensor Fields

We now review *tensor fields*, which are tensor-valued functions over some domain $\Omega \subset \mathbb{R}^K$. A tensor field can be thought of as K eigenvector fields, corresponding to the K eigenvalues. A *hyperstreamline* with respect to an eigenvector field $e_i(p)$ is a $3D$ curve that is tangent to e_i everywhere along its path. Two hyperstreamlines belonging to two different eigenvalues can only intersect at the right angle, since eigenvectors belonging to different eigenvalues must be mutually perpendicular.

Hyperstreamlines are usually curves. However, they can occasionally consist of only one point, where there are more than one choice of lines that correspond to the eigenvector field. This is precisely where the tensor field is degenerate. A point $p_0 \in \Omega$ is a *degenerate point* if $\mathbf{T}(p_0)$ is degenerate. The topology of a tensor field consists of its degenerate points.

In $2D$, the set of degenerate points of a tensor field are isolated points under numerically stable configurations, when the topology does not change given sufficiently small perturbation in the tensor field. An isolated degenerate point can be measured by its *tensor index* [19], defined in terms of the *winding number* of one of the eigenvector fields on a loop surrounding the degenerate point. The most fundamental types of degenerate points are *wedges* and *trisectors* (see Fig. 3), with a tensor index of $\frac{1}{2}$ and $-\frac{1}{2}$, respectively. Let $L\mathbf{T}_{p_0}(p)$ be the local linearization of $\mathbf{T}(p)$ at a degenerate point $p_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$, i.e.,

$$L\mathbf{T}_{p_0}(p) = \begin{pmatrix} a_{11}(x-x_0)+b_{11}(y-y_0) & a_{12}(x-x_0)+b_{12}(y-y_0) \\ a_{12}(x-x_0)+b_{12}(y-y_0) & a_{22}(x-x_0)+b_{22}(y-y_0) \end{pmatrix} \tag{3}$$

Then $\delta = \left|\begin{pmatrix} \frac{a_{11}-a_{22}}{2} & a_{12} \\ \frac{b_{11}-b_{22}}{2} & b_{12} \end{pmatrix}\right|$ is invariant under the change of basis. Moreover, it characterizes the type of degenerate points [12]. More specifically, p_0 is a wedge when $\delta > 0$ and a trisector when $\delta < 0$. When $\delta = 0$, p_0 is a higher-order degenerate point. A *major separatrix* is a hyperstreamline emanating from a degenerate point following the major eigenvector field. A *minor separatrix* is defined similarly. The directions in which a separatrix can occur at a degenerate point p_0 can be computed as follows.

Let $L\mathbf{T}_{p_0}(p)$ be the local linearization at p_0. Since $L\mathbf{T}_{p_0}(p)$ is linear, the major eigenvector field e_1 and the minor eigenvector field e_2 respectively have a constant orientation along any ray emanating from p_0. A ray whose direction is parallel to the major eigenvector field along it is referred to as a *major separatrix*, while a ray parallel to the minor eigenvector field along it is referred to as a *minor separatrix*. Finding either major separatrix or minor separatrix directions leads to a cubic polynomial with either one or three solutions under stable conditions [23]. It is known that around a trisector there must be three solutions, corresponding to three separatrices that divide the neighborhood of p_0 into three sectors, thus the namesake. Around a wedge there can be either one sector or three sectors.

The total tensor index of a continuous tensor field over a two-dimensional manifold is equal to the *Euler characteristic* of the underlying manifold. Consequently, it is not possible to remove one degenerate point. Instead, a pair of degenerate points with opposing tensor indexes (a wedge and trisector pair) must be removed simultaneously [19].

The topology in 3D tensor fields is more challenging. While triple degeneracies can exist, they are numerically unstable [14], i.e., can disappear under arbitrarily small perturbations. Stable topological features in 3D tensor fields consists of double degenerate points that form curves. A curve consists of either purely linear degenerate points, or purely planar degenerate points. Furthermore, along a degenerate curve the tensor field exhibits 2D tensor degenerate patterns [16].

To be more precise, consider a degenerate curve γ and a point $p_0 \in \gamma$. The *repeated plane* at p_0 is the plane that is perpendicular to the non-repeated eigenvector at p_0. Recall that the non-repeated eigenvector is the eigenvector corresponding to the non-repeated eigenvalue of $\mathbf{T}(p_0)$, which is the major eigenvalue for linear degenerate points and the minor eigenvalue for the planar degenerate points.

Let (v_1, v_2) be an orthonormal basis for the repeated plane at p_0 and v_3 be a unit non-repeated eigenvector for $\mathbf{T}(p_0)$ such that (v_1, v_2, v_3) form a right-handed orthonormal basis. Under this basis the tensor field has the form:

$$\begin{pmatrix} \mathbf{M}_{11}(p) & \mathbf{M}_{12}(p) & \mathbf{M}_{13}(p) \\ \mathbf{M}_{12}(p) & \mathbf{M}_{22}(p) & \mathbf{M}_{23}(p) \\ \mathbf{M}_{13}(p) & \mathbf{M}_{23}(p) & \mathbf{M}_{33}(p) \end{pmatrix} \tag{4}$$

When $p \to p_0$, $\mathbf{M}_{13}(p)$ and $\mathbf{M}_{23}(p)$ approach 0. Moreover, if p_0 is a linear degenerate point, the difference between the eigenvectors of $\mathbf{M}(p) = \begin{pmatrix} \mathbf{M}_{11}(p) & \mathbf{M}_{12}(p) \\ \mathbf{M}_{12}(p) & \mathbf{M}_{22}(p) \end{pmatrix}$ and the projection of the eigenvectors corresponding to λ_2 and λ_3 approach 0 as p approaches p_0. A similar statement can be made for a planar degenerate point, except that we replace λ_2 and λ_3 with λ_1 and λ_2, respectively.

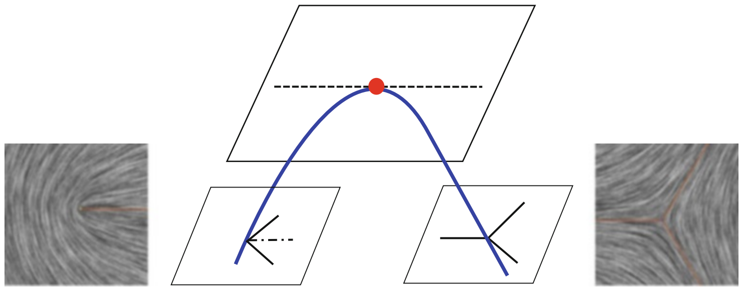


Fig. 3 Near a degenerate curve, 3D tensor fields exhibit 2D tensor patterns such as wedges (in the *left plane*, also the *inset* to the *left*) and trisectors (in the *right plane*, also the *inset* in the *right*). Such patterns persist along the curve until transition points (*top of the curve*), where the wedge and trisector cancel each other

$\mathbf{M}(p)$ restricted to the repeating plane P at p_0 has at least one degenerate point, p_0. The 2D tensor pattern near p_0 is typically either a wedge or a trisector. See Fig. 3 for an illustration. The tangent of degenerate curve γ at p_0 is $\bigtriangledown(\mathbf{M}_{11}(p) - \mathbf{M}_{22}(p)) \times \bigtriangledown\mathbf{M}_{12}(p)$. This is the direction in which $\mathbf{M}(p)$ remains degenerate [16]. When $\bigtriangledown(\mathbf{M}_{11}(p) - \mathbf{M}_{22}(p)) \times \bigtriangledown\mathbf{M}_{12}(p) = 0$, the tangent to the degenerate curve is parallel to the repeated plane at p_0. In this case, the 2D degenerate pattern of $\mathbf{M}$ inside the repeated plane is a higher-order degenerate point. It is referred to as a *transition point* as it is the boundary between wedge points on γ and trisector points on γ.

Figure 3 illustrates this with an example. When taking a stack of cut planes moving upwards, we can observe two 2D degenerate patterns (wedge and trisector) in the projected tensor field moving closer and eventually canceling each other at the transition point.

A *separating surface* is the union of all hyperstreamlines emanating from a degenerate curve following one of the eigenvector fields. There are three sheets of such a surface around the trisector segments along a degenerate curve and one or three sheets around the wedge segments.

A degenerate curve can be extracted by finding the zeroth levelset of the discriminant function [14]. In one such approach, degenerate points are first found on the faces of cells in the mesh on which the tensor data is represented. Given a face, we start with a random point in the plane containing the face and perform the Newton-Raphson method on the discriminant function. Once all such points are found on the faces of cells, they are connected by straight line segments that approximate the degenerate curves between these points. The tangent to the degenerate curves at the points on the faces are used to remove ambiguity in how these points are connected. In a more refined approach, numerical integration is used to actually trace out the segments between degenerate points on the faces.

Degenerate curves can also be extracted with the realization that they are *crease lines* of fractional anisotropy function and the mode function of the tensor field [17]. A point p_0 is a *ridge point* of a scalar function $\mathbf{f}$ if $\bigtriangledown\mathbf{f}(p_0) \times e_2 = \bigtriangledown\mathbf{f}(p_0) \times e_3 = 0$

where e_2 and e_3 are the eigenvectors corresponding to negative eigenvalues $\lambda_2 \geq \lambda_3$ of the Hessian of $\mathbf{f}$. A *valley point* p_0 satisfies that $\nabla \mathbf{f}(p_0) \times e_1 = \nabla \mathbf{f}(p_0) \times e_2 = 0$ where e_1 and e_2 are the eigenvectors corresponding to positive eigenvalues $\lambda_1 \geq \lambda_2$ of the Hessian of $\mathbf{f}$. A crease line can then be extracted using the well-known *Marching Cubes* method.

4 Open Problems

In this section we describe some of the problems that we feel need the attention of the various communities that deal with tensor fields.

4.1 Natural Parameterizations

Equation (1) describes a natural description of the curvature tensor. In fact, such a description applies to any 2D symmetric tensors. More specifically, a $2D$ tensor $\mathbf{T}$ can be rewritten as

$$\begin{pmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{12} & \mathbf{T}_{22} \end{pmatrix} = \frac{||\mathbf{T}||}{\sqrt{2}} \left[\sin\phi \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \cos\phi \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix} \right] \tag{5}$$

where $\theta = \arctan(\frac{2\mathbf{T}_{12}}{\mathbf{T}_{11}-\mathbf{T}_{22}})$ and $\phi = \arctan(\frac{\mathbf{T}_{11}+\mathbf{T}_{22}}{\sqrt{(\mathbf{T}_{11}-\mathbf{T}_{22})^2+4\mathbf{T}_{12}^2}})$. This can be considered as a reparameterization of the set of 2D symmetric tensors, a three-dimensional linear space, by spherical coordinates $\rho = ||\mathbf{T}|| \in [0, \infty)$, $\theta \in [0, 2\pi)$, and $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. This parameterization maps the set of unit tensors to S^2, the unit sphere. This sphere can be considered as sweeping the half circle in the right image of Fig. 1 by the vertical axis. Travelling along any latitude, the tensors maintain the eigenvalues while the eigenvectors rotate at the rate of $\frac{\theta}{2}$. Along any longitude, we see the same transition shown on the curvature tensor in Fig. 1 (right). Starting from the North Pole (two identical positive eigenvalues), going south transitions through two positive but unequal eigenvalues, one positive eigenvalue and one zero eigenvalue, one positive and one negative eigenvalue, then one zero and one negative eigenvalue, two negative but unequal eigenvalues, and eventually ending up at the South Pole (two equal negative eigenvalues). When interpolating between two tensors that do not necessarily have the same magnitude, the projection of the interpolation onto the unit sphere is the shorter segment of the greatest circle between the projection of the two tensors. This means that as one travels along this interpolation path, the eigenvectors of the first tensor rotates towards the eigenvectors of the second tensor monotonically.

Similar parameterization is highly desired for $3D$ tensor fields. However, there are some fundamental challenges. First, it is not obvious how the eigenvectors of a tensor depends on the entries of the tensor. Second, 3D rotations do not commute. Third, the space of 3D symmetric tensors is six-dimensional. While the unit traceless tensors form a four-dimensional subspace, it is still not clear how to model this subspace. Fourth, there is a four-way symmetry in the frames generated from a tensor. That is, there are four right-handed frames derived from the eigenvectors of a tensor.

Finding an intuitive, preferably geometric relationship between the entries of tensors and the corresponding eigenvectors of the same tensor can be key to finding a natural parameterization and thus interpolation for the set of 3D tensors.

4.2 Physical Interpretation for Local Degenerate Patterns

It is known that there are two 2D fundamental degenerate patterns: wedges and trisectors. When driving on two-way street networks, there are three most basic ways of reversing driving directions: (a) a U-turn (wedge), a three-point turn (trisector), and (c) driving in reverse (illegal). While this seems intuitive, the interpretation of wedges and trisectors in domain applications remains less clear.

For example, in the curvature tensor field, it is understood that wedges and trisectors appear around the umbilical points (equal principal curvatures). The type of the point (wedge or trisector) depends on the local Gaussian curvature. When the Gaussian curvature is positive, wedges appear. When the Gaussian curvature is negative, trisectors appear. Geometrically, wedges appear naturally near the tips of protrusions (elliptical), while the trisectors appear naturally in regions where two protrusions join each other (hyperbolic). In contrast, it is not clear what wedges and trisectors represent in more physical driven tensors such as the stress and strain tensors, which are important quantities in fluid and solid mechanics and material sciences.

Transition points are stable features in 3D tensor fields but rarely appear in 2D tensor fields. Developing a clear understanding of the physical meaning of wedges, trisectors, and transition points in terms of their physical interpretation can enable wider impacts of tensor field topology in many applications of tensor fields in science and engineering.

4.3 Index Theory and Topological Editing

A 2D degenerate point can be measured by its tensor index, which can be defined in a fashion similar to the Poincaré index for vector field singularities. Such an index has been extremely useful for 2D tensor field analysis and simplification. Recall that the total tensor index of all the degenerate points is a constant, i.e., the Euler characteristic of the underlying domain. Degenerate point removal must involve two degenerate points, with opposite tensor indexes, so that the resulting tensor field still has the same total tensor index.

How to measure a 3D degenerate point? Would it make more sense to define the index for a degenerate curve as a whole? This is one of the most fundamental problems in advancing the state of 3D tensor field visualization.

What are the fundamental operations for simplifying the topology of a tensor field? In other words, what are the fundamental bifurcations in 3D tensor fields? How do we realize these operations? The aforementioned tensor field indexes, if developed, can provide a critical insight on topological simplification.

4.4 Triple Degeneracy

Triple degenerate tensors are numerically unstable [14], which means that a linear degenerate curve does not intersect a planar degenerate curve in general. However, it is possible that two such curves pass through the same cell in the mesh, and their separation is small. In this case it may be necessary to study the behavior of the tensor field around the cell, which topologically is equivalent to that of a triple degenerate point.

How many degenerate curves can emanate from a triple degenerate point, and what kinds of degenerate points are on these curves (linear, planar, wedge, trisector, etc)? Answers to these questions are necessary in providing a complete picture of tensor field topology.

4.5 More Robust Degenerate Curve Extraction and Classification

Existing methods for extracting degenerate curves cannot guarantee that all degenerate curves can be found. Moreover, no algorithm has been published in locating the transition points on degenerate curves. This raises a fundamental challenge on the physical interpretation of tensor field topology in terms of the applications. Identifying a robust degenerate curve extraction and classification algorithm is highly desirable. The indexes for 3D tensor fields can help identify cases where some degenerate curves are missing from a degenerate curve extraction algorithm. A possible first direction is to study linear tensor fields given their simplicity.

4.6 3D Tensor Fields and Their 2D Projections

There are two typical scenarios in which people need to deal with 2D and 3D tensor fields simultaneously.

First, domain scientists often inspect the projection of 3D data sets on some two-dimensional submanifolds, such as a probe plane or the boundary surface of the domain [24, 25]. While it is straightforward to project a 3D tensor field onto a 2D submanifold, it is not clear how useful the analysis of the projected tensor field is for the analysis of the original 3D tensor fields. Under what conditions is such an approach beneficial, and when is it misleading? A closer inspection of this is critical to the widening use of tensor field visualization.

Second, one often needs to create a volumetric tensor field from the curvature tensor field (2D) on the boundary surface [26], especially for remeshing purposes where the volumetric mesh elements are required to respect to the features in the boundary surface. While it seems straightforward to require that the eigenvectors in the 2D tensors be also eigenvectors of the 3D tensors, it is challenging to determine which two of the three eigenvectors of the 3D tensors be used. Moreover, it is not obvious how to choose the third eigenvalue, and how this choice impacts the application. Further research is necessary.

4.7 Extension to 3D Asymmetric Tensor Fields

There has been some recent advances in the analysis of 2D asymmetric tensor fields [25,27,28]. While having complex-valued eigenvalues and eigenvectors make the situations more complicated, some similarities between 2D symmetric tensor field topology and 2D asymmetric tensor field topology have been noted, especially when the eigenvalues are complex-valued. A natural question is how much what we know about 3D symmetric tensor fields can be used to shed light on 3D asymmetric tensor fields. Exploration in this direction can represent a major step towards a complete understanding of tensor fields, whether symmetric or asymmetric. Moreover, many science and engineering applications deal with asymmetric tensors, such as the velocity gradient tensor in fluid mechanics and deformation gradient in earthquake modeling. While it is possible to convert the asymmetric tensor field into a symmetric one by multiplying with its transpose or simply removing the anti-symmetric component, such an approach can cause information loss. Direct analysis and visualization of 3D asymmetric tensor fields is highly desirable.

Conclusion
Tensor fields are finding applications in more and more domains in science, engineering, and medicine. While our understanding of 2D tensor fields has matured, our understanding of 3D tensor fields has not. A fundamental challenge for more research in tensor fields is simply its mathematical complexity, which usually presents a rather steep learning curve for an average computational scientist conducting visualization research. Many facts

(continued)

of tensor fields are scattered in tens, if not hundreds, of research papers. Extracting a big-picture view of tensor field research can be daunting. In this chapter, the authors attempt to summarize some of what we believe to be most important facts about tensor fields, organized in an order that we believe is intuitive. We also point out, based on our own research experience, some immediate and long-term future research directions in tensor field visualization. We hope that our effort will help advance the state of this field and bring the tensor field visualization closer to realizing its full potential in domain applications.

References

1. Kratz, A., Auer, C., Stommel, M., Hotz, I.: Comput. Graph. Forum **32**(1), 49 (2013). http://dblp.uni-trier.de/db/journals/cgf/cgf32.html#KratzASH13
2. Basser, P.J., Pierpaoli, C.: J. Magn. Reson. Ser. B **111**(3), 209 (1996)
3. Westin, C.F., Peled, S., Gudbjartsson, H., Kikinis, R., Jolesz, F.A.: In: ISMRM '97, p. 1742, Vancouver, (1997)
4. Cammoun, L., Castano-Moraga, C.A., Munoz-Moreno, E., Sosa-Cabrera, D., Acar, B., Rodriguez-Florido, M., Brun, A., Knutsson, H., Thiran, J., Aja-Fernandez, S., de Luis Garcia, R., Tao, D., Li, X.: Tensors in Image Processing and Computer Vision. Advances in Pattern Recognition. Springer, London, 2009. http://www.springer.com/computer/computer+imaging/book/978-1-84882-298-6
5. Zhang, S., Kindlmann, G., Laidlaw, D.H.: Visualization Handbook Academic Press, Orlando, (2004). http://www.cs.brown.edu/research/vis/docs/pdf/Zhang-2004-DTM.pdf
6. Delmarcelle, T., Hesselink, L.: IEEE Comput. Graph. Appl. **13**(4), 25 (1993)
7. Zheng, X., Pang, A.: In: Proceeding IEEE Visualization, pp. 249–256 (2003)
8. Cabral, B., Leedom, L.C.: In: Poceedings of ACM SIGGRAPH 1993. Annual Conference Series, pp. 263–272 (1993)
9. Zheng, X., Pang, A.: In: In: IEEE Visualization, pp. 379–386 (2002)
10. Neeman, A., Jeremic, B., Pang, A.: In: IEEE Visualization, p. 5 (2005)
11. Schultz, T., Kindlmann, G.L.: IEEE Trans. Vis. Comput. Graph. **16**(6), 1595 (2010)
12. Delmarcelle, T., Hesselink, L.: In: Proceedings IEEE Visualization '94 (1994)
13. Hesselink, L., Levy, Y., Lavin, Y.: IEEE Trans. Vis. Comput. Graph. **3**(1), 1 (1997)
14. Zheng, X., Pang, A.: In: Proceedings IEEE Visualization 2004, VIS '04, pp. 313–320. IEEE Computer Society, Washington (2004). doi:10.1109/VISUAL.2004.105. http://dx.doi.org/10.1109/VISUAL.2004.105
15. Zheng, X., Parlett, B.N., Pang, A.: IEEE Trans. Vis. Comput. Graph. **11**(4), 395 (2005)
16. Zheng, X., Parlett, B., Pang, A.: In: Proceedings IEEE Visualization 2005, pp. 551–558 (2005)
17. Tricoche, X., Kindlmann, G., Westin, C.F.: IEEE Trans. Vis. Comput. Graph. **14**(6), 1627 (2008). http://doi.ieeecomputersociety.org/10.1109/TVCG.2008.148
18. Tricoche, X., Scheuermann, G.: In: Geometric Modeling Methods in Scientific Visualization (2003)
19. Zhang, E., Hays, J., Turk, G.: IEEE Trans. Vis. Comput. Graph. **13**(1), 94 (2007)
20. Zhang, E., Mischaikow, K., Turk, G.: ACM Trans. Graph. **25**(4), 1294 (2006)

21. Wagner, C., Garth, C., Hagen, H.: In: Laidlaw, D.H., Vilanova, A. (eds.) New Developments in the Visualization and Processing of Tensor Fields. Mathematics and Visualization, pp. 363–379. Springer, Berlin (2012). doi:10.1007/978-3-642-27343-8-19. http://dx.doi.org/10.1007/978-3-642-27343-8-19
22. Nieser, M., Palacios, J., Polthier, K., Zhang, E.: IEEE Trans. Vis. Comput. Graph. **18**(6), 865 (2012). doi:10.1109/TVCG.2011.118. http://dx.doi.org/10.1109/TVCG.2011.118
23. Delmarcelle, T., Hesselink, L.: In: IEEE Computer Graphics and Applications, pp. 140–147 (1994)
24. Chen, G., Mischaikow, K., Laramee, R.S., Pilarczyk, P., Zhang, E.: IEEE Trans. Vis. Comput. Graph. **13**(4), 769 (2007)
25. Chen, G., Palke, D., Lin, Z., Yeh, H., Vincent, P., Laramee, R., Zhang, E.: IEEE Trans. Vis. Comput. Graph. **17**(12), 1979 (2011). doi:10.1109/TVCG.2011.170. http://dx.doi.org/10.1109/TVCG.2011.170
26. Takayama, K., Okabe, M., Ijiri, T., Igarashi, T.: ACM Trans. Graph. **27**(3), 53:1 (2008). doi:10.1145/1360612.1360652. http://doi.acm.org/10.1145/1360612.1360652
27. Zheng, X., Pang, A.: In: IEEE Proceedings on Visualization, pp. 3–10 (2005)
28. Zhang, E., Yeh, H., Lin, Z., Laramee, R.S.: IEEE Trans. Vis. Comput. Graph. **15**(1), 106 (2009)

A Comparison of Pareto Sets and Jacobi Sets

Lars Huettenberger and Christoph Garth

1 Introduction

The amount of data arising from computational or experimental treatment of scientific problems has grown strongly over the past decades. Not only have typical data sets increased in size, but also with respect to variety. For example, ensemble simulations, i.e. repeatedly conduct simulations with slightly varying parameters, or using different methods on the same problem to account for uncertainties and noise, has been a trend of the recent past. Considering the outcome of this process as a multifield data set, we observe that the image dimensionality has grown, too.

To provide visual analysis capabilities in such settings, several classes of techniques were explored in recent research, including projection approaches, hierarchical displaying or the usage of glyphs and icons [9]. However, due to the overwhelming size and complexity of data in many application areas, those visualizations alone are often unsatisfactory and would require further improvement. To offer a simplified representation, a proven approach has been to define important features and structures in the data on which the visualization can focus.

Topological structures are one possibility to characterize such features. Among other approaches, previous work on topological analysis of multifield data introduced *Jacobi sets* [11], *Morse decomposition* [24], and *Joint Contour Nets* [7] (JCNs). Those methods define their structures and criticality based on ideas of topological agreement such that the interpretation of the individual functions, for example with respect to extremity, can be transferred to the multifield data. To identify the common idea in these concepts and, e.g. use it to improve the current visualization tools or to decide under which scenario which tool is to be preferred,

L. Huettenberger • C. Garth (✉)
University of Kaiserslautern, Kaiserslautern, Germany
e-mail: l_huette@cs.uni-kl.de; garth@cs.uni-kl.de

those methods need to be put into context with each other. Previous work on Pareto sets [14] included an informal comparison to earlier topological concepts and highlighted differences to the other methods above as a first step.

In this context, the goal of this paper is to examine deeper connections between Pareto sets [14] and Jacobi sets [11]. Specifically, we prove a subset relation between these sets of critical points defined by both theories in the smooth case. Our result indicated further relations between both sets and other multifield approaches specifically the Morse decomposition and the JCN methods.

The paper is structured as follows: After we provide a brief discussion of prior work in the following section, we first recapture the fundamentals of Pareto sets (Sect. 3) and Jacobi sets (Sect. 4) and provide an alternative characterization for each set. We use these new definitions to outline a proof idea in the first part of Sect. 5 and carry out this idea in its second half. The implication of the proof results for future work are presented in Sect. 6 as well as a visual comparison of Pareto and Jacobi sets using two data sets as a test set.

2 Related Work

The following paragraphs provide background and discuss the wider context of multifield visualization into which our focus on topological structures falls. Certainly not all visualizations of multifield data are based on topological descriptions. A general, non-topological approach is to present the modalities in multiple linked but separate views [29]. Such techniques, however, do not include interaction between the views explicitly but require the user to explore the images on their own.

To guide the user through the data many different approaches are possible going from user-supported selection and merge of subsets or subvolumes to the definition of similarity measures.

Examples for the first area are Botchen et al. [2] or Nagaraj and Natarajan [19] who used fuzzy logic and density functions, respectively, to guide isosurface selection, or Woodring and Shen [30] who used boolean set operations to merge several volumes. The second area, similarity measures, can be used to build a graph-like structure to aid user-traversal through the separate modalities. Those measures include global correlation (see e.g. Sauber et al. [22]) or coherence (Nagaraj et al. [20]). Without such similarity definitions, Kniss et al. [16] proposed the usage of multidimensional transfer functions to add connections between the different views.

All these approaches aid users mainly to select and traverse through the data, but they provide only few hints on which data to select or where to go. Finding important structures inside the data is still left to the user. Topological approaches try to fill this gap.

Coming from the user-interactivity point of view, different papers introduced the idea of using topological representations to aid data traversal. For example, contour trees were applied for isosurfaces and volume rendering [8, 26] and Jacobi sets and Reeb graphs were used track volumetric features and quantities over time [4, 5, 27].

Coming from the mathematical side, topological approaches were introduced to provide an abstract simpler view on the data for the user which still contains all relevant features. For multiple scalar fields, such methods include Jacobi sets [11] which work with the intersection and restriction of level sets and are based on folding of smooth mappings by Whitney [28]. Stadler and Flame's work [23] on a mathematical approach to optimization problems on graphs inspired Huettenberger et al. [14], who use partial ordering to define dominance relations on multifield data.

Through the calculation of steepest gradients most practical scalar functions can be transformed into a vector field where for each location the scalar value is replaced by the gradient vector at that point. Vector field topology is often used in flow simulation and visualization. We refer to surveys like Laramee et al. [18] or Pobitzer et al. [21] for more information on this direction. Finding stationary points or orbits in vector field can be achieved by following trajectories as described by Helman and Hesselink [13]. Alternatively these features can be identified through the intersection of stream surfaces as done by Theisel and Weinkauf [25]. In prior work, using topological graphs, Chen at al. [10] introduced Morse Decomposition on single vector fields which were then extended by Szymczak to multiple vector fields incorporating super transition graphs [24].

For completeness, the following references present approaches for subset selection and merging as well as view abstraction that are not based on topological approaches or view points. For a general overview of multivariate and multifield visualization techniques see Bürger and Hauser [6]. Further relevant work includes Akiba and Ma [1] who suggested to couple information visualization with volume rendering to visualize time-dependent, multiple scalar fields. In another combination of information and scientific visualization techniques, Janicke et al. [15] used brushing and linking of a graph structure called attribute cloud to visualize multifield data. See Lampe et al. [17] for an approach to transform volumes into curves in a neighborhood-preserving fashion such that the curves can be laid out and analyzed in parallel. For an evolutionary approach, Fuchs et al. [12] used an evolutionary search algorithm to assist hypothesis formalization in multiple volumetric data.

3 Pareto Sets

In this section, we first present the definition of Pareto sets aimed to construct topological structures—specifically the definition of critical points and regions—in a multifield setting. In the second half of the section, we proceed to an alternative characterization of these critical points through point-specific linear programs. We then prove that a point is in the Pareto set if and only if the corresponding linear program has no solution, i.e. is infeasible.

3.1 Pareto Sets of Multiple Morse Functions

The following concepts are adapted from an approach by Stadler and Flamm [23] who described topological structures on graphs. We follow the presentation in [14] and extend it from a piecewise linear, simplicial setting to a smooth case.

Henceforth, we consider $\mathbb{M} \subset \mathbb{R}^d$ to be a d-manifold, $d > 0$, and f a function $f : \mathbb{M} \mapsto \mathbb{R}^k$, $k > 0$, such that $f = (f_1, \ldots, f_k)$ consists of k Morse functions $f_i : \mathbb{M} \mapsto \mathbb{R}$.

To compare two different points x and y in $\mathbb{M}$ with respect to their values in f we say that x *(weakly) dominates* y if $f_i(x) \geq f_i(y)$ for all $i = 1, \ldots, k$, and denote this as $x \succeq y$. Furthermore, x *(strictly) dominates* y, written as $x \succ y$, if for each index i the stronger condition $f_i(x) > f_i(y)$ holds. The points x and y are called *comparable* if either $x \succ y$ or $y \succ x$ holds, and *incomparable* otherwise. Intuitively, comparability is based on the notion that all functions f_i change in the same way—all either increase or decrease—when changing from x to y.

The definition of "$\succ$" provides a transitive relation, i.e. $x \succ y$ and $y \succ z$ imply $x \succ z$. Together with reflexivity ($x \succeq x$) and antisymmetry ($a \succeq b$ and $b \succeq a$ imply $a = b$) we receive a partial ordering of points w.r.t. their values in the image of f.

Through this dominance relation we can now describe *extremal* points of f. A point x is called a *local Pareto optimum* if there exists an open neighborhood $U(x)$ containing x such that all $y \in U(x) \setminus \{x\}$ are incomparable. Similarly, x is a *local Pareto minimum* or *local Pareto maximum*, if, for all $y \in U(x)$, y is either incomparable to x or strictly dominates or is strictly dominated by x, respectively. All other points are termed *regular*. In the remainder of this paper, we will drop the *local* attribute as we are not interested in global perspective, and refer to Pareto optima, Pareto minima, and Pareto maxima collectively as *Pareto extrema*, $\mathbb{P}$. It should be noted that in the unifield case ($k = 1$), minima and maxima are Pareto minima/maxima according to this definition. The definitions are illustrated for $k = 2$ functions in Fig. 1.

Given a point $x \in \mathbb{M}$ and an open neighborhood σ with $x \in \sigma$, we define a set of all points $y \in \sigma$ which are strictly dominated by x, i.e. for which $x \succ y$

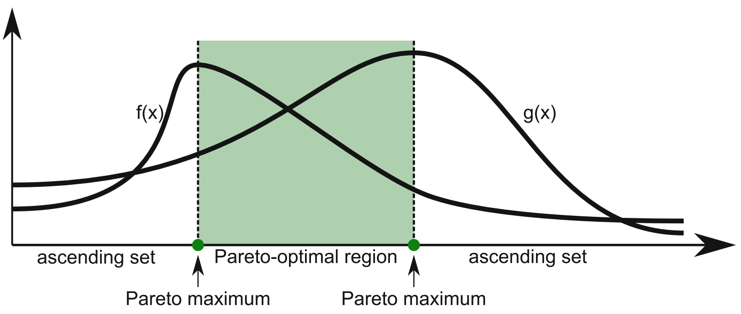


Fig. 1 Pareto sets for two one-dimensional functions

holds. For a sufficiently small neighborhood, this set can be characterized through the gradients ∇f_i for each function f_i. To see this, note the linear approximation to f_i, $f_i(y) \approx f(x) + \nabla(y-x)^T \cdot f_i(x)$ for x close to y and $\nabla f_i(x)$ the gradient of f_i computed at x. In other words, for two points $x, y \in \sigma$, we can approximate $f_i(x) > f_i(y)$ if y is contained in the set

$$H^+_{i,\sigma}(x) := \{\, y \in \sigma \mid (y-x)^T \cdot \nabla f_i(x) > 0 \,\}.$$

Considering all f_i, x is dominated by the *ascending set*

$$H^+_\sigma(x) := \bigcap_{i=1}^{k} H^+_{i,\sigma}(x).$$

The *descending set* $H^-_\sigma(x)$ is defined analogously, such that $H^\pm_\sigma(x) := H^+_\sigma(x) \cup H^-_\sigma(x)$ is the set of all points in σ that are comparable to x.

Therefore, $H^+_\sigma(x)$ provides a characterization of Pareto maxima that we denote with the new symbol $\mathbb{P}^+$ such that

$$\mathbb{P}^+ := \{x \in \mathbb{M} \mid H^+_\sigma(x) = \emptyset\}$$

The set of all Pareto minima $\mathbb{P}^-$ is given analogously, together with $\mathbb{P} = \mathbb{P}^+ \cup \mathbb{P}^-$, the set of all Pareto extrema or in general the *Pareto Set*.

For further reading, especially to understand the global structure defined through connectivity between the Pareto extrema, we refer to [14].

Note that for a small neighborhood σ around a point x, $H^+_\sigma(x)$ is a cone limited by $\{y \in \sigma \mid \exists_i (y-x) \cdot \nabla f_i(y) = 0\}$. Due to the use of the strict dominance this border is excluded from $H^+_\sigma(x)$.

In the special case that, for some $1 \le i < j \le k$, $\nabla f_i = -\nabla f_j$ holds at least in σ, $H^+_\sigma(x)$ is empty while $\{y \in \sigma \mid \exists_i (y-x)^T \cdot \nabla f_i(y) = 0\} = \{x\}$ must not be true, especially for $k = 2$. Conversely, however, if we require $\nabla f_i \neq -\nabla f_j$ for every pair of functions, enforced e.g. through a preprocessing step, we can replace the strict dominance relation in the above definition with the weak dominance relation without consequences for the following theorem. However, we keep the strict variant so we do not need any more restriction on the functions. This also makes the proofs for the theorem easier to follow.

3.2 A Linear Program for Pareto Extrema

Towards the main result of this paper, it is useful to characterize points of the Pareto set in terms of feasibility of a specific linear optimization problem. This allows us to apply mathematical theorems from linear programming for the proofs in Sect. 5.

To identify the Pareto set, we calculate an intersection of halfspaces characterized by the gradients of f; hence it is reasonable that the optimization problem should be based on a suitable description of the latter. Note that for each point $x \in \mathbb{M}$ those gradients are only related to a sufficiently small neighborhood $U(x)$. Hence, for each point x, each Morse function f_i can be locally approximated by a linear function g_i with the only condition that $g_i(x) = f_i(x)$ and $\nabla g_i(x) = \nabla f_i(x)$. With this function g_i, $H^+_{i,\sigma}(x) = \emptyset$ is equivalent to $H^+_{i,\mathbb{M}}(x) = \emptyset$ when f_i is substituted by g_i. Note that g_i is different for each x and not a global approximation of f_i. For easier reading we drop the σ in $H^+_\sigma(x)$ in the following theorem.

Theorem 1 *Let $\varepsilon > 0$. Given a point $p \in \mathbb{M}$, and denote $n_i := \nabla f_i(p)$.*
Define a $(k+1) \times (d+1)$-matrix

$$A_p := \begin{bmatrix} -n_{1,1} & \cdots & -n_{1,d} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ -n_{k,1} & \cdots & -n_{k,d} & 1 \\ 0 & \cdots & 0 & -1 \end{bmatrix}$$

and vectors

$$b = (-\varepsilon, \ldots, -\varepsilon, 0)^T \in \mathbb{R}^{k+1}$$
$$c = (0, \ldots, 0, 1)^T \quad \in \mathbb{R}^{d+1}.$$

Then $H^+(p) \neq \emptyset$ if and only if the linear program $\mathscr{P}^+_p$ defined by

$$\begin{aligned} &\textit{maximize} && c^T x \\ &\textit{subject to} && A_p x \leq b \end{aligned} \tag{1}$$

is feasible and unbounded. Feasible means that an $x \in \mathbb{R}^{d+1}$ exists with $A_p x \leq b$ holds and unbound means that for every x with $A_p x \leq b$ we can find an $x' \neq x$ with $A_p x' \leq b$ such that $c^T x < c^T x'$.

Proof Without loss of generality assume that $p = 0$ by translation such that through the introduction of $\varepsilon > 0$, p is not a solution. Otherwise, without ε, i.e. with $b = \mathbf{0}$ the zero vector, the linear problem would always be feasible with $x = \mathbf{0}$ as a solution.

We prove the two directions of the equivalence claim separately. For $\Rightarrow$, assume that $H^+(p) \neq \emptyset$ such that $y \in H^+(\mathbf{0})$ with $y \neq 0$ exists. Then, based on the definition of $H^+(\mathbf{0})$, $(y - \mathbf{0})^T \cdot n_i > 0$ for all $i = 1, \ldots, k$. Hence, $\varepsilon_i > 0$ exists such that $n_i^T \cdot y \geq \varepsilon_i$ for all $i = 1, \ldots, k$. Thus, for $\varepsilon_{\min} = \min_{i=1,\ldots,k} \varepsilon_i$ and a $\gamma_0 \geq \frac{\varepsilon}{\varepsilon_{\min}} > 0$,

$$n_i^T \cdot y \cdot \gamma \geq \varepsilon_{\min} \cdot \gamma \geq \varepsilon \text{ holds for all } i = 1, \ldots, k \text{ and any } \gamma \geq \gamma_0.$$

Note that we introduced an arbitrary γ to receive an infinite amount of solutions for $\mathscr{P}_p^+$. To find those solutions, for any $\gamma \geq \gamma_0$, we define a $q \geq 0$ such that the following inequality holds for all $i = 1, \ldots, k$.

$$(n_i^T \cdot y \cdot \gamma) - q \geq \varepsilon$$

This is true as long as q lies within the difference of $n_i^T \cdot y \cdot \gamma$ and ε, i.e.

$$0 \leq q \leq (y \cdot n_i^T \cdot \gamma) - \varepsilon, \text{for all } i = 1, \ldots, k.$$

Hence, for $q := \min_{i=1,\ldots,k}(n_i^T \cdot y \cdot \gamma) - \varepsilon$, it is easily computed that for $y' = (y_0, \ldots, y_k, q)$

$$A_p \cdot y' \leq b \text{ hold.}$$

To see this, consider the calculation of line i, $1 \leq i \leq k$, with $A_{p,i} \cdot y' \leq -\varepsilon$. Given the definition of A_p this calculation is equivalent to $-n_i^T \cdot y + q \leq -\varepsilon$ and thus, is simply an inversion to our above argumentation, everything is multiplied by -1. Due to the inversion the parameter q changes into $-q \leq 0$ such that the last line, $A_{p,k+1} \cdot y' \leq 0$ is also ensured. Hence $\mathscr{P}^+$ is feasible. Furthermore, with $c^T y' = q = \gamma \cdot \min_{i=1,\ldots,k}(n_i^T \cdot y) - \varepsilon$ and $\gamma \geq \gamma_0 > 0$ not bounded from above, the program is also unbounded.
Conversely for $\Leftarrow$, let $y \in \mathbb{R}^{k+1}$ be a solution to the linear program $\mathscr{P}_p^+$. Define $y' = (y_0, \ldots, y_k)$ such that, based on the definition of $\mathscr{P}_p^+$,

$$-n_i^T \cdot y' + y_{k+1} \leq -\varepsilon \text{ and therefore } n_i^T \cdot y' \geq \varepsilon + y_{k+1} \text{ hold.}$$

Hence, with $A_{p,k+1} \cdot y = -1 \cdot y_{k+1} \leq 0$, we find that

$$n_i^T \cdot y' > 0$$

for all $i = 1, \ldots, k$, such that $y' \neq \mathbf{0}$ is in $H^+(\mathbf{0})$ by definition of the n_i.

□

The linear program $\mathscr{P}_p^-$ for which the gradient vectors are negated is defined analogously, and its relation to $H^-(p)$ is proven in the same way. Also note that, as a direct implication of Theorem 1, $p \in \mathbb{M}$ is not Pareto maximal, i.e. $p \notin \mathbb{P}^+$ if and only if $\mathscr{P}_p^+$ is feasible and unbound.

4 Jacobi Sets

In the following, we briefly revisit the concepts underlying the Jacobi sets. A more thorough treatment can be found in [11].

4.1 Background

The Jacobi set of $k \leq d$ Morse functions f_i on a common d-manifold $\mathbb{M}$ is defined through level set intersection. The *level set* for one function f_i, $1 \leq i \leq k$, and scalar value $t \in \mathbb{R}$ is the set $\mathbb{M}_t := f_i^{-1}(t)$. For multiple functions, the intersection of the level sets, excluding function f_l, is defined by $\mathbb{M}_{t,l} := \bigcap_{i \neq l} f_i^{-1}(t_i)$ for a vector $t \in \mathbb{R}^k$. The Jacobi set $\mathbb{J}$ is the closed set of all point x that are critical in f_l restricted on $\mathbb{M}_{f(x),l}$ for some index $1 \leq l \leq k$ with respect to scalar field topology.

Note that a level set in a d-manifold is of dimension $d - 1$ and the closed intersection of $k - 1$ level sets is of dimension $d - (k - 1)$.

Conversely, we have $x \in \mathbb{J}$ if and only if after removing all points from $\mathbb{M}$ with different values than x with respect to $k - 1$ of the Morse function, x is critical w.r.t. the (k)th function, thus a minimum, maximum or saddle of some degree. Note that x can become isolated in $\mathbb{M}_{f(x),l}$ which we then consider as critical too. In case of $k > d$, this results in $\mathbb{J} = \mathbb{M}$.

In another approach towards Jacobi sets, presented by Edelsbrunner [11], criticality of a point $x \in \mathbb{M}$ is defined through the gradients of the functions f_i at x. Consider the gradients ∇f_i of the function $i \neq l$ at a point $x \in \mathbb{M}_t$ for some index $1 \leq l \leq k$ and a k-vector t. Those gradients span the linear subspace of vector normals of $\mathbb{M}_t$. If ∇f_l at point x also belongs to this linear subspace, x is critical in f_l restricted on $\mathbb{M}_{t,l}$.

In general, Jacobi sets can be described as the closure of a set of points x for which there is $\lambda \in \mathbb{R}^k$ such that $\lambda \neq 0$ and $\sum_{i=1}^{k} \lambda_i \cdot \nabla f_i(x) = \mathbf{0}$. Hence,

$$\mathbb{J} := cl \left\{ x \in \mathbb{M} \mid \exists \lambda \in \mathbb{R}^k \setminus \{0\} \text{ s.t. } \sum_{i=1}^{k} \lambda_i \nabla \cdot f_i(x) = \mathbf{0} \right\}$$

Note that in this linear combination negative parameters λ_i are allowed, thus the direction of the gradient is ignored.

For further use we add the notation of the *positive Jacobi Set* $\mathbb{J}^+$ that includes the additional restriction that $\lambda_i \geq 0$ for all $1 \leq i \leq k$ and thereby results in a positive linear combination.

$$\mathbb{J}^+ := \left\{ x \in \mathbb{M} \mid \exists \lambda \in \mathbb{R}^k \setminus \{\mathbf{0}\} \text{ s.t. } \sum_{i=1}^{k} \lambda_i \cdot \nabla f_i(x) = \mathbf{0} \text{ and } \lambda_i \geq 0 \text{ f.a. } i \right\} \tag{2}$$

We obtain $\mathbb{J}^+ \subseteq \mathbb{J}$ as a subset of the Jacobi set. In an analogous fashion, we also add the notation of negative Jacobi sets $\mathbb{J}^-$. Note that both sets in our definition of $\mathbb{J}^+$ and $\mathbb{J}^-$ do not include the closure. We do this to avoid complications in the following proofs with points that lie in the closure but not in the actual set.

Furthermore, note how the alternative definition is based on the gradient vectors rather than on the values of the Morse functions. Thus, the translation of this definition for multiple scalar fields to multiple vector fields is immediate if each vector field is equivalent to a gradient fields of some Morse function.

For the following Eq. (3) we also note its relation to the definition of critical triangles by Szymczak [24], defined for k piecewise constant vector fields over a 2-dimensional manifold. Szymczak calls a triangle critical iff the convex hull of its vectors—inside a triangle the separate vector fields are constant—contains the zero vector. We present Definition 3 and its relation to Jacobi sets to use the results in our main proof and, additionally, to put Pareto sets transitively in relation with Szymczak's definition.

$$\mathbb{C}^+ := \left\{ x \in \mathbb{M} \mid \exists \lambda \in \mathbb{R}^k \text{ s.t. } \sum_{i=1}^{k} \lambda_i \cdot v_i(x) = \mathbf{0} \wedge \sum_{i=1}^{k} \lambda_i = 1 \wedge \lambda_i \geq 0 \text{ f.a. } i \right\} \tag{3}$$

Claim Under the assumption that for every $x \in \mathbb{M}$, $v_i(x) = \nabla f_i(x)$ with f_i the Morse functions as given above it holds that $\mathbb{C}^+ = \mathbb{J}^+$.

Proof $\subseteq$: Obviously, if a convex combination with parameter λ results in the zero-vector, those parameters are also a positive linear combination that result in the zero-vector. Furthermore, $\sum \lambda_i = 1 \wedge \lambda_i \geq 0$ f.a. i excludes $\lambda = 0$ as parameter.
$\supseteq$: If the positive linear combination with parameter λ results in the zero-vector, we can conclude that $\sum \lambda_i = c \neq 0$ holds. The right inequality is implied by the requirement that $\lambda \neq \mathbf{0}$ and $\lambda_i \geq 0$ for $0 \leq i \leq k$. Hence, with the new parameter $\lambda' = \lambda / c$ we receive $\sum \lambda_i = 1$ and the convex combination still results in the zero-vector.

□

4.2 A Linear Program for Jacobi Critical Points

As with the Pareto sets, we aim to characterize points in the Jacobi sets through the feasibility of a linear program. This translation allows us to use mathematical results from linear program in the proof in Sect. 6 and furthermore provides a new approach to compute Jacobi sets.

Theorem 2 *Let again $\varepsilon > 0$, $p \in \mathbb{M}$, and $n_i = \nabla f_i(p)$ to define a $(d+1)\times(k+1)$-matrix*

$$B_p := \begin{bmatrix} -n_{1,1} & \cdots & -n_{d,1} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ -n_{1,d} & \cdots & -n_{m,d} & 0 \\ 1 & \cdots & 1 & -1 \end{bmatrix}$$

and vectors

$$b = (-\varepsilon, \dots, -\varepsilon, 0)^T \in R^{k+1}$$
$$c = (0, \dots, 0, 1)^T \quad \in R^{d+1}$$

Then $p \in \mathbb{J}^+$ *if and only if the linear program* $\mathcal{D}_p^+$

$$\begin{aligned} \textit{minimize} \quad & b^T y \\ \textit{subject to} \quad & B_p\, y \;=\; c \\ \textit{and} \quad & y_i \geq 0 \end{aligned} \tag{4}$$

is feasible and unbounded.

Proof For $\Rightarrow$, let without loss of generality $p = \mathbf{0}$ by translation and assume that $p \in \mathbb{J}^+$. By Eq. (2) and the proven equality $\mathbb{J}^+ = \mathbb{C}^+$ there is a $\lambda \in \mathbb{R}^d$ such that

$$\sum_{i=1}^{k} \lambda_i \cdot n_i \;=\; \mathbf{0} \quad \text{with} \quad \lambda_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{k} \lambda_i = 1$$

holds. Thus, we define a possible solution for $\mathcal{D}_p^+$ with arbitrary $\alpha > 0$.

$$y := (\alpha\lambda_1, \dots, \alpha\lambda_d, (\alpha - 1))^T$$

Since $\sum_{i=1}^{k} \lambda_i n_i = \mathbf{0}$ holds, also $\sum_{i=1}^{k} \lambda_i \alpha n_i = 0$ is true such that the first d rows of $B_p y = c$ are true. With

$$\left(\sum_{i=1}^{k} y_i\right) - y_{(k+1)} = \left(\sum_{i=1}^{k} \lambda_i\right)\alpha - (\alpha - 1) = 1$$

all conditions of $\mathcal{D}_p^+$ are fulfilled by y which is therefore feasible. Furthermore, since

$$b^T y \;=\; -\varepsilon \cdot \alpha \cdot \sum_{i=1}^{k} \lambda_i \;=\; -\varepsilon \cdot \alpha$$

can be arbitrary small with respect to α, $\mathcal{D}_p^+$ is also unbounded.
Conversely for $\Leftarrow$, assume that $\mathcal{D}_p^+$ is feasible with $y \in \mathbb{R}^{k+1}$ as a possible solution. First, define $\alpha := y_{(k+1)} + 1$ and parameter vector $\lambda \in \mathbb{R}^k$ with

$$\lambda_i = y_i/\alpha \;\; \text{for all} \;\; 1 \leq i \leq k.$$

The problem condition $y_i \geq 0$ holds for all i especially $i = k + 1$ which implies $\lambda_i \geq 0$ for all i. This and the last row of the equation system $B_p \cdot y = c$, namely $\sum_{i=0}^{k} y_i - y_{k+1} = 1$, furthermore imply that $\sum_{i=0}^{k} \lambda_i = 1$ holds. Also, based on the first d rows of B_p,

$$\left(\sum_{i=1}^{k} n_i \cdot y_i = \mathbf{0}\right) \Rightarrow \left(\sum_{i=1}^{k} n_i \cdot \lambda_i = \mathbf{0}\right) \text{ holds.}$$

Hence, with λ as parameters, $p \in \mathbb{J}^+$ according to Eq. (2). □

As before, we can define a linear program $\mathscr{D}_p^-$ for which an analogous statement is easily proven, namely that $p \in \mathbb{J}^-$ if and only if $\mathscr{D}_p^-$ is feasible and unbounded.

5 Result

Having characterized both Pareto extrema and Jacobi critical points in terms of the feasibility of linear programs, we proceed to state the main result. The proof is based on the *weak duality* theorem for a *primal-dual* system [3]. The following section will briefly reiterate this statement and its preconditions.

Given a linear program in *primal* form

$$\begin{aligned} &\text{maximize } c^T x \\ &\text{subject to } Ax \leq b \end{aligned}$$

and a corresponding problem in *dual* form

$$\begin{aligned} \text{minimize} &\quad b^T y \\ \text{subject to} &\quad A^T y = c \\ \text{and} &\quad y \geq 0 \end{aligned}$$

for which the matrix A and the vectors b and c are the same, linear programming theory states the following result.

Theorem 3 (Weak Duality)

- *The primal program is unbounded* $\Leftrightarrow$ *the dual program is infeasible*
- *The dual program is unbounded* $\Leftrightarrow$ *the primal program is infeasible.*

The complete proof can be found in [3] and in many other texts on linear optimization. We proceed to use the previous results from Sects. 3.2 and 4.2 to easily prove the following result. We are able to do so since $\mathscr{P}_p^+$, $\mathscr{P}_p^-$, $\mathscr{D}_p^+$, and $\mathscr{D}_p^-$ are all in primal and dual form, respectively.

Theorem 4 *Given a d-manifold $\mathbb{M} \subseteq \mathbb{R}^d$ and k Morse functions $f_i : \mathbb{M} \mapsto \mathbb{R}$ with $k \leq d$, we obtain the following relations:*

$$\begin{aligned} &\textit{(i)} && \mathbb{P}^+ = \mathbb{J}^+ \\ &\textit{(ii)} && \mathbb{P}^- = \mathbb{J}^- \\ &\textit{(iii)} && \mathbb{P} \subseteq \mathbb{J} \end{aligned}$$

Proof (i). Given a point $p \notin \mathbb{P}^+$ we know that, by definition of $\mathbb{P}^+$, $H^+(p) \neq \emptyset$. Therefore, using Theorem 1, the linear program $\mathscr{P}_p^+$ is feasible and unbounded. Since $\mathscr{P}_p^+$ is in primal form, weak duality provides that the corresponding dual problem $\mathscr{D}_p^+$ is infeasible, hence neither feasible nor unbound. Thus, Theorem 2 implies that $p \notin \mathbb{J}^+$ and therefore $\mathbb{J}^+ \subseteq \mathbb{P}^+$.

The reverse statement, $\mathbb{P}^+ \subseteq \mathbb{J}^+$, follows analogously using the second line of the weak duality theorem. Combined, we receive $\mathbb{P}^+ = \mathbb{J}^+$.

Claim (ii) follows analogously by considering $\mathscr{P}_p^-$ and $\mathscr{D}_p^-$ instead of $\mathscr{P}_p^+$ and $\mathscr{D}_p^+$.

(iii). We use (i) and (ii) for the second equality of

$$\mathbb{P} = \mathbb{P}^+ \cup \mathbb{P}^- = \mathbb{J}^+ \cup \mathbb{J}^- \subseteq \mathbb{J}.$$

□

Note that (iii) can be trivially extended to the case $k > d$ since then $\mathbb{J} = \mathbb{M}$ and therefore $\mathbb{P} \subseteq \mathbb{J}$ as a direct result. $\mathbb{J} = \mathbb{M}$ follows directly from the level set-based definition of Jacobi sets. For each point p, after the restriction of a function to the level set intersection of the other $k - 1 \geq d$ functions, only p remains which is obviously critical under such an restriction.

Examples in the following section and in previous work by Huettenberger et al. [14] furthermore show that the subset relation (iii) is strong, i.e. $\mathbb{P} \subset \mathbb{J}$ is possible.

6 Discussion and Conclusion

6.1 Discussion

The result obtained here is based on a continuous formulation and requires Morse functions over a d-manifold while the practical application would need to work on a piecewise linear setting. Numerical experiments seem to indicate that there is a similar correspondence, and its proof is the focus of ongoing work.

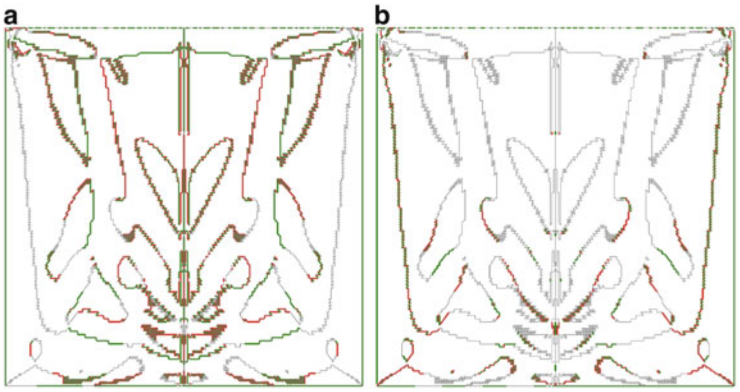


Fig. 2 Pareto sets and Jacobi sets for a two-dimensional example with two functions. In both images we used the same functions but inverted one of them for the results in the second image

We calculated Pareto and Jacobi sets for a selection of artificial and practical data sets, given on two-dimensional simplicial meshes using the algorithms described in [14] and [11]. Figures 2 and 3 provide the results from two datasets as examples.

First, in a fluid-filled can with a rotating bottom values like velocity, pressure, etc. are measured to identify locations of possible vortices. On a cross-section through the can, both the Jacobi set and the Pareto sets are calculated and presented in Fig. 2 for two of these measurements. Pareto extrema are shown in red (minima) and green (maxima), respectively, while points in the Jacobi set are colored gray. Note that $\mathbb{P}_1 \subset \mathbb{J}_1$ holds.

In Fig. 2b, one of these component functions is negated, and the results are shown using the same coloring as in Fig. 2b; here, $\mathbb{P}_2 \subset \mathbb{J}_2$. Changing the sign of the component functions, which inverts the gradient vectors, does not change the Jacobi set since the definition of $\mathbb{J}$ allows positive and negative parameter λ_i. We therefore have the equality $\mathbb{J}_1 = \mathbb{J}_2$. We restricted the orientation of the gradients through only positive or only negative parameter λ_i in the definitions of positive and negative, respectively, Jacobi sets. Since we can neglect other possible orientations based on the symmetry of Jacobi sets, it follows that for the closure $cl\{\mathbb{J}_1^+ \cup \mathbb{J}_2^+\} = \mathbb{J}_1$ holds in these images. Hence, using our previous theorem, $cl\{\mathbb{P}_1^+ \cup \mathbb{P}_2^+\} = \mathbb{J}_1$ is also true. This can obviously be extended to cases with even more underlying functions as long as $k \leq d$, which allows the calculation of the Jacobi set through the calculation of the Pareto sets.

As we consider the two-dimensional can dataset in Fig. 2, the points $p \in \mathbb{P}$ are those locations where the gradients of both functions point in parallel and in opposite directions. The points $p \in \operatorname{int} \mathbb{J} \setminus \mathbb{P}$ are the locations where the gradients point in the same direction. Hence $\mathbb{J} \setminus \mathbb{P}$ includes all points with parallel gradient vectors, a common direction over all fields of steepest ascent. Note how the Jacobi set connects the components in the Pareto set through one-dimensional paths. Hence we can interpret $\mathbb{J} \setminus \mathbb{P}$ as connections of the components in $\mathbb{P}$ which form the

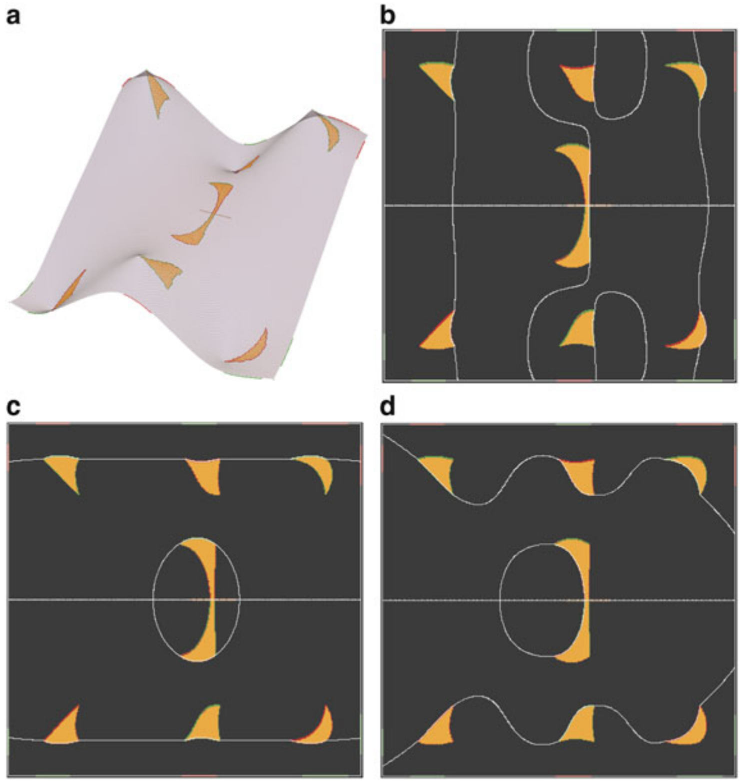


Fig. 3 Pareto sets and Jacobi sets for a two-dimensional example with three Gaussian functions. Fig. 3a illustrates the behavior of one of the three functions. The remaining images show the Jacobi sets for the three possible pairings of the three functions separately

fastest paths through the domain. Future work towards multifield visualization and topological structures should include a study of this observation to aim for incorporation of Jacobi and Pareto sets to build global connectivities.

The second data set consists of three artificial functions obtained from sums of Gaussians. Each component function has three minima and maxima. The location of the minima and maxima are only slightly perturbed among the three function, cf. Fig. 3a for an illustration of one of the functions. The Jacobi sets are calculated and colored as in the previous figure, however since Jacobi sets are only defined for $k \leq d$, we calculate them separately for each two-pairings of variables, shown in the three Fig. 3b–d. Pareto sets based on all three functions are also shown in these figures with the same coloring as in Fig. 2, but with additional yellow for Pareto optima.

Note that the Pareto sets for each pairing of two functions can be calculated through the intersection of the corresponding Jacobi sets and the Pareto set based on all three functions. Hence, it was sufficient to only calculate the Pareto set once.

Also note how the Jacobi sets from all three figures, if combined, form the border of the components in the Pareto set. Our assumption and additional direction of research is that this introduces, together with our main proof, a new approach to calculate Pareto sets for a large number of functions through the combined Jacobi sets for small number of functions.

6.2 Conclusion

This paper proved the following three subset and equivalence relations:

$$\begin{aligned} &\text{(i)} \quad \mathbb{P}^{+} = \mathbb{J}^{+} \\ &\text{(ii)} \quad \mathbb{P}^{-} = \mathbb{J}^{-} \\ &\text{(iii)} \quad \mathbb{P} \subseteq \mathbb{J} \end{aligned}$$

The previous subsection also provided the equivalence relation $cl\{\mathbb{P}_1^{+} \cup \mathbb{P}_2^{+}\} = \mathbb{J}_1$ for the case of two underlying functions. The discussed concept can however be extended to a arbitrary number of functions. This and the translation of the Jacobi and Pareto set definition to linear programs provide new approaches to calculate those sets.

New calculation tools and a better understanding of both sets and there relation to each other are only the immediate results.

This paper also suggested an equality between Jacobi sets and the critical points defined in the Morse decomposition approach [24], resulting in our subsequent result $\mathbb{J}^{+} = \mathbb{C}^{+}$. However, Pareto sets and Jacobi sets are only two possible approaches to define criticality in multifield scalar data and other methods exist as well as methods for multifield vector or tensor fields. Thus, this paper is intended as a step towards understanding the general concepts of multifield topology and their practically application, for example for simplification techniques.

References

1. Akiba, H., Ma, K.L.: A tri-space visualization interface for analyzing time-varying multivariate volume data. In: Proceedings of Eurographics/IEEE VGTC Symposium on Visualization, pp. 115–122 (2007)
2. Botchen, R., Lauser, A., Weiskopf, D., Ertl, T.: Flow feature visualization using logical operators on multivariate fields. In: International Symposium on Flow Visualization (2008)
3. Boyd, R., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2007)
4. Bremer, P.T., Bringa, E.M., Duchaineau, M.A., Gyulassy, A.G., Laney, D., Mascarenhas, A., Pascucci, V.: Topological feature extraction and tracking. J Phys. Conf. Ser. **78**, 012007 (2007)

5. Bremer, P.T., Weber, G., Pascucci, V., Day, M., Bell, J.: Analyzing and tracking burning structures in lean premixed hydrogen flames. IEEE Trans. Vis. Comput. Graph. **16**(2), 248–260 (2010)
6. Bürger, R., Hauser, H.: Visualization of multi-variate scientific data. In: EuroGraphics State of the Art Reports, pp. 117–134 (2007)
7. Carr, H., Duke, D.: Joint contour nets: Computation and properties. In: Proceedings of PacificVis (2013)
8. Carr, H., Snoeyink, J.: Path seeds and flexible isosurfaces using topology for exploratory visualization. In: VISSYM '03: Proceedings of the symposium on Data visualisation 2003, pp. 49–58 (2003)
9. Chan, W.W.Y.: A Survey on Multivariate Data Visualization. Deparment of Computer Science and Engineering, Hong Kong University of Science and Technology (2006)
10. Chen, G., Mischaikow, K., Laramee, R.S., Pilarczyk, P., Zhang, E.: Vector field editing and periodic orbit extraction using morse decomposition. IEEE Trans. Vis. Comput. Graph. **13**(4), 769–785 (2007)
11. Edelsbrunner, H., Harer, J.: Jacobi sets of multiple Morse functions. In: Foundations of Computational Mathematics, vol. 312, pp. 37–57. Cambridge University Press, Cambridge (2004)
12. Fuchs, R., Waser, J., Groller, M.E.: Visual human + machine learning. IEEE Trans. Vis. Comput. Graph. **15**(6), 1327–1334 (2009)
13. Helman, J.L., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. Computer **22**(8), 27–36 (1989)
14. Huettenberger, L., Heine, C., Carr, H., Scheuermann, G., Garth, C.: Towards multifield scalar topology based on pareto optimality. Comput. Graph. Forum **32**(3), 341–350 (2013)
15. Janicke, H., Bottinger, M., Scheuermann, G.: Brushing of attribute clouds for the visualization of multivariate data. IEEE Trans. Vis. Comput. Graph. **14**(6):1459–1466 (2008)
16. Kniss, J., Kindlmann, G., Hansen, C.D.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In: Proceedings of Visualization, pp. 255–262, 562 (2001)
17. Lampe, O.D., Correa, C., Ma, K.L., Hauser, H.: Curve-centric volume reformation for comparative visualization. IEEE Trans. Vis. Comput. Graph. **15**(6), 1235–1242 (2009)
18. Laramee, R., Hauser, H., Zhao, L., Post, F.: Topology-based flow visualization, the state of the art. In: Hauser, H., Hagen, H., Theisel, H. (eds.) Topology-Based Methods in Visualization, Mathematics and Visualization, pp 1–19. Springer, Berlin (2007)
19. Nagaraj, S., Natarajan, V.: Relation-aware isosurface extraction in multifield data. IEEE Trans. Vis. Comput. Graph. **17**(2), 182–191 (2011)
20. Nagaraj, S., Natarajan, V., Nanjundiah, R.S.: A gradient-based comparison measure for visual analysis of multifield data. Comput. Graph. Forum **30**(3), 1101–1110 (2011)
21. Pobitzer, A., Peikert, R., Fuchs, R., Schindler, B., Kuhn, A., Theisel, H., Matkovic, K., Hauser, H.: On the way towards topology-based visualization of unsteady flow: the state of the art. In: Hauser, H., Reinhard, E. (eds.) Eurographics 2010 - State of the Art Reports, Eurographics Association (2010)
22. Sauber, N., Theisel, H., Seidel, H.P.: Multifield-graphs: An approach to visualizing correlations in multifield scalar data. IEEE Trans. Vis. Comput. Graph. **12**(5), 917–924 (2006)
23. Stadler, P.F., Flamm, C.: Barrier Trees on Poset-Valued Landscapes. Genet. Program. Evolvable Mach. **4**, 7–20 (2003)
24. Szymczak, A.: Stable morse decompositions for piecewise constant vector fields on surfaces. Comput. Graph. Forum **30**(3), 851–860 (2011)
25. Theisel, H., Weinkauf, T., Hege, H.C., Seidel, H.P.: Grid-independent detection of closed stream lines in 2D vector fields. In: Proceedings of Vision, Modeling and Visualization, pp. 421–428, Stanford (2004)
26. Weber, G.H., Dillard, S.E., Carr, H., Pascucci, V., Hamann, B.: Topology-controlled volume rendering. IEEE Trans. Vis. Comput. Graph. **13**(2), 330–341 (2007)

27. Weber, G., Bremer, P.T., Day, M., Bell, J., Pascucci, V.: Feature tracking using Reeb graphs, pp. 241–253, Mathematics and Visualization (2011)
28. Whitney, H.: On singularities of mappings of Euclidean spaces: mappings of the plane into the plane. Ann. Math. **62**, 374–410 (1955)
29. Wilhelm, A.: Handbook of Data Visualization. Springer Handbooks Comp. Statistics, Springer, Berlin (2008)
30. Woodring, J., Shen, H.W.: Multi-variate, time varying, and comparative visualization with contextual cues. IEEE Trans. Vis. Comput. Graph. **12**, 909–916 (2006)

Deformations Preserving Gauss Curvature

Anne Berres, Hans Hagen, and Stefanie Hahmann

1 Introduction

In industrial surface generation, it is important to consider surfaces with minimal areas for two main reasons: these surfaces require less material than non-minimal surfaces, and they are cheaper to manufacture. Based on a prototype, a so-called masterpiece, the final product is created using small deformations to adapt a surface to the desired shape. We present a linear deformation technique preserving the total curvature of the masterpiece. In particular, we derive sufficient conditions for these linear deformations to be total curvature preserving when applied to the masterpiece. It is useful to preserve total curvature of a surface in order to minimise the amount of material needed, and to minimise bending energy [9, 15].

Efimov was the first to introduce partial differential equations as a tool to study infinitesimal bending. He gives an overview of the state of the art of infinitesimal bendings in his textbook [6]. Hagen et al. [10] visualise the momentarial rotation field that is associated with infinitesimal bending. They then use the structure of this rotation field as a tool to analyse the deformations that were generated by this bending. Hahmann et al. [11] investigate numerical aspects of discretising the deformation vector field. Ivanova and Subitov [13] examine infinitesimal bendings of surfaces of revolution and of polyhedra. Meziani [18] studies infinitesimal bending of homogeneous surfaces that have a flat point and positive curvature.

A. Berres (✉) • H. Hagen
University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern, Germany
e-mail: berres@cs.uni-kl.de; hagen@cs.uni-kl.de

S. Hahmann
Univ. Grenoble-Alpes, LJK, Inria, Grenoble France

LJK-INRIA Rhone Alpes, Innovallee, 655 av. de l'Europe, 38334 Saint Ismier Cedex, France
e-mail: Stefanie.Hahmann@inria.fr

More recent works on infinitesimal bending for curves and non-parametric surfaces have been published by L. Velimirovi et al. They study total mean curvature variation on oriented, boundary-free surfaces [22], and they visualise changes of bent curves as surfaces constructed from different stages of deformation [21]. Eigensatz et al. [8] use curvature as a tool to control surface deformation. They extend this work to allow various user-specified local restrictions on deformation [7]. Other works have addressed perturbations preserving the topological form of polyhedra [1], and deformations preserving ambient isotopy of curves [14, 16].

In this work, rather than studying total curvature changes after bending, or using curvature as a tool to deform surfaces, we employ total curvature as a tool to restrict bending and avoid large changes. We assume a rigid material that can be bent out of shape through exterior deformations but that cannot be stretched in tangent direction through interior deformations, as common in engineering [2, 4].

Section 2 gives an introduction into some fundamentals of differential geometry that our method is based on. In Sect. 3, we describe and prove our approach, rounded off by two examples in Sect. 4.

2 Fundamentals of Differential Geometry

We start by defining parametrised surfaces, tangents, derivatives, and Gauß frames. Then, we recall the definitions of the first and second fundamental forms, and finally, we discuss various well-established definitions of curvature. For more definitions, see [5].

A *parametrised C^r surface* is a C^r-differentiable mapping $X : U \to \mathbb{E}^3$ of an open domain $U \subset \mathbb{E}^2$ into the Euclidean space $\mathbb{E}^3$, whose differential $\mathrm{d}X$ is one-to-one for each $q \in U$.

Remark 1 (a) A change of variables of X is a diffeomorphism $\tau : \tilde{U} \to U$, where τ is an open domain in $\mathbb{E}^2$, such that τ's differential $\mathrm{d}\tau$ always has rank $= 2$, if the determinant of its Jacobian matrix $\det(\tau^*) > 0$ is orientation-preserving.

(b) Relationship: the change of variables defines an equivalence relation on the class of all parametrised surfaces. An equivalence class of parametrised surfaces is called a surface in $\mathbb{E}^3$.

(c) Let us denote in the following $X_u := \frac{\partial X}{\partial u}, X_w := \frac{\partial X}{\partial w}, X_{uv} := \frac{\partial^2 X}{\partial u \partial w}$ or alternatively $X_i, X_j, i, j \in \{u, w\}$. The differential $\mathrm{d}X$ is one-to-one if and only if $\frac{\partial X}{\partial u}$ and $\frac{\partial X}{\partial w}$ are linearly independent.

We can define a tangent plane which is spanned by the tangents of the surface. This tangent plane, in conjunction with the surface normal, defines a local coordinate system on the manifold.

Definition 1 (a) The tangent plane is a two-dimensional linear subspace $T_{\mathbf{u}}X$ of $\mathbb{E}^3$ generated by $\mathrm{span}\{X_u, X_w\}$, and it is called the *tangent space of X at* $\mathbf{u} = (u, w) \in U$.

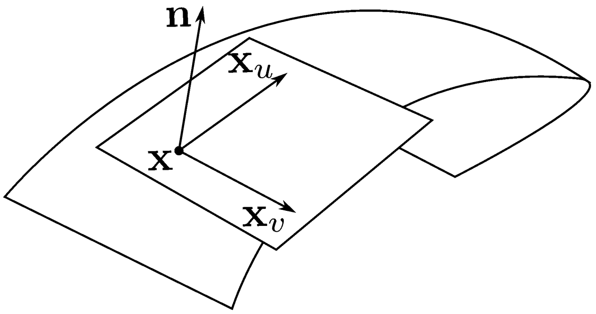


Fig. 1 Gauß frame $\{X_u, X_w, N\}$ for the surface X

(b) Elements of $T_\mathbf{u}X$ are called *tangent vectors*.
(c) The vector field $N := \frac{[X_u, X_w]}{||[X_u, X_w]||}$, where $[.,.]$ is the cross product, is called a *unit normal field*.
(d) The map $N : U \to S^2 \subset \mathbb{E}^3$ is called *Gauß map*, and the moving frame $\{X_u, X_w, N\}$ is called the *Gauß frame* of the surface as displayed in Fig. 1.

Some properties of the surface can be determined using the first and second fundamental forms. The first fundamental form allows to make measurements on the surface: lengths of curves, angles between tangent vectors, areas of regions, etc. without referring Back to the ambient space $\mathbb{E}^3$.

Definition 2 Let $X : U \to \mathbb{E}^3$ be a surface. The bilinear form of $T_\mathbf{u}X$ induced by the scalar product $\langle \cdot, \cdot \rangle$ of $\mathbb{E}^3$ by restriction is called the *first fundamental form* $I_\mathbf{u}$ of the surface.

Remark 2 Properties of the first fundamental form:

(a) The matrix representation of the first fundamental form with respect to the basis $\{X_u, X_w\}$ of $T_\mathbf{u}X$ is given by

$$\begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix} = \begin{pmatrix} \langle X_u, X_u \rangle \langle X_u, X_w \rangle \\ \langle X_w, X_u \rangle \langle X_w, X_w \rangle \end{pmatrix} . \tag{1}$$

(b) Let us denote by

$$g := \det(g_{ij})$$

the determinant of the first fundamental form.
(c) The first fundamental form is symmetric, positive definite, and a geometric invariant.

The second fundamental form allows us to study surface curvature and torsion. One especially interesting consequence of the second fundamental form can be found in the Weingarten equations which will prove useful when considering the main theorem of this paper.

Definition 3 Let $X : U \to \mathbb{E}^3$ be a surface and $\mathbf{u} \in U$.

(a) The linear map $L : T_{\mathbf{u}}X \to T_{\mathbf{u}}X$ defined by $L := -\mathrm{d}N_{\mathbf{u}} \cdot \mathrm{d}X_{\mathbf{u}}$ is called the *Weingarten map*.
(b) The bilinear form $II_{\mathbf{u}}$ defined by $II_{\mathbf{u}}(A, B) := \langle L(A), B\rangle$ for each $A, B \in T_{\mathbf{u}}X$ is called the *second fundamental form* of the surface.

Remark 3 Properties of the second fundamental form:

(a) The matrix representation of $II_{\mathbf{u}}$ with respect to the canonical basis $\{\mathbf{e}_1, \mathbf{e}_2\}$ of $T_{\mathbf{u}}\mathbb{E}^2$ (identified with $\mathbb{E}^2$) and the associated basis $\{X_u, X_w\}$ of $T_{\mathbf{u}}X$ is given by

$$\begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} = \begin{pmatrix} \langle -N_u, X_u\rangle & \langle -N_u, X_w\rangle \\ \langle -N_w, X_u\rangle & \langle -N_w, X_w\rangle \end{pmatrix} = \begin{pmatrix} \langle N, X_{uu}\rangle & \langle N, X_{uw}\rangle \\ \langle N, X_{wu}\rangle & \langle N, X_{ww}\rangle \end{pmatrix}, \tag{2}$$

i.e.

$$h_{ij} := \langle -N_i, X_j\rangle = \langle N, X_{ij}\rangle .$$

We can assume that $h_{12} = h_{21}$ since we are considering C^r-continuous surfaces.
(b) Let us denote by

$$h := \det(h_{ij})$$

the determinant of the second fundamental form.
(c) We call two geometric objects *congruent* to each other iff. there is an isometric transformation (i.e. only translation, rotation, and reflection are employed) from one to the other. Congruences preserve lengths and angles.
(d) The second fundamental form is invariant under congruences of $\mathbb{E}^3$ and orientation-preserving changes of variables.
(e) It can be shown that $\langle N_i, N\rangle = 0$; $i = 0, 1$. Thus, N_i can be represented by the local frame of the tangent plane, and the following relation holds

$$N_i = -\sum_{k=0}^{1} h_i^k X_k , \tag{3}$$

where the following equations

$$\begin{aligned} h_1^1 &= \frac{h_{11}g_{22} - h_{12}g_{12}}{g} & h_1^2 &= \frac{h_{12}g_{11} - h_{11}g_{12}}{g} \\ h_2^1 &= \frac{h_{12}g_{22} - h_{22}g_{12}}{g} & h_2^2 &= \frac{h_{22}g_{11} - h_{12}g_{12}}{g} \end{aligned} \tag{4}$$

are called *Weingarten equations* [5]. Further, N_{ij} can be expressed in terms of the Gauß frame, and it can be shown that the following relation holds

$$X_{ij} = h_{ij}N + \sum_{k=0}^{1} \Gamma_{ij}^{k} X_k \,. \tag{5}$$

where $\Gamma_{ij}^{k} = \langle X_k, X_{ij} \rangle$ are called the *Christoffel Symbols*.

Curvature is of great interest in the context of differential geometry. The minimal and maximal curvatures k_1, k_2 at a surface point are the basis for the more interesting definitions of mean curvature and Gauß curvature. In this work, we examine total curvature of surfaces under deformation in normal direction.

Considering surface curves, we get to know the geometric interpretations of the second fundamental form:

Let $A := \lambda^1 X_u + \lambda^2 X_w$ be a tangent vector with $||A|| = 1$. If we intersect the surface with the plane given by N and A, we get an intersection curve y with the following properties:

$$y'(s) = A \text{ and } \mathbf{e}_2 = \pm N \,,$$

where $\mathbf{e}_2$ is the principal normal vector of the space curve y.

The implicit function theorem implies the existence of this so-called *normal section curve*. To calculate the minimal and maximal curvature of a normal section curve (the so-called *normal section curvature*), we can use the method of *Lagrange multipliers* because we are looking for extreme values of the normal section curvature k_N with the condition $g_{ij}\lambda^i\lambda^j = 1 = ||y'||$ (Fig. 2).

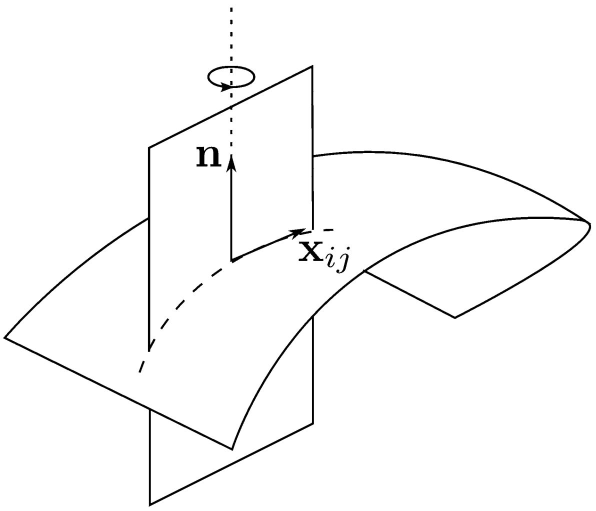


Fig. 2 Construction of normal section curves

As a result of these considerations, we can define various notions of curvature:

Definition 4 Let $X : U \to \mathbb{E}^3$ be a surface and $A = \lambda^1 X_u + \lambda^2 X_w$ a tangent vector of X at $\mathbf{u}$.

(a) The Weingarten map L is self-adjoint.
(b) The *normal section curvature* $k_N(\lambda^1, \lambda^2)$ can be computed as:

$$k_N(\lambda^1, \lambda^2) = \frac{h_{ij}\lambda^i\lambda^j}{g_{ij}\lambda^i\lambda^j}\,.$$

Unless the normal section curvature is the same for all directions (umbilical points), there are two perpendicular directions A_1 and A_2 in which k_N attains its absolute maximum and its absolute minimum.
(c) A_1 and A_2 are the *principal directions*.
(d) The corresponding normal section curvatures, k_1 and k_2, are called *principal curvatures* of the surface.
(e) Let $X : U \to \mathbb{E}^3$ be a surface and $y : I \to \mathbb{E}^3$ be a surface curve. We denote by $\hat{y}(t)$ the orthogonal projection of $y(t)$ on the tangent plane $T_{\mathbf{u}}X$ at (an arbitrary) point $P := X(\mathbf{u})$. The *geodesic curvature* k_g of y at P is defined as the curvature of the projected curve $\hat{y}(t)$ at P. A curve $y(t)$ on a surface X is called *geodesic* if its geodesic curvature k_g vanishes identically.
(f) $k_g = \det(\dot{y}, \ddot{y}, N)$, where dots denote derivatives with respect to the arc length of y.
(g) $H := \mathrm{trace}(L) = \frac{1}{2} \cdot (k_1 + k_2)$ is called the *mean curvature*.
(h) $K := k_1 \cdot k_2 = \det(L) = \frac{\det(II)}{\det(I)}$ is called the *Gauß curvature*.
(i) *Total Gauß curvature*, or short, *total curvature*, is defined as $K_{\mathrm{tot}} = \iint_X K \mathrm{d}X$.

Remark 4 (Geodesics and Curvature)

(a) An arc of minimum length on a surface joining two arbitrary points must be an arc of a geodesic.
(b) Assuming the boundary of a surface is given and we have to fit in a surface patch of minimal area, then the minimal curvature of this patch has to vanish, in which case, the mean curvature $H \equiv 0$ will also vanish.

3 Deformations

Let $X(u, w)$ be the masterpiece of an industrial surface. Let us further assume that it is a minimal surface (i.e. $H \equiv 0$), such that it covers a minimal area. This masterpiece should be deformed along its normal direction $N(u, w)$ by applying a deformation function $F(u, w)$ ($F : U \to \mathbb{E}$). Deformations along the normal mean that interior deformations of the surface are not permitted (no inner bending).

We consider linear deformations of the form

$$\tilde{X}(u,w,t) := X(u,w) + t \cdot F(u,w) \cdot N(u,w)\,, \tag{6}$$

for $t \in (-\varepsilon, \varepsilon)$, $\tilde{g} = g + o(t^2)$, such that $o(t^2)$ constitutes an infinitesimal change. Let us notice that the more general case of linear deformations

$$\tilde{X}(u,w,t) = X(u,w) + tZ(u,w)\,, \tag{7}$$

where $Z(u,w)$ is a continuous vector field ($Z : U \to \mathbb{E}^3$), is called an infinitesimal bending if $\mathrm{d}s_t^2 = \mathrm{d}s^2 + o(t^2)$, i.e. the difference of the squares of the line elements of these surfaces has at least second order [6, 10, 11].

Let us first prove two properties of minimal surfaces which will be needed to prove Theorem 1.

Lemma 1 *For a minimal surface $X(u,w)$, i.e. a surface with $H \equiv 0$, we get*

(a) $[X_u, N_w] + [N_u, X_w] = 0\,,$
(b) $\langle N, h_{11}N_{ww} + h_{22}N_{uu} - h_{12}N_{uw} - h_{12}N_{wu}\rangle = 0\,,$

where $N = \frac{[X_u,X_w]}{g}$ *for* $g = \det\begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22}\end{pmatrix} = g_{11}g_{22} - g_{12}^2\,.$

Proof We will prove part (a) and part (b) of this Lemma separately.

(a) To prove Lemma 1a, we can expand Eq.3 to

$$N_u = \frac{h_{12}g_{12} - h_{11}g_{22}}{g}X_u + \frac{h_{11}g_{12} - h_{12}g_{11}}{g}X_w \tag{8}$$

$$N_w = \frac{h_{22}g_{12} - h_{12}g_{22}}{g}X_u + \frac{h_{12}g_{12} - h_{22}g_{11}}{g}X_w\,, \tag{9}$$

from which we can immediately conclude the assumption:

$$\begin{aligned}
[X_u, N_w] + [N_u, X_w] &= \frac{h_{12}g_{12} - h_{22}g_{11}}{g}[X_u, X_w] + \frac{h_{12}g_{12} - h_{11}g_{22}}{g}[X_u, X_w] \\
&= (h_{12}g_{12} - h_{22}g_{11} + h_{12}g_{12} - h_{11}g_{22}) \cdot \frac{[X_u, X_w]}{g} \\
&= -(h_{11}g_{22} + h_{22}g_{11} - 2h_{12}g_{12}) \cdot N \\
&= -\frac{h_{11}g_{22} + h_{22}g_{11} - 2h_{12}g_{12}}{g} \cdot g \cdot N \\
&= -(k_1 + k_2) \cdot g \cdot N \\
&= -2H \cdot g \cdot N = 0\,.
\end{aligned}$$

(b) To prove Lemma 1b, we first compute the second derivatives of N. Then, using the Weingarten equations, we can conclude the following relations:

$$\begin{aligned}
N_{uu} &= -\frac{\partial h_1^1}{\partial u} X_u - h_1^1 X_{uu} - \frac{\partial h_1^2}{\partial u} X_w - h_1^2 X_{wu} \\
N_{ww} &= -\frac{\partial h_2^1}{\partial w} X_u - h_2^1 X_{uw} - \frac{\partial h_2^2}{\partial w} X_w - h_2^2 X_{ww} \\
N_{uw} &= -\frac{\partial h_1^1}{\partial w} X_u - h_1^1 X_{uw} - \frac{\partial h_1^2}{\partial w} X_w - h_1^2 X_{ww} \\
N_{wu} &= -\frac{\partial h_2^1}{\partial u} X_u - h_2^1 X_{uu} - \frac{\partial h_2^2}{\partial u} X_w - h_2^2 X_{wu} \,.
\end{aligned}$$

Next, we look at the scalar product of the normal vector and its second partial derivatives. From this computation, we receive all basic components needed to express part the formula given in part (b) of this Lemma:

$$\begin{aligned}
\langle N, N_{uu}\rangle &= -h_1^1\langle N, X_{uu}\rangle - h_1^2\langle N, X_{wu}\rangle &&= -h_1^1 h_{11} - h_1^2 h_{12} \\
\langle N, N_{ww}\rangle &= -h_2^1\langle N, X_{uw}\rangle - h_2^2\langle N, X_{ww}\rangle &&= -h_2^1 h_{12} - h_2^2 h_{22} \\
\langle N, N_{uw}\rangle &= -h_1^1\langle N, X_{uw}\rangle - h_1^2\langle N, X_{ww}\rangle &&= -h_1^1 h_{12} - h_1^2 h_{22} \\
\langle N, N_{wu}\rangle &= -h_2^1\langle N, X_{uu}\rangle - h_2^2\langle N, X_{wu}\rangle &&= -h_2^1 h_{11} - h_2^2 h_{12} \,.
\end{aligned}$$

We want to show that $\langle N, h_{11}N_{ww} + h_{22}N_{uu} - h_{12}N_{uw} - h_{12}N_{wu}\rangle = 0$. Taking the above results, combined with Eq. 4, we arrive at

$$\begin{aligned}
&\langle N, h_{11}N_{ww} + h_{22}N_{uu} - h_{12}N_{uw} - h_{12}N_{wu}\rangle \\
&= \langle N, h_{11}N_{ww}\rangle + \langle N, h_{22}N_{uu}\rangle - \langle N, h_{12}N_{uw}\rangle - \langle N, h_{12}N_{wu}\rangle \\
&= -h_{11}h_2^1 h_{12} - h_{11}h_2^2 h_{22} - h_{22}h_1^1 h_{11} - h_{22}h_1^2 h_{12} \\
&\quad + h_{12}h_1^1 h_{12} + h_{12}h_1^2 h_{22} + h_{12}h_2^1 h_{11} + h_{12}h_2^2 h_{12} \\
&= -h_{11}h_{22}(h_1^1 + h_2^2) + (h_{12})^2(h_1^1 + h_2^2) \\
&= (h_{11}h_{22} - (h_{12})^2)(-h_2^2 - h_1^1) \\
&= \frac{h}{g}(h_{12}g_{12} - h_{11}g_{22} + h_{12}g_{12} - h_{22}g_{11}) \\
&= \frac{h}{g}(-2Hg) \\
&= -2hH = 0 \,.
\end{aligned}$$

$\square$

We are now interested in shape-preserving modification of the masterpiece. We consider infinitesimal deformations which do not change the Gauß curvature, and therefore preserve the total curvature of the minimal surface.

We restrict ourselves to exterior deformations, i.e. deformations in normal direction. Interior deformations, such as perturbations in the tangent plane, are not permitted. This restriction serves the purpose of exaggerating or reducing features that are present in the masterpiece but refraining from introducing additional perturbations. We can now introduce the main theorem of this paper:

Theorem 1 *A linear deformation*

$$\tilde{X}(u,w,t) = X(u,w) + tF(u,w)N(u,w) \tag{10}$$

of a minimal surface $X(u,w)$ with $t \in (-\varepsilon,\varepsilon)$, and $\tilde{g} = g + o(t^2)$ preserves the Gauß curvature if

$$\begin{aligned} \mathrm{D}F := {} & h_{11}F_{ww} + h_{22}F_{uu} - 2h_{12}F_{uw} - F_u(h_{11}\Gamma^1_{22} - h_{12}\Gamma^1_{12} + h_{22}\Gamma^1_{11})\sqrt{g} \\ & + F_w(h_{11}\Gamma^2_{22} - h_{12}\Gamma^2_{12} + h_{22}\Gamma^2_{11})\sqrt{g} \\ = {} & 0\,, \end{aligned}$$

and therefore preserves the total curvature $\iint_S K\mathrm{d}s$ of our minimal surface $X(u,w)$.

To examine the impact of linear deformation as given in Eq. (10), we need to observe changes in some surface properties. We start with normal vectors, along which we perturb the surface. Normal vectors deform as follows:

$$\tilde{N}(u,w,t) = \frac{1}{||\tilde{N}||}\{[X_u, X_w] + t\cdot[F_wX_u - F_uX_w, N]\} + \mathrm{o}(t^2)\,.$$

The deformed second fundamental form $\widetilde{II}$, defined as

$$\begin{pmatrix} \tilde{h}_{11} & \tilde{h}_{12} \\ \tilde{h}_{12} & \tilde{h}_{22} \end{pmatrix} = \begin{pmatrix} \langle \tilde{N}, \tilde{X}_{uu}\rangle\langle \tilde{N}, \tilde{X}_{uw}\rangle \\ \langle \tilde{N}, \tilde{X}_{wu}\rangle\langle \tilde{N}, \tilde{X}_{ww}\rangle \end{pmatrix},$$

can be written as

$$\begin{aligned} \tilde{h}_{11} &= \langle N + t\cdot[F_wX_u - F_uX_w, N], X_{uu} + tF_{uu}N + 2tF_uN_u + tFN_{uu}\rangle + o(t^2) \\ &= h_{11} + t\cdot\{F_{uu} + \det(F_wX_u - F_uX_w, N, X_{uu}) + F\langle N, N_{uu}\rangle\} + o(t^2) \\ \tilde{h}_{22} &= h_{22} + t\cdot\{F_{ww} + \det(F_wX_u - F_uX_w, N, X_{ww}) + F\langle N, N_{ww}\rangle\} + o(t^2) \\ \tilde{h}_{12} &= h_{12} + t\cdot\{F_{uw} + \det(F_wX_u - F_uX_w, N, X_{uw}) + F\langle N, N_{uw}\rangle\} + o(t^2) \\ \tilde{h}_{21} &= h_{21} + t\cdot\{F_{wu} + \det(F_wX_u - F_uX_w, N, X_{wu}) + F\langle N, N_{wu}\rangle\} + o(t^2)\,. \end{aligned}$$

We know that $\tilde{K} = \frac{\det(\widetilde{II})}{\det(\tilde{I})}$, and we already have the determinant $\det(\tilde{I}) = g$ of the first fundamental form, which remains identical to $\det(I)$ up to an infinitesimal change under deformation. To compute $\tilde{K}$, we have to compute the determinant of the second fundamental form, $\det(\widetilde{II}) = \tilde{h}_{11} \cdot \tilde{h}_{22} - \tilde{h}_{12} \cdot \tilde{h}_{12}$.

$$\begin{aligned}
\det(\widetilde{II}) &= \tilde{h}_{11} \cdot \tilde{h}_{22} - \tilde{h}_{12} \cdot \tilde{h}_{12} \\
&= h_{11}h_{22} - h_{12}h_{12} + o(t^2) \\
&\quad + t \cdot \{h_{11}F_{ww} + h_{11}\det(F_wX_u - F_uX_w, N, X_{ww}) + h_{11}F\langle N, N_{ww}\rangle\} \\
&\quad + t \cdot \{h_{22}F_{uu} + h_{22}\det(F_wX_u - F_uX_w, N, X_{uu}) + h_{22}F\langle N, N_{uu}\rangle\} \\
&\quad - t \cdot \{h_{12}F_{wu} + h_{12}\det(F_wX_u - F_uX_w, N, N_{wu}) + h_{12}F\langle N, N_{wu}\rangle\} \\
&\quad - t \cdot \{h_{12}F_{uw} + h_{12}\det(F_wX_u - F_uX_w, N, N_{uw}) + h_{12}F\langle N, N_{uw}\rangle\} \\
&= h_{11}h_{22} - h_{12}^2 + o(t^2) \\
&\quad + t\{h_{11}F_{ww} + h_{11}\det(F_wX_u - F_uX_w, N, X_{ww}) \\
&\qquad + h_{22}F_{uu} + h_{22}\det(F_wX_u - F_uX_w, N, X_{uu}) \\
&\qquad - 2h_{12}F_{uw} - 2h_{12}\det(F_wX_u - F_uX_w, N, N_{uw})\}.
\end{aligned}$$

As we know from Lemma 1b, $\langle N, h_{11}N_{ww} + h_{22}N_{uu} - h_{12}N_{uw} - h_{12}N_{wu}\rangle = 0$ holds. Assuming $F_{uw} = F_{wu}$, we can conclude that

$$\begin{aligned}
\tilde{K} &= \frac{\tilde{h}_{11}\tilde{h}_{22} - \tilde{h}_{12}^2}{g} + o(t^2) \\
&= K + t \cdot \{h_{11}F_{ww} + h_{22}F_{uu} - 2h_{12}F_{uw} \\
&\qquad + h_{11}(F_w\Gamma_{22}^2 - F_u\Gamma_{22}^1)\det(X_u, N, X_w) \\
&\qquad + h_{22}(F_w\Gamma_{11}^2 - F_u\Gamma_{11}^1)\det(X_u, N, X_w) \\
&\qquad + 2h_{12}(F_w\Gamma_{12}^2 - F_u\Gamma_{12}^1)\det(X_u, N, X_w)\}.
\end{aligned}$$

Since $\det(X_u, N, X_w) = -\det(N, X_u, X_w) = -\langle N, [X_u, X_w]\rangle = -\sqrt{g}$, the Gauß curvature changes als follows under deformation:

$$\begin{aligned}
\tilde{K} = K + t \cdot \{&h_{11}F_{ww} + h_{22}F_{uu} - 2h_{12}F_{uw} + h_{11}(F_w\Gamma_{22}^2 - F_u\Gamma_{22}^1)\sqrt{g} \\
&+ h_{22}(F_w\Gamma_{11}^2 - F_u\Gamma_{11}^1)\sqrt{g} - 2h_{12}(F_w\Gamma_{12}^2 - F_u\Gamma_{12}^1)\sqrt{g}\}.
\end{aligned}$$

□

This concludes the proof of the main theorem of this paper.

4 Examples

In the following examples, we consider linear deformations, assuming bilinear distribution function $F(u, w) = au + bw + c$ which has the derivatives $F_u = a$, $F_w = b$, and $F_{uw} = F_{wu} = 0$.

Example 1 (Helicoid) We deform a helicoid X, which is a minimal surface of the form (Fig. 3)

$$X(u, w) = \begin{pmatrix} u\cos w \\ u\sin w \\ d \cdot w \end{pmatrix},$$

with $d = \frac{d_0}{2\pi}$, where d_0 is the number of windings.

This gives us the following derivatives and normal vector:

$$X_u = \begin{pmatrix} \cos w \\ \sin w \\ 0 \end{pmatrix} \quad X_w = \begin{pmatrix} -u\sin w \\ u\cos w \\ d \end{pmatrix} \quad X_{uw} = \begin{pmatrix} -\sin w \\ \cos w \\ 0 \end{pmatrix}$$

$$N = \frac{[X_u, X_w]}{||[X_u, X_w]||} = \frac{1}{\sqrt{u^2 + d^2}} \begin{pmatrix} d\sin w \\ -d\cos w \\ u \end{pmatrix}.$$

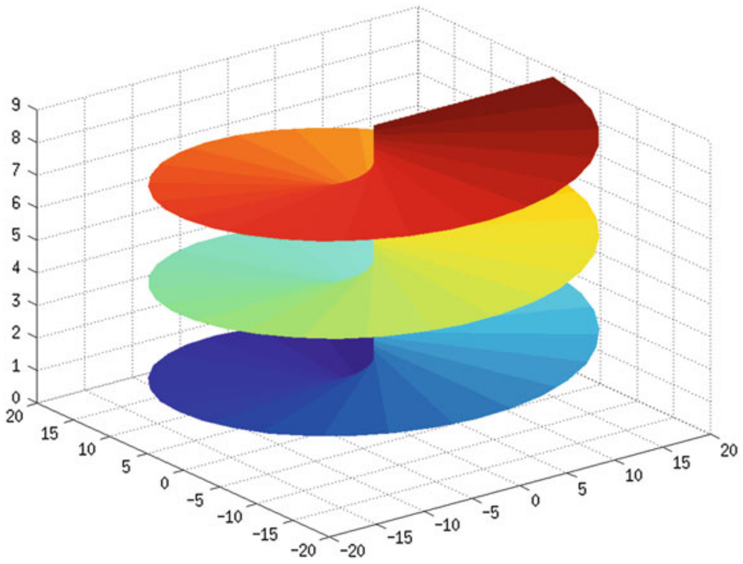


Fig. 3 A helicoid with $u, w \in [0, 6\pi]$, $d = \frac{3}{2\pi}$

Next, we compute the elements g_{ij} of the first fundamental form, and the elements h_{ij} of the second fundamental form.

$$\begin{aligned}
g_{11} &= \langle X_u, X_u\rangle = \cos^2 w + \sin^2 w + 0 && = 1\\
g_{12} &= \langle X_u, X_w\rangle = -u\sin w\cos w + u\sin w\cos w + 0 && = 0\\
g_{22} &= \langle X_w, X_w\rangle = u^2\sin^2 w + u^2\cos^2 w + d^2 && = u^2 + d^2\\
h_{11} &= \langle N, X_{uu}\rangle = \frac{d\,\sin w\cos w - d\,\cos w\sin w}{\sqrt{u^2+d^2}} && = 0\\
h_{12} &= \langle N, X_{uw}\rangle = \frac{-d\,\sin^2 w - d\,\cos^2 w}{\sqrt{u^2+d^2}} && = \frac{-d}{\sqrt{u^2+d^2}}\\
h_{22} &= \langle N, X_{ww}\rangle = \frac{-du\cos w\sin w + du\,\cos w\sin w}{\sqrt{u^2+d^2}} && = 0\,.
\end{aligned}$$

If we compute DF for our surface X and our deformation function F, and set $DF = 0$ (Theorem 1), we end up with

$$\begin{aligned}
\mathrm{D}F &= h_{11}F_{ww} + h_{22}F_{uu} - 2h_{12}F_{uw} - (h_{11}\Gamma^1_{22} - h_{12}\Gamma^1_{12} + h_{22}\Gamma^1_{11})\cdot F_u\sqrt{g}\\
&\quad + (h_{11}\Gamma^2_{22} - h_{12}\Gamma^2_{12} + h_{22}\Gamma^2_{11})\cdot F_w\sqrt{g}\\
&= 0\cdot F_{ww} + 0\cdot F_{uu} - 2\frac{-d}{\sqrt{u^2+d^2}}\cdot F_{uw}\\
&\quad - \left(0\cdot\Gamma^1_{22} - \frac{-d}{\sqrt{u^2+d^2}}\Gamma^1_{12} + 0\cdot\Gamma^1_{11}\right)\cdot F_u\sqrt{u^2+d^2}\\
&\quad + \left(0\cdot\Gamma^2_{22} - \frac{-d}{\sqrt{u^2+d^2}}\Gamma^2_{12} + 0\cdot\Gamma^2_{11}\right)\cdot F_w\sqrt{u^2+d^2}\\
&= \frac{2d}{\sqrt{u^2+d^2}}\cdot F_{uw} + d\cdot 0\cdot F_u + d\cdot u\cdot F_w\\
&= \frac{2d}{\sqrt{u^2+d^2}}\cdot 0 + du\cdot F_w\\
&= du\cdot F_w\\
&= du\cdot b \overset{!}{=} 0\\
&\Leftrightarrow b = 0\,,
\end{aligned}$$

since $\Gamma^1_{12} = \langle X_{uw}, X_u\rangle = 0$ and $\Gamma^2_{12} = \langle X_{uw}, X_w\rangle = u\,.$

Therefore, to make the deformation (total) curvature-preserving, the bilinear distribution function has to be simplified to the form $au + c$.

The resulting deformation is shown in Fig. 4. The influence of the linear coefficient $a \in \{0, 0.5, 1\}$ is given in the first row: in the beginning, the surface is a helicoid, but with increasing a, it deforms to a funnel. This effect is amplified by the scaling parameter $t \in \{1, 1.25, 1.5\}$, since both are multipliers for the normal, as seen in the second row. The influence of the constant coefficient $c \in \{0, 0.5, 1\}$ is given in the third row: in the beginning, the helicoid's centre curve is a straight line, but with increasing c, it deforms into a helix, dragging along the adjacent portions of the surface. The last row demonstrates the effect of $t \in \{1, 2, 3\}$ on this additive portion: the almost vertical surface parts are stretched from little more than a line to long sheets hanging down.

In real-world examples, parameters have to be chosen carefully (and small) to avoid such drastic deformations. We used extremely large parameters for this example to convey a general impression of the nature of change.

Our method is targeted at infinitesimal deformations. For the sake of illustration, we have chosen extremely large parameters for the deformations in Fig. 4. More realistically, one has to choose a much smaller t since we assume $o(t^2)$ to be negligible in our proof. Thus arises $t < 1$ as a necessary requirement.

With an initial $K_{\text{tot}} = 0.0011$, we consider a change of $\Delta K_{\text{tot}} = 1$ a sufficiently small change. The discretised helicoid consists of 10,201 points, so this results an average change of 0.000098 in Gauss curvature per point. This threshold is first reached for $t = 0.015$ with $F = 1$, and it is last reached for $t = 0.086$ with $F = u$ in our example.

In Figs. 5 and 6, we use the same deformation function parameters as in Fig 4. Both Figures illustrate how ΔK_{tot} changes with increasing t. In Fig. 5, we show the change until the threshold of $\Delta K_{\text{tot}} = 1$ is reached. In Fig. 6, we continue deforming until $t = 1.6$, the maximum deformation used for the upper half of Fig. 4, to demonstrate the instabilities occurring for large t. In these cases, the prototype of a model has to be adapted before applying further infinitesimal bendings.

For this particular example, the signs of a and c do not affect ΔK_{tot} when varied individually since the helicoid is symmetric and applying the deformation with opposite sign results in a similar deformation in opposite direction.

Example 2 (Fandisk)

Large industrial surface models are typically composed of smaller parts. For example, consider a turbine: it is composed of fan blades, fandisks, and many other components. It would not necessarily make sense to deform the entire model at once, but it is relatively easy to modify a single part like a fan blade or a fandisk.

In this example, we present deformations on Hoppe's fandisk model [12]. We have recreated the part marked in the rendering of the original model (Fig. 7a) from Bézier surface patches (Fig. 7b). As most real-world examples, this model has hard edges. We preserve them as surface patch boundaries between adjacent patches.

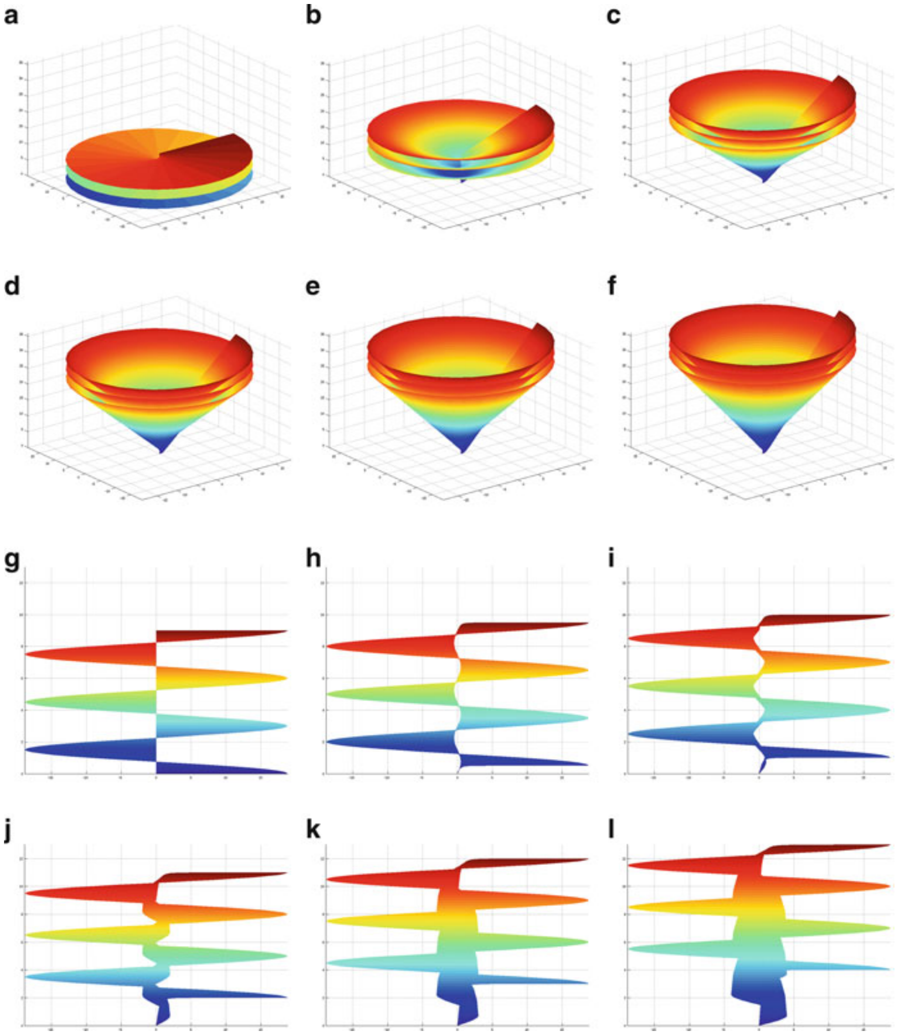


Fig. 4 The impact of deformation on the helicoid, shown separately for a, c, t. In the *first* and *third row*, only $F = au + c$ is varied. The *first row* shows a varying coefficient a with a fixed coefficient $c = 0$, while the *third row* shows a varying coefficient c with a fixed coefficient $a = 0$. For the *second* and *fourth row*, we keep F fixed, while varying the scaling parameter t in order to demonstrate the influence of scaling on the linear and constant coefficients. Figures (**a**) and (**g**) display the same surface from different perspectives. (**a**) $F = 0 t = 1$, (**b**) $F = 0.5u , t = 1$, (**c**) $F = u , t = 1$, (**d**) $F = u , t = 1.2$, (**e**)$F = u , t = 1.4$, (**f**) $F = u , t = 1.6$, (**g**) $F = 0 , t = 1$, (**h**) $F = 0.5 , t = 1$), (**i**) $F = 1 , t = 1$), (**j**) $F = 1 , t = 2$, (**k**) $F = 1 , t = 3$, (**l**) $F = 1 , t = 4$

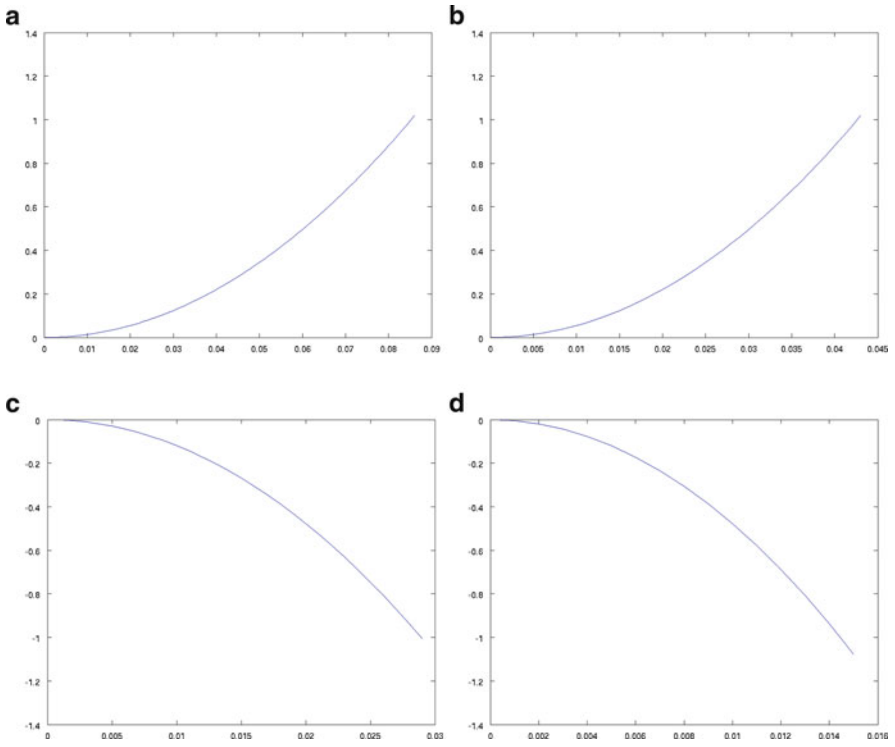


Fig. 5 Plot of ΔK_{tot} (*vertical*) over increasing t (*horizontal*) up to $|\Delta K_{\mathrm{tot}}| = 1$. (**a**) $F = 0.5u$, (**b**) $F = 1u$, (**c**) $F = 0.5$, (**d**) $F = 1$

To deform the entire model rather than a single patch at a time, we take the average of adjacent surface normals to perturb edges. Note that this can only be done on oriented manifolds.

We now take the technique we developed for minimal surfaces and adapt it to general surfaces. Our goal remains to keep the surface area as minimal as possible so the material cost remains as minimal as possible.

Now, we deform all surface patches with

$$\tilde{X}(u,w,t) = X(u,w) + t \cdot F(u,w) \cdot N(u,w)\,,$$

where

$$F(u,w) = au + bw + c$$

is our deformation function.

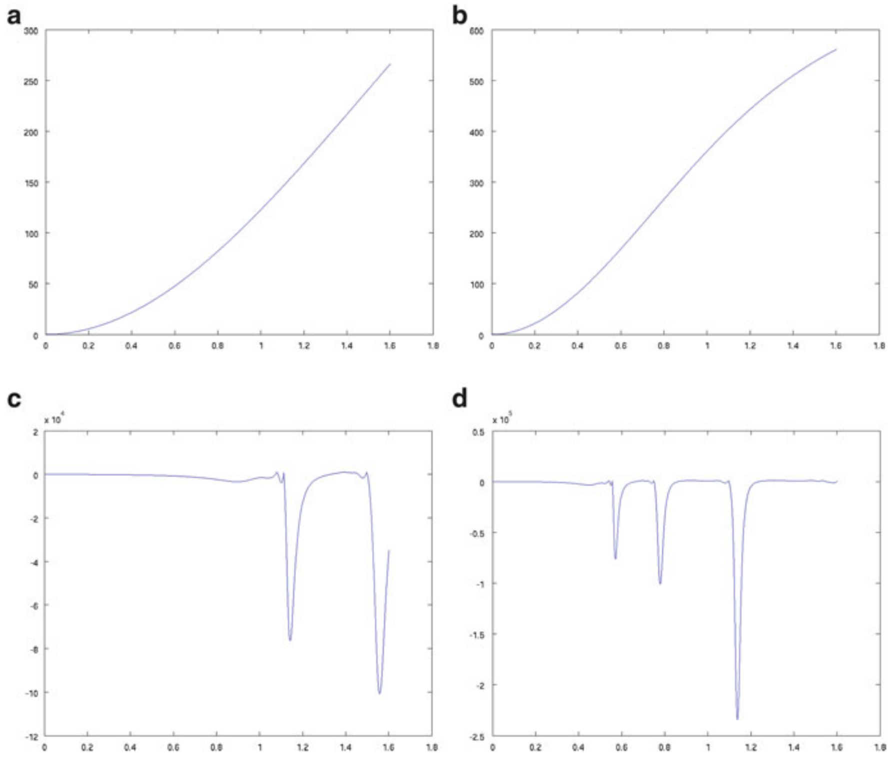


Fig. 6 Plot of ΔK_{tot} (*vertical*) over increasing t (*horizontal*) up to $t = 1.6$. (**a**) $F = 0.5u$, (**b**) $F = 1u$, (**c**) $F = 0.5$, (**d**) $F = 1$

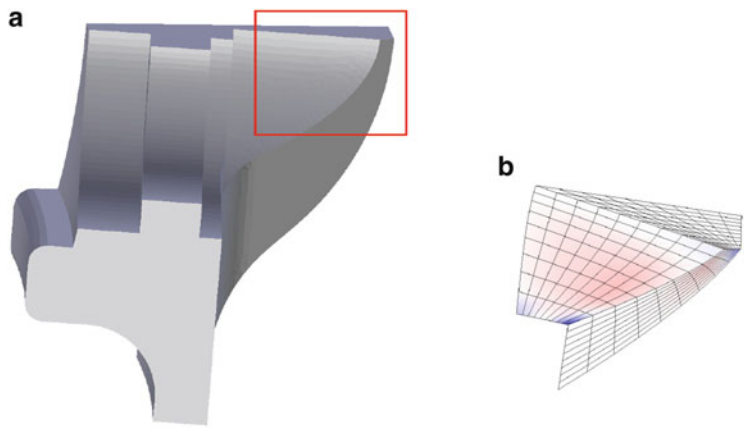


Fig. 7 Fandisk model by Hoppe [12] and a portion of it recreated from Bézier surface patches. (**a**) Model shown in Blender, (**b**) recreated as surface patches

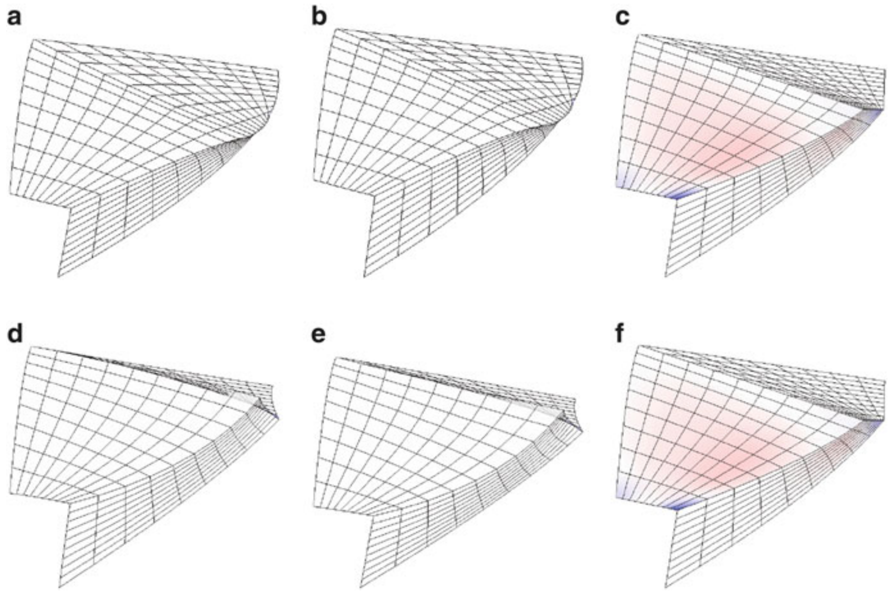


Fig. 8 Fandisk model under deformation with $t = 0.1$. (**a**) $F(u, w) = u$. ($a > 0$), (**b**) $F(u, w) = w$. ($b > 0$), (**c**) $F(u, w) = 1$. ($c > 0$), (**d**) $F(u, w) = -u$. ($a < 0$), (**e**) $F(u, w) = -w$. ($b < 0$), (**f**) $F(u, w) = -1$. ($c < 0$)

In Fig. 8, we demonstrate the effect of isolated changes of a, b, c on the deformation. Figure 9 illustrates some deformations with combined parameter changes.

The colour map in Figs. 7b, 8, and 9 depends on the Gauß curvature at each point. Blue areas are minima of Gauß curvature, red areas are maxima of Gauß curvature relative to the rest of the model. White areas are close to the median Gauß curvature.

In Fig. 10, we show changes in ΔK_{tot} over a deformation with $t \in [0, 10]$. For relatively small values of t, the deformation-induced change is stable. However, as the deformation grows, instabilities begin to occur for t approximately between 0.5 and 2.0. For extremely large values of t, the deformation is stable again, however the deformed surface no longer looks similar to the original one.

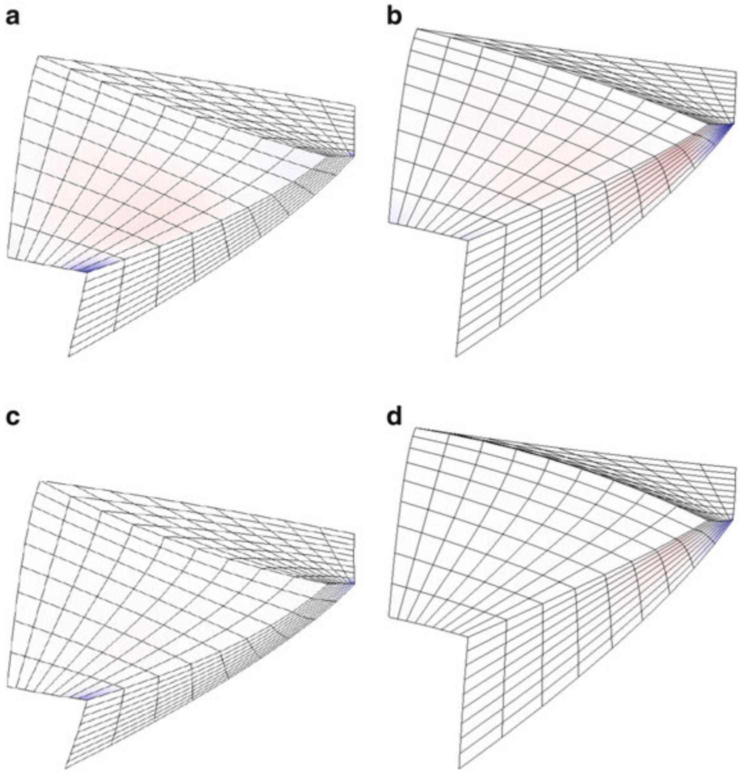


Fig. 9 Isolated change of one parameter at a time with $t = 0.1$. (**a**) $F(u, w) = u - w$, (**b**) $F(u, w) = -u + w$, (**c**) $F(u, w) = 2u - 2w$, (**d**) $F(u, w) = -2u + 2w$

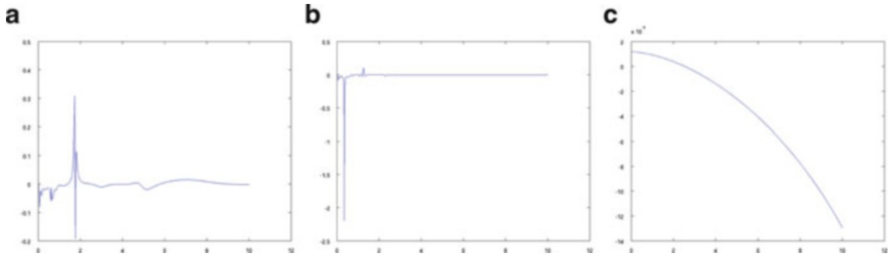


Fig. 10 Change in ΔK_{tot} for $t \in [0, 10]$. (**a**) $F = u$, (**b**) $F = w$, (**c**) $F = 1$

Conclusion

The goal of this work is to deform the masterpiece in a meaningful way, i.e. enhance or decrease features; this can be done by perturbing in normal direction. Since there is a direct connection between total curvature and bending energy, the restriction of total curvature serves to restrict the bending energy. This maintains a surface area that is as minimal as possible and therefore reduces material cost.

We have presented a method to perturb surfaces without altering their total curvature, thereby keeping their bending energy low. This results in surfaces with a small surface are which can be manufactured at lower cost than surfaces which have a higher total curvature and higher bending energy. The surface and deformation function given in Example 1 have nice analytic descriptions, so it is possible to make all computations manually. For the surface in Example 2, this is not possible since we have to define normals on hard edges.

We can deform a surface along normal direction, both outward ($F(u, w) > 0$) to increase features, and inward ($F(u, w) < 0$) to decrease features. While the examples in Fig. 4 only present the results for deformation with a positive F, the results look very similar (but upside down) for a negative F.

In real-world examples, the surface description is a lot more complicated, making it more difficult to comprehend what exactly happens to the surface during deformation.

A lot of such complex models possess sharp edges on which tangents and normals are not clearly defined. In these cases, they have to be estimated from the neighbourhood of an edge.

Our method is subjected to the same numerical limitations as partial differential equations. It is proven for objects with an analytic description, however, they are applicable to a meshes at the sacrifice of accuracy. In our first example, we computed normals and tangents analytically, but the actual deformation is performed on a discretised version of the model. Given an arbitrary mesh, our approach is limited by the availability of tangents and normals. Solutions to this are presented by [3, 17, 19, 20]. If the surface has a boundary, we are, again, limited by the availability of tangents and normals. However, given this information, the deformation procedure does not discriminate between boundary points and interior points. For a given surface patch, one can assume the normal and tangent on the boundary to be identical to its neighbourhood. Under infinitesimal deformations, the genus of a model will be preserved but if a deformation is very large, deformations can introduce self-intersections.

It is possible to introduce a flow on a given surface. One of the important and complicated challenges we want to address in the future is to apply

(continued)

our deformations in a way that they not only preserve the index sum of all singularities of a vector field defined on this surface, but also leaves the indices of each singularity unchanged.

Acknowledgements We would like to thank the German Research Foundation (DFG) for supporting this work as part of the International Research and Training Group (IRTG 1131) *Visualization of Large and Unstructured Data*.

References

1. Andersson, L.-E., Dorney, S.M., Peters, T.J., Stewart, N.F.: Polyhedral perturbations that preserve topological form. Comput. Aided Geom. Design **12**(8), 785–799 (1995)
2. Becker, J., Wieser, C., Fell, S., Steiner, K.: A multi-scale approach to material modeling of fuel cell diffusion media. Int. J. Heat Mass Transf. **54**(7), 1360–1368 (2011)
3. Berres, A.S., Hagen, H.: Feature-preserving discrete normal fields for Bézier polygons (2015, to be published)
4. Buck, M., Iliev, O., Andrä, H.: Multiscale Finite Element Coarse Spaces for the Analysis of Linear Elastic Composites. Fraunhofer-Institut für Techno-und Wirtschaftsmathematik, Fraunhofer (ITWM) (2012)
5. Do Carmo, M.P.: Differential Geometry of Curves and Surfaces. Prentice Hall, Upper Saddle River (1976)
6. Efimov, N.: Flaechenverbiegungen im Grossen (in German). Akademie-Verlag, Berlin (1957)
7. Eigensatz, M., Pauly, M.: Positional, metric, and curvature control for constraint-based surface deformation. Comput. Graph. Forum **28**(2), 551–558 (2009)
8. Eigensatz, M., Sumner, R.W., Pauly, M.: Curvature-domain shape processing. Comput. Graph. Forum **27**(2), 241–250 (2008)
9. Gravesen, J., Ungstrup, M.: Constructing invariant fairness measures for surfaces. Adv. Comput. Math. **17**(1–2), 67–88 (2002)
10. Hagen, H., Hahmann, S.: Stability conditions for free form surfaces. In: Proceedings of the Computer Graphics International 1998 (CGI'98), pp. 41–48. IEEE Computer Society, Washington (1998)
11. Hahmann, S., Hagen, H.: Numerical aspects of stability investigations on surfaces. In: Proceedings of the 6th IMA Conference on the Mathematics of Surfaces, pp. 291–308. Clarendon Press, New York (1996)
12. Hoppe, H.: Progressive meshes. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 99–108. ACM, New York (1996)
13. Ivanova-Karatopraklieva, I., Sabitov, I.Kh: Bending of surfaces. Part ii. J. Math. Sci. **74**(3), 997–1043 (1995)
14. Jordan, K.E., Miller, L., Moore, E.L.F., Peters, T.J., Russell, A.C.: Modeling time and topology for animation and visualization. Theor. Compu. Sci. **405**(1–2), 41–49 (2008)
15. Joshi, P., Sequin, C.H.: Energy minimizers for curvature-based surface functionals. CAD Conference, Waikiki, Hawaii, pp. 607–617, June 2007
16. Li, J., Peters, T.J.: Isotopic convergence theorem. Journal of Knot Theory and Its Ramifications 22.03 (2013)
17. Meyer, M., Desbrun, M., Schörder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: Hege, H.-C., Polthier, K. (eds.) Visualization and Mathematics III, Mathematics and Visualization, pp. 35–57. Springer, Berlin/Heidelberg (2003)

18. Meziani, A.: Infinitesimal bendings of high orders for homogeneous surfaces with positive curvature and a flat point. J. Differ. Equ. **239**(1), 16–37 (2007)
19. Tellier, P., Debled-Rennesson, I.: 3d discrete normal vectors. In: Discrete Geometry for Computer Imagery, pp. 447–458. Springer, Berlin (1999)
20. Thürmer, G., Wüthrich, C.A.: Normal computation for discrete surfaces in 3d space. Comput. Graph. Forum **16**, C15–C26 (1997)
21. Velimirovic, L.S., Ciric, M.S.: On the total mean curvature of piecewise smooth surfaces under infinitesimal bending. Appl. Math. Lett. **24**(9), 1515–1519 (2011)
22. Velimirovic, L.S., Rancic, S.R., Zlatanovic, M.Lj.: Visualization of infinitesimal bending of curves. In: Gautschi, W., Mastroianni, G., Rassias, T.M. (eds.) Approximation and Computation of Springer Optimization and Its Applications, vol. 42, pp. 469–480. Springer, New York (2011)

Part IV
Multi-Variate Data Analysis: Classification and Visualization of Vector Fields

Lyapunov Time for 2D Lagrangian Visualization

Filip Sadlo

1 Introduction

Vector fields play an important role in various domains, and appropriate understanding is essential in many respects. While in the early years scientific visualization tended to employ the Eulerian view, e.g., using color coding or arrow glyphs for local quantities, it is nowadays increasingly the Lagrangian view based on trajectories that is employed for visualization of vector fields (Fig. 1). This is in particular the case for time-dependent vector fields, where the straightforward Eulerian view would necessitate time-dependent visualizations, while concepts based on time-dependent trajectories, i.e., pathlines, in these fields are able to provide a notion of their true dynamics in a static picture (Fig. 1). Visualization techniques based on integral curves can be categorized into two main fields: dense and sparse. While dense visualizations basically represent the vector field at each point of its domain, e.g., by drawing a dense set of instantaneous trajectories (i.e., streamlines), sparse techniques, on the other hand, typically focus on salient features, e.g., they try to draw only the most important streamlines. Prominent examples of the two extremes are line integral convolution [5] (LIC) (Fig. 2), where a dense texture is obtained by smearing noise along trajectories, and vector field topology [13, 14] (Fig. 3), where isolated zeros (i.e., critical points) with saddle-type behavior of the vector field are extracted and streamlines (i.e., separatrices) are computed from these points forward and reverse, providing the essential structure of a vector field by separating its regions of qualitatively different behavior.

These two concepts, dense and sparse visualization, are typically applied separately, but cases where both are combined in an additive manner are not uncommon,

F. Sadlo (✉)
University of Stuttgart, Stuttgart, Germany
e-mail: sadlo@visus.uni-stuttgart.de

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*,
Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_10

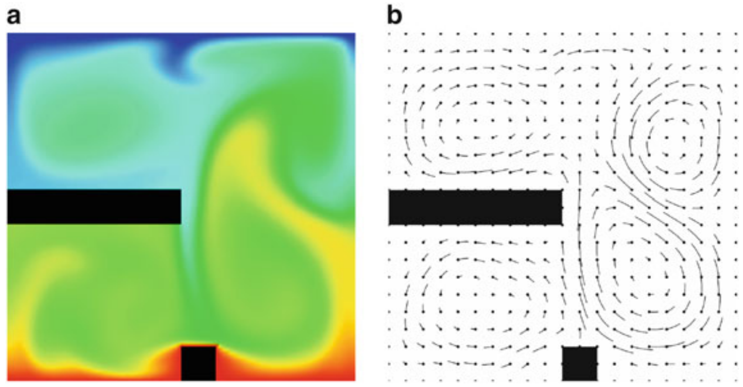


Fig. 1 Buoyant Flow dataset. The time-dependent 2D air flow is driven by a heated boundary at the *bottom* and a cooled boundary at the *top*, while gravity is pointing downward. (**a**) Temperature distribution at time $t_0 = 50.5008$ s, with *blue* at 278.15 K and *red* at 348.15 K. (**b**) Velocity field at the same time step visualized with streamlines of length 0.04 s (seeds by *black dots*)

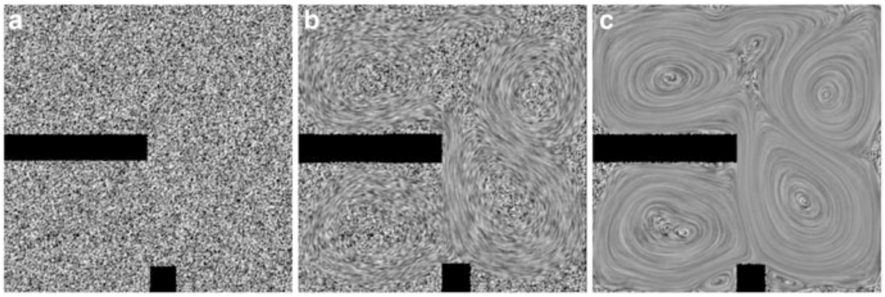


Fig. 2 Line integral convolution in Buoyant Flow dataset at $t_0 = 50.5008$ s, based on streamlines in original field (no normalization of velocity). No advection LIC (**a**) reveals noise texture, advection with $T = \pm 0.01$ s (**b**) indicates flow, while advection time $T = \pm 0.1$ s (**c**) reveals its features

since both complement each other well. While, for example, vector field topology provides a notion of the overall transport due to the vector field, LIC provides detailed morphological information about this dynamics and is therefore often used to augment topology-based visualization (Fig. 3).

Traditional vector field topology is defined in terms of asymptotic behavior of streamlines as time goes to $\pm\infty$, i.e., in practice the separatrices are integrated until they converge sufficiently close to other critical points, periodic orbits (closed streamlines), or reach the domain boundary. This raises several issues, in particular with respect to appropriateness and visual clutter. Separatrices can grow arbitrarily far from the seeding saddle point, with the result that the phenomenon of interest (the saddle-type flow behavior at the saddle point which causes the respective separatrices) has only a very remote connection to that part of the separatrix.

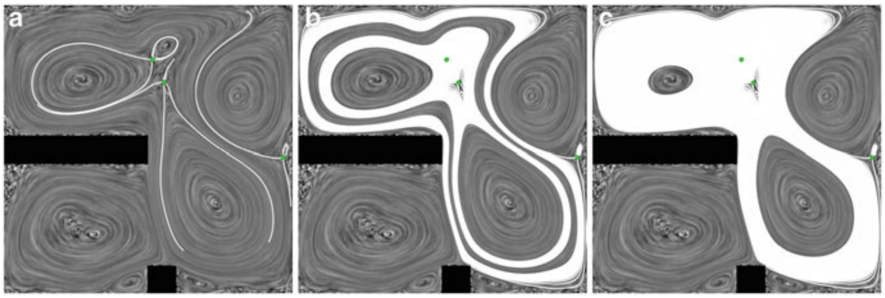


Fig. 3 Traditional vector field topology extracting saddle-type critical points (*green*) and integrating separatrices (*white*) therefrom in forward and reverse direction. LIC with $T = \pm 0.1$ s provides context. (**a**) Separatrices of integration length 2.0 s indicate qualitatively different regions of the vector field. (**b**) Increasing integration of separatrix streamlines to 100.0 s can clutter visualization (in this case due to non-vanishing divergence), in particular if integration time is further increased (1,000.0 s) (**c**). Besides insignificant visualization, this can cause prohibitive computational cost

If it makes sense to visualize and investigate such long separatrices typically depends on the application. However, such long streamlines suffer from error accumulation during integration, and visual clutter during inspection. Visual clutter can in particular impede proper analysis if the vector field exhibits vortical flow that is not divergence-free. The example shown in Fig. 3c does, due to constraints in computation time, not show the full-size separatrices, which would fill the complete vortical structure, resulting in insignificant visualization. Hence, limiting the length of separatrices can make sense both with respect to error accumulation and perception.

A further drawback of traditional topology-based visualization of vector fields is that it takes an instantaneous view on the field because it is based on streamlines. Thus, in the case of time-dependent vector field data, it would need to be applied to isolated time steps, which cannot provide appropriate insight into the true time-dependent dynamics. During the last decade, the finite-time Lyapunov exponent [11] (FTLE) field proved successful in providing a basis for topology-based visualization of time-dependent vector fields. As will be detailed in Sect. 3, the FTLE measures the divergence of pathlines with finite time length and is therefore able to indicate the boundaries between regions of qualitatively different time-dependent transport behavior. As a consequence, computation of the FTLE field requires to start a pathline at each sample point (and time step of the FTLE field), turning this concept into a large-scale problem, even for vector fields with very low resolution. The rationale behind this is that due to its Lagrangian nature, the FTLE does not reflect the local properties of a vector field. Since each pathline traverses a substantial part of the vector field, the FTLE typically exhibits highly complex structures, far beyond the variation of, e.g., tensor-product linear interpolation of the vector field itself (Fig. 4). In fact, its detail, including the length and sharpness of its ridges, grows with increasing finite advection time T (Fig. 5). The high computational cost led to

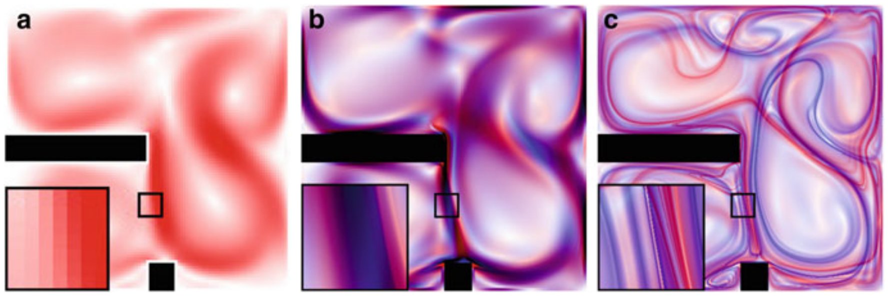


Fig. 4 Pathline-based FTLE in Buoyant Flow dataset, with $t_0 = 50.5008$ s, forward (*red*) and reverse (*blue*), with zoomed-in region (*black box*). Resolution of FTLE features is independent of original vector field data. (**a**) Velocity magnitude reflects resolution of original data. (**b**) FTLE with advection time $T = \pm 0.1$ s exhibits features that require higher resolution, and FTLE with advection time $T = \pm 1.0$ s (**c**) requires even higher resolution

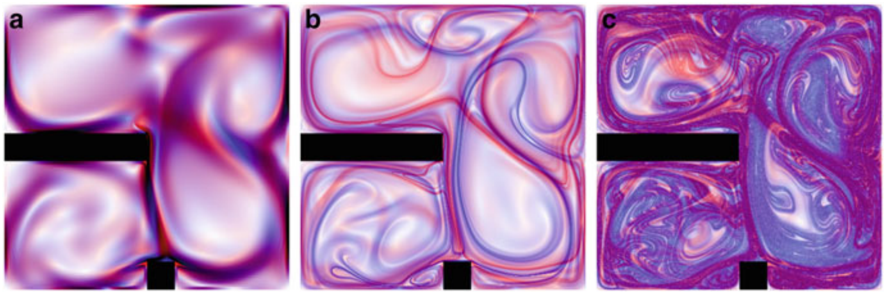


Fig. 5 Pathline-based FTLE with $t_0 = 50.5008$ s, forward (*red*) and reverse (*blue*). Increasing advection time T increases length of its ridges, $T = \pm 0.1$ s (**a**), $T = \pm 1.0$ s (**b**), and $T = \pm 5.0$ s (**c**). Insufficient advection time results fuzzy features which do not reflect the topology of the vector field (**a**). Too high advection time makes it very difficult to sample the features due to aliasing (**c**)

various acceleration techniques, e.g., related to adaptive mesh refinement [8,25] and using distributed compute environments [20].

So far, Lagrangian visualization techniques, such as the FTLE or LIC, typically employ uniform integration time, i.e., all underlying integral curves are of equal integration length. Depending if the vector field is normalized or not, this leads to constant-length or constant-time integral curves, respectively. In time-dependent fields, where such a normalization is not directly applicable, the integration time has to be adjusted to the phenomenon under investigation. While finding a global integration time that fits the overall visualization goal might be possible in some cases, it is rather common that a vector field exhibits different spatial and temporal scales, which necessitate different finite time intervals for Lagrangian analysis. This difficulty was the motivation for the finite-size Lyapunov exponent (FSLE) introduced by Aurell et al. [2]. Instead of prescribing the integration time and measuring the divergence of the endpoints of the trajectories, as in the case of the

FTLE, the FSLE prescribes a separation factor and measures how long it takes for particles (trajectories) to separate by this factor. Hence, the FSLE can be seen as the dual for the FTLE. In other words, instead of prescribing a time scale for the analysis, as in the case of the FTLE, the FSLE employs a spatial scale by varying the time scale.

In this chapter, we employ Lyapunov time (LT) for the visualization of 2D time-dependent vector fields to address the aforementioned issues. Lyapunov time reflects the limits of predictability of a system, i.e., it relates to the duration over which trajectories are not dominated by error accumulation. Hence, we propose to utilize LT for steering the integration duration in trajectory-based visualization. Due to its close relation to the FTLE and FSLE, the resulting visualizations are related to time-dependent vector field topology. In Sect. 2, we give an overview of less closely related work, followed by a more formal introduction to the FTLE, FSLE, and LT concepts in Sect. 3. In Sect. 4, we present the results of LT-guided visualization, and in section "Conclusion" we draw conclusions and discuss possible future work.

2 Related Work

Most closely related works in the field of dense Lagrangian visualization are texture advection techniques such as line integral convolution due to Cabral and Leedom [5], and delocalization of quantities due to Fuchs et al. [7], where quantities such as temperature or vortex indicators are averaged along pathlines.

Traditional (streamline-based) vector field topology was established in visualization by the works due to Perry and Chong [23], Helman and Hesselink for 2D [13] and 3D [14] flow, and Globus et al. [9]. Later works include those by Hauser et al. [18] and Theisel et al. [33], the former in the field of dynamical systems, and the latter regarding the visualization of homoclinic (heteroclinic) orbits, i.e., streamlines that connect the same (different) saddle point in 3D. Beyond that, the vector field topology concept was utilized in a variety of visualization techniques, e.g., for achieving significant streamline placement [38] or guiding streamsurface-based visualization in recirculating flow [21]. Other works in the context of recirculating flow include the visualization of the magnetic field in fusion reactors [30] and recirculation in general [22]. More recently, Bachthaler et al. [4] derived a vector field topology concept for the visualization of magnetic flux in 2D fields.

In the field of time-dependent vector field topology, Haller proposed [11] and Shadden et al. defined [32] Lagrangian coherent structures (LCS), the counterpart to separatrices in time-dependent fields, to be ridges in the FTLE field. A good introduction to the topic is the tutorial due to Shadden [31] and the survey by Pobitzer et al. [24]. Sadlo and Peikert [26] proposed to extract LCS using the height ridge concept due to Eberly [6]. Shadden et al. [32] and later Sadlo et al. [28] investigate the choice of the uniform finite time scope for FTLE computation, while Kasten et al. [17] proposed a local computation, and Üffinger et al. [36] examined the linearization involved in traditional FTLE computation and proposed alternatives beyond first-order approximation. While the majority of research focuses on the geometric shape of LCS, Bachthaler et al. [3] presented a visualization technique for the dynamics within LCS in 2D fields, i.e., for the stretching and squeezing in tangential direction. Beyond that, the LCS concept was also extended to tensor fields [15, 35] and transport in advection-diffusion flow [29], and used for computational steering of flow simulations [1]. Recent advances in time-dependent vector field topology include the reinterpretation of LCS as streakline-based vector field topology in 2D [27] and 3D [37] vector fields. Based on the findings of Haller [10], Ide et al. [16], and Mancho et al. [19], Sadlo et al. reinterpreted hyperbolic trajectories [10] as degenerate streaklines with hyperbolic behavior and their manifolds (LCS) as streak manifolds converging forward or reverse to these degenerate streaklines within finite time intervals. In contrast to traditional LCS extraction based on FTLE ridges, streak-based LCS avoid false positives caused by shear flow (similar to [12]), provide high-quality results, and are of lower computational cost, however, with the drawback that the degenerate streaklines (hyperbolic trajectories) have to be determined at high accuracy.

None of these techniques employ varying integration time for sets of trajectories and neither do they involve predictability considerations in this respect, in contrast to the approach based on Lyapunov time we propose.

3 FTLE, FSLE, and Lyapunov Time

As introduced above, the FTLE $\sigma_{t_0}^T(\mathbf{x})$ measures Lagrangian separation by determining the growth of the distance between particles during their advection over a finite advection time T. Haller [11] proposed to determine the FTLE by means of the flow map $\boldsymbol{\phi}_{t_0}^T(\mathbf{x})$, which maps the seed points $\mathbf{x}$ of pathlines started at time t_0 to their endpoints after advection for time T. Based on this, the FTLE can be obtained using the spectral norm $\|\cdot\|_2$ as

$$\sigma_{t_0}^T = \frac{1}{|T|} \ln \|\nabla \boldsymbol{\phi}_{t_0}^T\|_2 = \frac{1}{|T|} \ln \sqrt{\lambda_{\max}\left((\nabla \boldsymbol{\phi}_{t_0}^T)^\top \nabla \boldsymbol{\phi}_{t_0}^T\right)}\ , \tag{1}$$

with $\lambda_{\max}(\cdot)$ representing the major eigenvalue. It can be readily seen that the FTLE basically measures the gradient of the flow map, i.e., a linearization of the largest separation of pathline endpoints with uniform advection time T.

The FSLE can be seen as a dual approach: here, the separation factor s is prescribed and the time T_s is measured, until this factor is achieved. As proposed by Sadlo and Peikert [25], the FSLE $\hat{\sigma}_{t_0}^s(\mathbf{x})$ can be also computed from the flow map, using the FTLE as follows:

$$\hat{\sigma}_{t_0}^s = \frac{1}{|T_{t_0}^s|} \ln s\ , \tag{2}$$

with

$$T^s_{t_0} = \underset{|T|}{\arg\min} \, \|\nabla \boldsymbol{\phi}^T_{t_0}\|_2 \overset{!}{=} s \ . \tag{3}$$

For efficient computation [25], the flow map $\boldsymbol{\phi}^T_{t_0}(\mathbf{x})$ can be computed incrementally for increasing T until Eq. (3) is satisfied.

The Lyapunov time $\tau^e_{t_0}(\mathbf{x})$ represents the smallest time interval that is necessary for a perturbation started at $\mathbf{x}$ and time t_0 to grow by a factor of Euler's number e,

$$\tau^e_{t_0} = \underset{|T|}{\arg\min} \, \|\nabla \boldsymbol{\phi}^T_{t_0}\|_2 \overset{!}{=} e \ . \tag{4}$$

Hence, using Eq. (2), LT relates to the FSLE as follows:

$$\tau^e_{t_0} = \frac{1}{\hat{\sigma}^e_{t_0}} \ln e = \frac{1}{\hat{\sigma}^e_{t_0}} \ . \tag{5}$$

Throughout this chapter, we compute LT according to Eq. (5), using the algorithm [25] for FSLE computation. Due to its high computational complexity, the cost for LT computation typically predominates that of the subsequent LT-constrained trajectory-based visualization (Sect. 4).

It is of course possible to compute the FTLE, FSLE, and LT for stationary vector fields, or for isolated time steps of time-dependent fields, using streamlines. It has to be noted, however, that the resulting structures cannot be consistent with traditional vector field topology in general, although they may be similar sometimes [26, 34], because, while the FTLE, FSLE, and LT are Galilean-invariant, traditional vector field topology is not. This is demonstrated in Fig. 6.

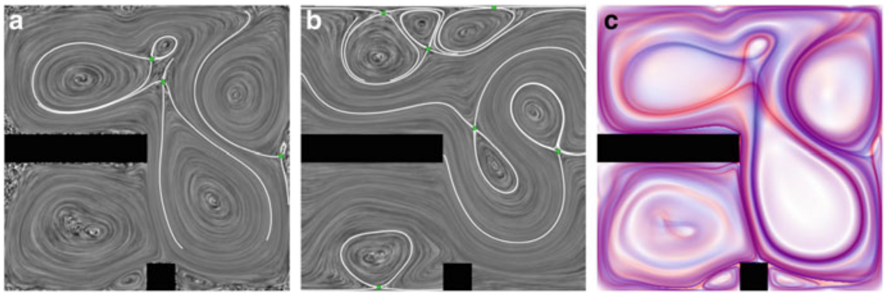


Fig. 6 Galilean invariance of FTLE in Buoyant Flow dataset at $t_0 = 50.5008$ s. Traditional vector field topology is not Galilean-invariant, i.e., result from original frame of reference (**a**) (Fig. 3a) differs from result if observer moves from right to left at speed 0.05 m/s (**b**). (**c**) Streamline-based FTLE with $T = 1$ s, forward (*red*) and reverse (*blue*), is identical in both frames and exhibits, in this example, features similar, but not identical, to (**a**)

4 Lyapunov Time for Visualization

The utility of constraining integration time with LT in trajectory-based visualization is demonstrated with three applications using the Buoyant Flow dataset (Fig. 1). First, we include LT in sparse trajectory-based visualization (Sect. 4.1). Then we exemplify LT-based visualization by means of LIC (Sect. 4.2). Finally, we provide results for LT-constrained delocalization of quantities (Sect. 4.3).

Since LIC requires streamlines for its generation and to support comparability and interpretability, we use streamlines for generating all results in this section, i.e., the FTLE, FSLE, and LT as well the derived trajectories and delocalizations are based on streamlines. In practice, however, our approach does address both steady and time-dependent vector fields. Our results are based on the fields shown in Fig. 7. Figure 7a shows the streamline-based FSLE with separation factor e and Fig. 7b the LT $\tau^{e}_{50.5008}$ derived therefrom. Since $\tau^{e}_{50.5008}$ is rather restrictive, we also provide results for $\tau^{10}_{50.5008}$, the "Lyapunov time" with separation factor 10 (Fig. 7c).

4.1 Sparse LT Trajectories

Motivated by the difficulties demonstrated in Fig. 3, we first employ LT for limiting the integration length of separatrices (Fig. 8). Since the goal of vector field topology is to depict the regions of qualitatively different behavior and since divergence is comparably small in the examined data, we aim at separatrices that perform a "single turn", i.e., that reach the critical point where they were seeded. This way they indicate the substantial regions and at the same time the effects of divergence. While a global selection of integration time cannot achieve this goal (see Figs. 3a and 8a), constraining integration by $8.0 \cdot \tau^{e}_{50.5008}$ achieves this goal for the saddles at (i) and (ii) (Fig. 8b)—providing a significant visualization of the dynamics.

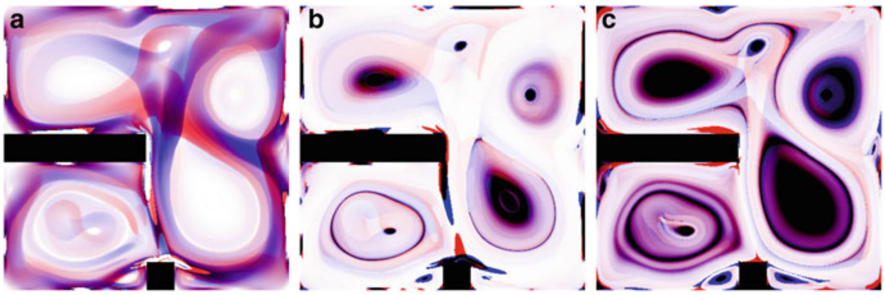


Fig. 7 (**a**) FSLE $\hat{\sigma}^{e}_{50.5008}$ forward (*red*) and reverse (*blue*). (**b**) LT $\tau^{e}_{50.5008}$ computed from (**a**), forward (*red*) and reverse (*blue*). (**c**) "Lyapunov time" $\tau^{10}_{50.5008}$ with scaling factor 10 for comparison

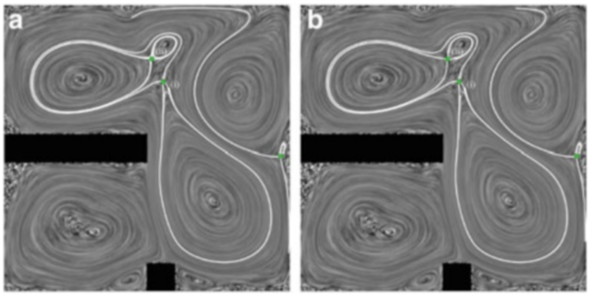


Fig. 8 (**a**) Visualization by separatrices with uniform integration length of 3 s. While the separatrices just reach the originating saddle point at (i), they pass it at (ii), resulting in insignificant visualization there. (**b**) Visualization by separatrices constrained to $8.0 \cdot \tau^{e}_{50.5008}$. Separatrices reach saddle point both at (i) and (ii), providing a more concise and better comprehensible visualization. Confer Fig. 3a for uniform integration length of 2 s, where separatrices reach (ii) but not (i)

Next, we limit integration time with LT for regularly seeded streamlines (Fig. 9). Figures 9a, d show uniform integration in forward and reverse direction, respectively. Note that in our current approach, we do not employ streamline placement to achieve an even placement of the lines—instead we use rather short integration times to prevent excessive visual clutter. We want to address streamline placement that involves LT as future work. Limiting integration length with LT not only incorporates predictability aspects, it provides at the same time structures that reflect the coherent structures. We found that limiting integration time with $\tau^{e}_{50.5008}$ (Figs. 9b, e) was rather restrictive, we preferred using $\tau^{10}_{50.5008}$ (Figs. 9c, f), as this provided more pronounced structures. As illustrated in Figs. 9g, h, our approach reveals the regions of qualitatively different behavior.

4.2 *LT LIC*

The results from regularly seeded streamlines from Sect. 4.1 directly motivate limiting integration in LIC using LT (Fig. 10). Again, $\tau^{10}_{50.5008}$, in particular $0.01 \cdot \tau^{10}_{50.5008}$ (Fig. 10f), leads to results that depict coherent regions in a more pronounced manner. In LT LIC visualizations, chaotic (less predictable) regions can be identified by more noisy areas whereas predictable regions are represented with more line-like structures. Compared to LT trajectories (Sect. 4.1), LT LIC provides a more continuous and more detailed picture of predictability and coherence.

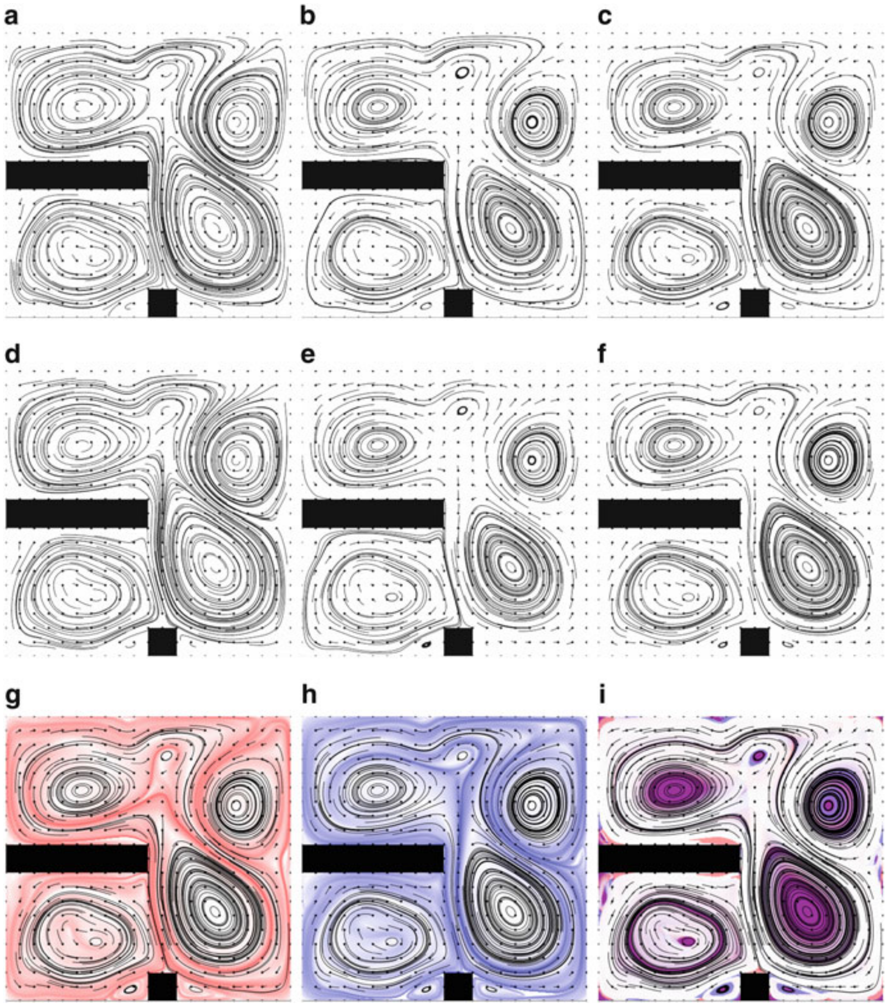


Fig. 9 (**a**)–(**c**) Forward streamlines at $t_0 = 50.5008$ s, of length 0.4 s (**a**), $0.3 \cdot \tau^{e}_{50.5008}$ (**b**), and $0.1 \cdot \tau^{10}_{50.5008}$ (**c**). (**d**)–(**f**) Same for reverse streamlines. It is apparent that $0.1 \cdot \tau^{10}_{50.5008}$ provides more expressive results. (**g**) Comparison of (**c**) with forward FTLE from Fig. 6c shows that coherent regions are well represented. (**h**) Same for (**f**) with reverse FTLE. (**i**) Forward (**c**) and reverse (**f**) streamlines together with forward (*red*) and reverse (*blue*) $0.1 \cdot \tau^{10}_{50.5008}$ for context

4.3 LT Delocalization

In our final example, we employ LT for the delocalization [7] of scalar quantities. The concept of delocalization was originally used to make vortex indicators more coherent. For each sample point, a trajectory is seeded there, a scalar field, in our case temperature, is averaged along the trajectory, and the result is stored at the

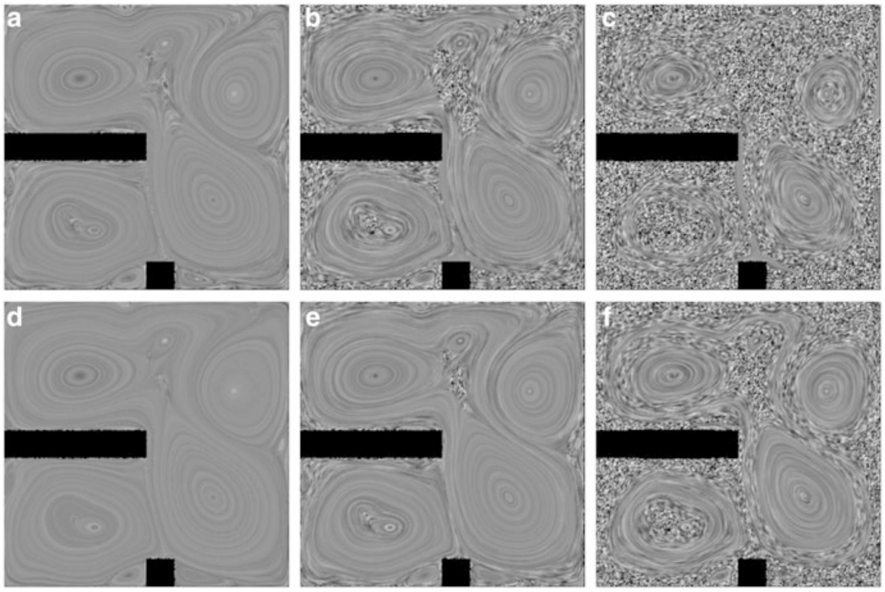


Fig. 10 Limiting LIC with forward and reverse $\tau^{e}_{50.5008}$, using $1.0 \cdot \tau^{e}_{50.5008}$ (**a**), $0.1 \cdot \tau^{e}_{50.5008}$ (**b**), and $0.01 \cdot \tau^{e}_{50.5008}$ (**c**) shows coherent regions. However, using forward and reverse $\tau^{10}_{50.5008}$, using $1.0 \cdot \tau^{10}_{50.5008}$ (**d**), $0.1 \cdot \tau^{10}_{50.5008}$ (**e**), and $0.01 \cdot \tau^{10}_{50.5008}$ (**f**) provides more significant results

seed point of the trajectory. Hence, delocalization can be interpreted as Lagrangian averaging of quantities (Fig. 11).

In contrast to the original approach (Figs. 11a–e), which uses uniform integration length, we limit the trajectories by LT (Figs. 11g–i). While the temperature field from the simulation is subject to excessive diffusion (Fig. 11c) due to numerical diffusion in the solver, delocalization provides a tool for investigating the advection of quantities with reduced diffusion, in particular the delocalization in direction reverse to the flow. Compared to traditional delocalization, LT delocalization takes at the same time into account predictability of the involved trajectories, which limits the resulting structures but prevents misinterpretation with respect to accumulation of integration error. Nevertheless, we find it more difficult to interpret the overall results from LT delocalization—providing a visualization of the underlying LT trajectories for context is important.

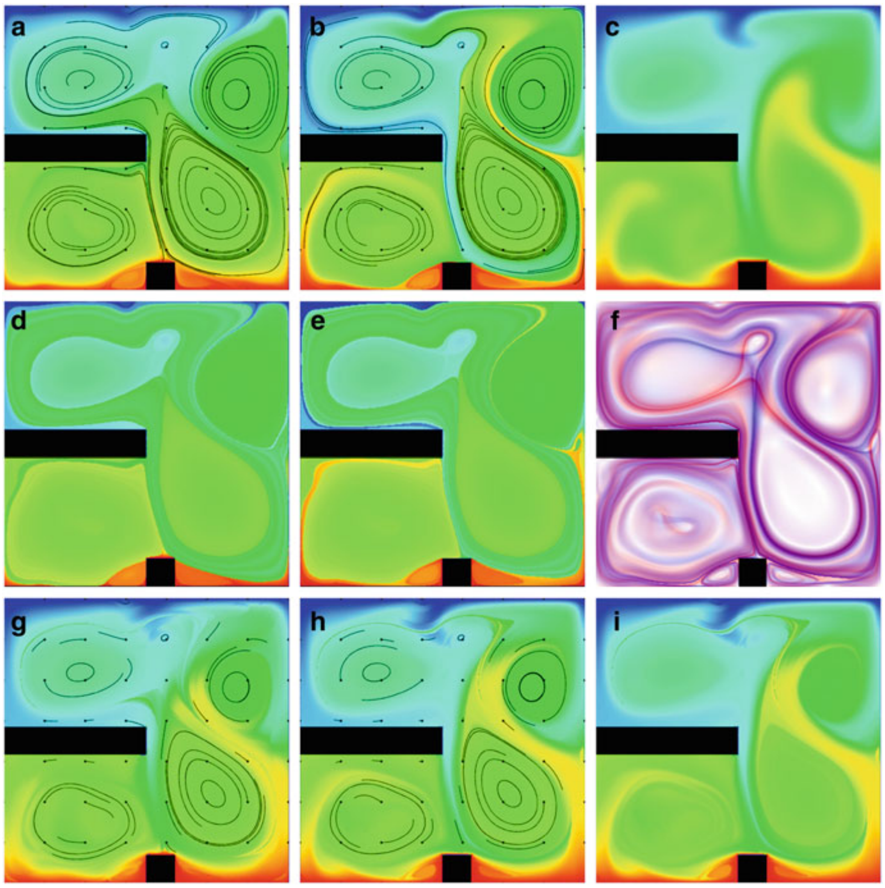


Fig. 11 Delocalization of temperature (**c**) (*blue* at 278.18 K, *red* at 348.18 K) at $t_0 = 50.5008$ s, using streamlines of length 1.0 s forward (**a**) and reverse (**b**), with selected streamlines (*black*). Delocalization with length 10.0 s forward (**d**) and reverse (**e**) reveals coherent structures (compare streamline-based FTLE (**f**), same as Fig. 6c). Limiting delocalization time with $1.0 \cdot \tau^{10}_{50.5008}$ forward (**g**) and reverse (**h**),(**i**) does not provide coherent structures as clearly, but shows predictability, i.e., only predictable structures with respect to temperature advection are shown

Conclusion
We introduced Lyapunov time in trajectory-based visualization and demonstrated its use for separatrices and direct visualization by sets of trajectories, for LIC, and for the delocalization of quantities. The resulting visualizations not only reflect predictability limitations in these concepts, they reveal at the same time the topological structure of the vector fields due to the close relation

(continued)

between Lyapunov time and the FSLE field. As future work inspired from our results in Sect. 4.1, we plan to develop streamline placement strategies that take into account Lyapunov time, both for predictability purposes and topological expressiveness.

Acknowledgements The author would like to thank the German Research Foundation (DFG) for financial support within the *Cluster of Excellence in Simulation Technology* (EXC 310).

References

1. Ament, M., Frey, S., Sadlo, F., Ertl, T., Weiskopf, D.: GPU-based two-dimensional flow simulation steering using coherent structures. In: Proceedings of Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (2011)
2. Aurell, E., Boffetta, G., Crisanti, A., Paladin, G., Vulpiani, A.: Predictability in the large: an extension of the concept of Lyapunov exponent. J. Phys. A **30**, 1–26 (1997)
3. Bachthaler, S., Sadlo, F., Dachsbacher, C., Weiskopf, D.: Space-time visualization of dynamics in Lagrangian coherent structures of time-dependent 2D vector fields. In: Proceedings of International Conference on Information Visualization Theory and Applications, pp. 573–583 (2012)
4. Bachthaler, S., Sadlo, F., Weeber, R., Kantorovich, S., Holm, C., Weiskopf, D.: Magnetic flux topology of 2D point dipoles. Comput. Graph. Forum **31**(3), 955–964 (2012)
5. Cabral, B., Leedom, L.C.: Imaging vector fields using line integral convolution. In: Proceedings of 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, pp. 263–270. ACM, New York (1993)
6. Eberly, D.: Ridges in Image and Data Analysis. Computational Imaging and Vision. Kluwer Academic, Boston (1996)
7. Fuchs, R., Peikert, R., Sadlo, F., Alsallakh, B., Gröller, M.E.: Delocalized unsteady vortex region detectors. In: Proceedings of International Workshop on Vision, Modeling and Visualization, pp. 81–90 (2008)
8. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Vis. Comput. Graph. **13**(6), 1464–1471 (2007)
9. Globus, A., Levit, C., Lasinski, T.: A tool for visualizing the topology of three-dimensional vector fields. In: Proceedings of IEEE Visualization, vol. 408, pp. 33–40 (1991)
10. Haller, G.: Finding finite-time invariant manifolds in two-dimensional velocity fields. Chaos **10**(1), 99–108 (2000)
11. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Physica D **149**(4), 248–277 (2001)
12. Haller, G.: A variational theory of hyperbolic Lagrangian coherent structures. Physica D **240**(7), 574–598 (2011)
13. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. IEEE Comput. **22**(8), 27–36 (1989)
14. Helman, J., Hesselink, L.: Visualizing vector field topology in fluid flows. IEEE Comput. Graph. Appl. **11**(3), 36–46 (1991)
15. Hlawatsch, M., Vollrath, J.E., Sadlo, F., Weiskopf, D.: Coherent structures of characteristic curves in symmetric second order tensor fields. IEEE Trans. Vis. Comput. Graph. **17**(6), 781–794 (2011)

16. Ide, K., Small, D., Wiggins, S.: Distinguished hyperbolic trajectories in time-dependent fluid flows: analytical and computational approach for velocity fields defined as data sets. Nonlinear Process. Geophys. **9**(3/4), 237–263 (2002)
17. Kasten, J., Petz, C., Hotz, I., Noack, B.R., Hege, H.-C.: Localized finite-time Lyapunov exponent for unsteady flow analysis. In: Proceedings of International Workshop on Vision, Modeling and Visualization, pp. 265–274 (2009)
18. Löffelmann, H., Doleisch, H., Gröller, E.: Visualizing dynamical systems near critical points. In: Proceedings of Spring Conference on Computer Graphics and its Applications, pp. 175–184 (1998)
19. Mancho, A.M., Small, D., Wiggins, S., Ide, K.: Computation of stable and unstable manifolds of hyperbolic trajectories in two-dimensional, aperiodically time-dependent vector fields. Physica D **182**, 188–222 (2003)
20. Panagiotidis, A., Kauker, D., Sadlo, F., Ertl, T.: Distributed computation and large-scale visualization in heterogeneous compute environments. In: Proceedings of 11th International Symposium on Parallel and Distributed Computing, pp. 87–94 (2012)
21. Peikert, R., Sadlo, F.: Topology-guided visualization of constrained vector fields. In: Topology-Based Methods in Visualization, pp. 21–34. Springer, Berlin (2007)
22. Peikert, R., Sadlo, F.: Visualization methods for vortex rings and vortex breakdown bubbles. In: Proceedings of EuroVis, pp. 211–218 (2007)
23. Perry, A.E., Chong, M.S.: A description of eddying motions and flow patterns using critical-point concepts. Ann. Rev. Fluid Mech. **19**, 125–155 (1987)
24. Pobitzer, A., Peikert, R., Fuchs, R., Schindler, B., Kuhn, A., Theisel, H., Matkovic, K., Hauser, H.: The state of the art in topology-based visualization of unsteady flow. Comput. Graph. Forum **30**(6), 1789–1811 (2011)
25. Sadlo, F., Peikert, R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. IEEE Trans. Vis. Comput. Graph. **13**(6), 1456–1463 (2007)
26. Sadlo, F., Peikert, R.: Visualizing Lagrangian coherent structures and comparison to vector field topology. In: Topology-Based Methods in Visualization II, pp. 15–30. Springer, Berlin (2009)
27. Sadlo, F., Weiskopf, D.: Time-dependent 2-D vector field topology: an approach inspired by Lagrangian coherent structures. Comput. Graph. Forum **29**(1), 88–100 (2010)
28. Sadlo, F., Üffinger, M., Ertl, T., Weiskopf, D.: On the finite-time scope for computing Lagrangian coherent structures from Lyapunov exponents. In: Topological Methods in Data Analysis and Visualization II, pp. 269–281. Springer, Berlin (2012)
29. Sadlo, F., Karch, G.K., Ertl, T.: Topological features in time-dependent advection-diffusion flow. In: Topological Methods in Data Analysis and Visualization III, pp. 217–231. Springer, Berlin (2014)
30. Sanderson, A., Chen, G., Tricoche, X., Pugmire, D., Kruger, S., Breslau, J.: Analysis of recurrent patterns in toroidal magnetic fields. IEEE Trans. Vis. Comput. Graph. **16**(6), 1431–1440 (2010)
31. Shadden, S.: LCS tutorial. http://mmae.iit.edu/shadden/LCS-tutorial/contents.html (2014)
32. Shadden, S.C., Lekien, F., Marsden, J.E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. Physica D **212**, 271–304 (2005)
33. Theisel, H., Weinkauf, T., Hege, H.-C., Seidel, H.-P.: Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In: Proceedings of IEEE Visualization, pp. 225–232 (2003)
34. Tricoche, X., Garth, C., Sanderson, A., Joy, K.: Visualizing invariant manifolds in area-preserving maps. In: Topological Methods in Data Analysis and Visualization II, pp. 109–124. Springer, Berlin (2012)
35. Tricoche, X., Hlawitschka, M., Barakat, S., Garth, C.: Beyond topology: a Lagrangian metaphor to visualize the structure of 3D tensor fields. In: New Developments in the Visualization and Processing of Tensor Fields. Springer, Berlin (2012)

36. Üffinger, M., Sadlo, F., Kirby, M., Hansen, C., Ertl, T.: FTLE computation beyond first-order approximation. In: Short Paper Proceedings of Eurographics, pp. 61–64 (2012)
37. Üffinger, M., Sadlo, F., Ertl, T.: A time-dependent vector field topology based on streak surfaces. IEEE Trans. Vis. Comput. Graph. **19**(3), 379–392 (2013)
38. Wu, K., Liu, Z., Zhang, S., Moorhead, R.J.: Topology-aware evenly spaced streamline placement. IEEE Trans. Vis. Comput. Graph. **16**(5), 791–801 (2010)

Geometric Algebra for Vector Field Analysis and Visualization: Mathematical Settings, Overview and Applications

Chantal Oberson Ausoni and Pascal Frey

1 Introduction

Nowadays, complex numerical simulations (e.g. in climate modelling, weather forecast, aeronautics, genomics, etc.) produce very large data sets, often several terabytes, that become almost impossible to process in a reasonable amount of time. Among other challenges, storage, transfer, analysis and visualization are the more crucial. This requires developing new methods and implementing new algorithms to efficiently process this large quantity of information. On the other hand, in mathematics or theoretical physics, problems are commonly posed in high-dimensional spaces and require specific methods to reduce their dimension and make the solutions understandable. In both cases, there is a critical need for an abstract, general purpose method of analysis capable of extracting the salient features of the complex data. Unfortunately, numerical algorithms are too often inadequate to perceive the mathematical properties or the general structure of the objects considered. In this chapter, we will explain how the formal language of geometric algebras may be one of these analysis tools, as it provides a unified framework bringing us closer by the topological aspects of geometrical problems, in a wide range of applications, including scientific visualization. The main strength

C. Oberson Ausoni
Sorbonne Universités, UPMC Univ Paris 06, Institut du Calcul et de la Simulation, 75005 Paris, France
e-mail: chantal.oberson-ausoni@upmc.fr

P. Frey (✉)
Sorbonne Universités, UPMC Univ Paris 06, Institut du Calcul et de la Simulation, 75005 Paris, France

Sorbonne Universités, UPMC Univ Paris 06, UMR 7598, Laboratoire Jacques-Louis Lions, 75005, Paris, France
e-mail: pascal.frey@upmc.fr

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_11

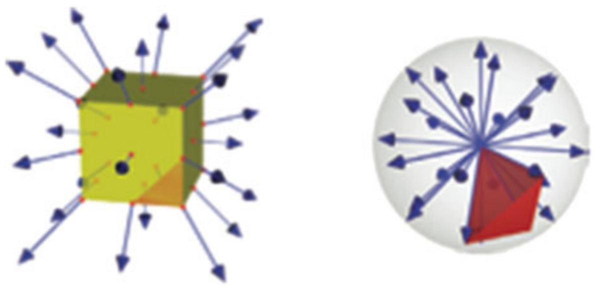


Fig. 1 Sampling a vector field over a cube (*left*) and summing the trivectors on the unit sphere (*right*), to compute an approximation of the index, see 4.2 (reprinted from [7]). Note that such a trivector is a volume in space and not a triple of vectors

of geometric algebra lies in the elegance and the generality (ubiquity) of its formulations, which can be injected within the classical Euclidean framework as well as in differential geometry. In this perspective, concepts and ideas introduced should not replace existing theories and tools, but complement them and shed new light on them.

Based on the work of Grassmann, Clifford's geometric algebras, born in the mid nineteenth-century, consider algebraic operators along with three main products to describe the spatial relations characteristic to geometric primitives in a coordinate-free approach. The many possibilities offered by Clifford algebras and geometric algebras (hereafter denoted GA), and especially their geometrically intuitive aspects, have been emphasized by numerous scientists. For instance, the physicist D. Hestenes has acknowledged their importance to relativistic physics [22]. Likewise, the mathematicians G.-C. Rota [17], I.R. Porteous [27] and J. Snygg [31], among others, have largely promoted the geometric compactness and simplicity of GA, hence contributing to broaden the field to further applications in computer graphics and scientific visualization.

The next section will briefly present the main concepts and the basic manipulation rules of Clifford and geometric algebras. Then, the specific case of vector fields defined on d-dimensional spaces or on differential manifolds will be addressed in Sect. 3. In the last section, we will show how geometric algebra can be efficiently used to understand the algebraic structure of vector fields and implemented (Fig. 1).

2 Clifford and Geometric Algebras

Leaning on the earlier concepts of Grassmann's exterior algebra and Hamilton's quaternions, Clifford intended his *geometric algebra* to describe the geometric properties of vectors, planes and eventually higher dimensional objects. Basically, Clifford algebra for $\mathbb{R}^n$ is the minimal enlargement of $\mathbb{R}^n$ to an associative algebra with unit capturing the metric, geometric and algebraic properties of Euclidean space [16]. In general, geometric algebras are distinguished from Clifford algebras by their restriction to real numbers and their emphasis on geometric interpretation and physical applications.

Note. Our intent in this section is to give an elementary and coherent account of the main concepts of Clifford and geometric algebras. The reader who is interested in the theoretical aspects of geometric algebras is referred to the textbooks [16, 19, 22], among others. Computational aspects of geometric algebra and its usability in research or engineering applications are discussed in [7, 20]. We privileged a continuous and straightforward digest, deliberately avoiding the conventional succession of definitions and theorems commonly found in most textbooks. Furthermore, most of the concepts in this section are presented in a general setting. The material in this section is intended to be fairly basic but readers unfamiliar with abstract mathematical concepts should skip the formal definition, as well as the advanced concepts in Sects. 2.1.2 and 2.2.1.

2.1 Clifford Algebra

Clifford algebra can be introduced in many ways; the approach we follow here separates the algebraic structure from the geometric interpretation of the product.

2.1.1 Basic Notions and Definitions

Formal Definition Let V be a vector space over a field K, and let $Q : V \to K$ be a quadratic form on V. A Clifford algebra $Cl(V, Q)$ is an associative algebra over K, with identity element $\mathbb{1}$, together with a linear map $i : V \to Cl(V, Q)$ satisfying, for all $v \in V$, the *contraction rule* $i(v)^2 = Q(v)\mathbb{1}$, such that the following universal property is fulfilled [24]:

Given any other associative algebra A over K and any linear map $j : V \to A$ such that, for all $v \in V$, $j(v)^2 = Q(v)\mathbb{1}_A$, there is a unique algebra homomorphism $f : Cl(V, Q) \to A$, for which the following diagram commutes:

$$\begin{tikzcd} V \arrow[r, "i"'] \arrow[dr, "j"'] & Cl(V,Q) \arrow[d, "f"] \\ & A \end{tikzcd}$$

Note that the existence and the uniqueness (up to unique isomorphism) of a Clifford algebra for every pair (V, Q) can be established by considering a quotient algebra of a tensor algebra.

The product defining the Clifford algebra will be called *geometric product* and denoted as: $u\,v$, for $u, v \in Cl(V, Q)$ (with a small space between the factors). One usually considers V as a linear subspace of $Cl(V, Q)$, thus dropping the inclusion in the definition of the Clifford algebra, leading $u\,u = u^2 = Q(u)$. Consequently, the vector space V is not closed under multiplication as, for example, $u\,u$ is a scalar and not an element of V. The contraction rule also implies that every $v \in V$ has an inverse $v^{-1} = \frac{v}{Q(v)}$, unless Q is degenerate.

Intuitive Interpretation of the Geometric Product One can classically consider the product of two elements $a, b \in V$ as the sum of a symmetric and an antisymmetric part:

$$a\,b = \underbrace{\frac{1}{2}(a\,b + b\,a)}_{<a,b>} + \underbrace{\frac{1}{2}(a\,b - b\,a)}_{a \wedge b} .$$

– In this setting, the symmetric part $< a, b >$ corresponds to the bilinear form ϕ associated to the quadratic form Q thanks to the polarization identity: $\phi(a, b) = \frac{1}{2}(Q(a + b) - Q(a) - Q(b)) = \frac{1}{2}((a + b)\,(a + b) - a\,a - b\,b) =< a, b >$, this, of course, as a consequence of the contraction rule $v^2 = Q(v)$. When Q is non-degenerate, it is an *inner product*.
– The antisymmetric part $a \wedge b$ has, if non-zero, to be understood as a new entity, that is neither a scalar nor a vector. For Q non-degenerate, the so defined *outer product* has a very simple interpretation: $a \wedge b$, for $a, b \in V$, geometrically represents an oriented plane segment, and can be characterized by an algebraic area (the usual area of the parallelogram with the vectors a and b as sides) and the *attitude* (angular position) of this plane.[1]

The Graded Clifford Algebra Consider again the Clifford algebra $Cl(V, Q)$, V and Q like above. We define G_0 as the inclusion of the scalars K in $Cl(V, Q)$. Given an orthonormal basis $\{e_1, e_2, \dots\}$ of V, let G_n be the part of $Cl(V, Q)$ generated from the products $\prod_{j=1}^{n} e_{i_j}$, for $1 \leq i_1 < \cdots < i_n$. The direct sum $\bigoplus_{n=0}^{\infty} G_n$ is then the *graded* Clifford algebra. The elements of G_n are called *n-vectors*, where n is the *grade*. Elements can be of "mixed grade", like the product $a\,b$ of two elements in V, which is a sum of a scalar (grade 0) and a bivector (grade 2). A *multivector* A can be decomposed as a sum $A = \sum_{r=0}^{\infty} A_r$, where $A_r = \langle A \rangle_r$ is of grade r.

Extension of the Definition of Outer Product The outer product of two multivectors A_k (grade k) and B_ℓ (grade l) is defined as the grade $|k + \ell|$- part of the product $A_k\,B_\ell$, writing $A_k \wedge B_\ell = \langle A_k\,B_\ell \rangle_{k+\ell}$. This product extends by linearity on the whole Clifford algebra. For any $n \leq dim(V)$, *n-blades* are defined recursively as outer products of n vectors $a_1 \wedge \cdots \wedge a_n = (a_1 \wedge \cdots \wedge a_{n-1}) \wedge a_n$. By convention, 0-blades are scalars. A n-blade is a n-vector, but the converse is not true. More precisely [15], a sum of two blades A and B is another blade iff they are of the same grade k **and** share a common factor of grade $k - 1$ or k.

[1]The geometric interpretation of the decomposition of the geometric product in outer and inner products will be explained again for $V = \mathbb{R}^3$ at the beginning of Sect. 2.2.

2.1.2 Advanced Concepts

Factorization of blades with the geometric product yields two equivalent forms for a blade: one based on the outer product, the other on the geometric product. Actually, for any arbitrary quadratic form Q, given a k-blade A_k, it is possible to find an orthogonal basis $\{v_1, \cdots, v_k\}$ of this blade.[2] It implies the double formulation $A_k = v_1 \wedge v_2 \wedge \cdots \wedge v_k = v_1\, v_2 \cdots v_k$. For example, if $a, b \in V$, with $Q(a)$ non-zero, we have $a \wedge b = a \wedge \left(b - \frac{<a,b>}{Q(a)} a\right) = a \left(b - \frac{<a,b>}{Q(a)} a\right)$.

The *meet* and *join* are non-linear operations, corresponding to the blade intersection and union. Suppose we have an orthogonal factorization of two blades A and B, i.e., they are given with their orthogonal factorizations $A = A'\, C$ and $B = C\, B'$, C being the largest common factor. In this very simple case,[3] $M = A \cap B = C$ and $J = A \cup B = (A'\, C) \wedge B'$.

Two important involutions are defined on $Cl(V, Q)$: *reversion* and *grade involution*. On a r-blade $A = (a_1 \wedge a_2 \wedge \cdots \wedge a_r)$, the reversion $A^\dagger$ consists of reversing the order of the constitutive vectors (or, because the outer product is antisymmetric on vectors, changing the sign $r(r-1)/2$ times); the grade involution $A^\#$ consists of reversing the sign of every constitutive vector:

$$A^\dagger = a_r \wedge a_{r-1} \wedge \cdots \wedge a_1 = (-1)^{r(r-1)/2} a_1 \wedge a_2 \wedge \cdots \wedge a_r \qquad A^\# = (-1)^r A\,.$$

The reversion and grade involution extend by linearity on $Cl(V, q)$: if $A = \sum_{r=0}^{\infty} A_r$,

$$A^\dagger = \sum_{r=0}^{\infty} (-1)^{r(r-1)/2} A_r \qquad A^\# = \sum_{r=0}^{\infty} (-1)^r A_r\,.$$

The *even* (resp. *odd*) multivectors are the ones with $A^\# = A$ (resp. $A^\# = -A$).

Using the reversion and the selection of the scalar part $\langle\cdot\rangle_0$, let us define a bilinear form on $Cl(V, Q)$. On blades A_k and B_ℓ, we set:

$$A_k * B_\ell = \begin{cases} \langle A_k^\dagger\, B_\ell\rangle_0 & \text{if } k = \ell \neq 0 \\ A_0 \cdot B_0 & \text{if } k = \ell = 0 \\ 0 & \text{else} \end{cases}\,.$$

Extending it linearly to multivectors A and B, we obtain the general formula $A * B = \langle A^\dagger\, B\rangle_0$. Proof of the equivalence between both formulations can be found in [22, p. 13]. On vectors, this bilinear form clearly corresponds to the inner

[2]A general demonstration (also valid for a degenerate Q) is given for example in [9, p. 88]. In Euclidean spaces, the well-known Gram-Schmidt orthogonalization can be used.

[3]The dualization introduced in Sect. 2.2 makes more general equations for M and J possible.

product: $a * b =< a, b >$. When Q is non-degenerate, it is non-degenerate, and it is sometimes called *Clifford scalar product*.

2.2 Geometric Algebras

The case $V = \mathbb{R}^n$ and Q non-degenerate leads to a series of specific definitions and results. As a matter of fact, we have for example:

- Every non-zero blade A_r has an inverse $\frac{A_r^\dagger}{A * A} = \frac{A_r^\dagger}{\langle A^\dagger A \rangle_0}$.
- If, in addition, Q is positive definite, then we can define the modulus of element A as $|A| = \sqrt{A^\dagger * A} = \sqrt{\langle A^\dagger A \rangle_0}$, since for an element $a_1 \cdots a_r$, $(a_1 \cdots a_r)^\dagger (a_1 \cdots a_r) = Q(a_r) \cdots Q(a_1) \geq 0$.
- In $\mathbb{R}^3$, the existence of an inverse vector has a very clear interpretation. For a given vector $v \in \mathbb{R}^3$ and a given scalar a, the equation $< v, w >= a$ defines the affine plane $w_0 + v^\perp$. Likewise, given v and a bivector A, the equation $v \wedge w = A$ defines the affine line $w_0 + \lambda v$. In both cases, there is no unique solution. However, in the setting of geometric algebra, the equation $v\, w = A$ leads to the unique solution $w = v^{-1} A$ (corresponding to the intersection of a plane $< v, w >= A_0$ and of a line $v \wedge w = A_2$).

Such a Clifford algebra, in the case $V = \mathbb{R}^n$ and Q non-degenerate, is called *geometric algebra*. Let (p, q) be the signature of the quadratic form Q, i.e., Q diagonalizes in $Q(v) = v_1^2 + \cdots + v_p^2 - v_{p+1}^2 - \cdots - v_{p+q}^2$ (Sylvester's law of inertia). We write $\mathbb{R}^{p,q}$ for V and $Cl_{p,q}$ for the associated geometric (Clifford) algebra.

Taking a basis $\{e_1, e_2, \ldots, e_n\}$ of $\mathbb{R}^n$, using the element $\mathbb{1}$ to span the scalars and all products $\prod_{j=1}^{r} e_{i_j}$ for $1 \leq i_1 < \cdots < i_r \leq n (r \in \mathbb{N}_n)$ to span the multivectors, the set $\{\mathbb{1}, e_1, e_2, \ldots, e_n, e_1\, e_2, e_1\, e_3, \ldots, e_1\, e_2\, \ldots\, e_n\}$ will form a basis for $Cl_{p,q}$, with $2^n = \sum_{r=0}^{n} \binom{n}{r}$ elements. The element $\mathrm{I}_n = e_1\, e_2\, \ldots e_n$ is called *pseudoscalar* and is defined to a scalar multiple, since all n-blades are proportional.

2.2.1 Duality and Reciprocal Frames

The dual A^* of a multivector A is defined as $A^* = A\, \mathrm{I}_n^{-1}$. The duality operation transforms a r-vector A_r into an $(n-r)$-vector $A_r\, \mathrm{I}_n^{-1}$; in particular, it maps scalars into pseudoscalars. The duality relation states $(A \wedge B)^* = A \rfloor B^*$, where $\rfloor$ denotes the left contraction.[4] The inclusion of an element x in a given subspace $\mathscr{A}$ specified by a blade A can be defined in two ways:

[4]For two blades A and B of grades a and b, the left contraction $A \rfloor B$ is $\langle A\, B \rangle_{b-a}$ when $a \leq b$, it is zero otherwise. When blade A is contained in blade B, it equals the geometric product $A\, B$ [7].

– the direct way: $x \in \mathscr{A} \iff x \wedge A = 0$
– the dual way: $x \in \mathscr{A} \iff x \rfloor A^* = 0$.

Given a basis $\{b_1, \ldots, b_n\}$ of $\mathbb{R}^{p,q}$, $n = p + q$, we can define a *reciprocal frame* $\{b^1, \ldots, b^n\}$, through the formula $b^i = (-1)^{i-1}(b_1 \cdots \wedge b_{i-1} \wedge \check{b}_i \wedge b_{i+1} \cdots \wedge b_n)\, \mathrm{I}_n^{-1}$, where $\mathrm{I}_n = b_1 \wedge \ldots \wedge b_n$ and the $\check{}$ -sign mentions the element removed from the list. The two basis are mutually orthogonal: $< b_i, b^j >= \delta^i_j$. Since the reciprocal of an orthonormal basis is itself, this definition is needed only in non-Euclidean cases. It is also useful in differential geometry.

A vector of $\mathbb{R}^{p,q}$ can be written $a : \sum_{i=1}^n a_i e^i$ with $a_i =< a, e_i >$ or $a = \sum_{i=1}^n a^i e_i$ with $a_i =< a, e_i >$ and $a^i =< a, e^i >$. If we have a multivector basis $\{e_\alpha | \alpha \in \{1, \cdots, 2^n\}\}$, we can also define a reciprocal frame $\{e^\alpha | \alpha \in \{1, \cdots, 2^n\}\}$.

2.2.2 Versors, Rotors, Spinors and Rotations

One of the main features of GA is its ability to deal with the rotations. Indeed, a unique object R can be used to compute the rotation of any subspace X, writing a conjugation with the geometric product:

$$\mathscr{R}(X) = R\, X\, R^{-1}\,.$$

The equation $x = a\, x\, a^{-1}$ gives the reflection of an arbitrary vector x along the a-line (a invertible). Its opposite $x = -a\, x\, a^{-1}$ gives the reflection in the dual hyperplane $A = a^*$. Two consecutive reflections form a simple rotation, which can be written as follows: $x" = -b\, x'\, b^{-1} = b\, a\, x\, a^{-1}\, b^{-1} = (b\, a)\, x\, (b\, a)^{-1}$. It is a rotation of twice the angle between a and b in the plane containing a and b. The element $a\, b$ is called a *2-versor*. In general, a *k-versor* is a multivector that can be written as the geometric product of k invertible vectors $v = v_1\, v_2 \ldots v_k$. By the Cartan-Dieudonné Theorem [6, 8], every isometry of $\mathbb{R}^{p,q}$ can be reduced to at most $n = p + q$ reflections in hyperplanes. It means that we can write every orthogonal transformation f with a k-versor U $(k \leq n)$ and the conjugation: $f(x) = (-1)^k U\, x\, U^{-1}$.

In all spaces of signatures $(n, 0)$, $(0, n)$, $(n-1, 1)$ or $(1, n-1)$, including the Euclidean spaces, every rotation can be written in exponential form[5]:

$$\mathscr{R}(x) = S\, x\, S^\dagger \text{ with } S = e^{\frac{1}{2}(i_1\, \theta_1 + \cdots + i_m\, \theta_m)}, i_1, \cdots, i_m \text{ orthogonal 2-blades}\,.$$

Note that a rotation of a non-Euclidean space is defined to be an orthogonal transformation of determinant one continuously connected to identity. The element

[5]Quite naturally, the exponential of a blade A is defined with the usual power series $\sum_{k=0}^{\infty} \frac{A^k}{k!}$. The additivity $\exp(A + B) = exp(A)\, exp(B)$ is not true in general. The circular and hyperbolic functions of blades are also defined with power series.

S given by the exponential form of preceding equation is a *rotor*, i.e., an even versor S satisfying $S\,S^{\dagger} = 1$.

A linear map f: $V \rightarrow V$ can be extended in a function $\underline{f} : Cl(V, Q) \rightarrow Cl(V, Q)$ while preserving the outer product:

$$\underline{f}(a_1 \wedge a_2 \wedge \cdots \wedge a_r) = f(a_1) \wedge f(a_2) \wedge \cdots \wedge f(a_r)\,.$$

It is then called an *outermorphism*. In particular, the reflection of a blade A_k in a dual hyperplane a^* is $(-1)^k a\,A_k\,a^{-1}$ and the rotation of a blade by a rotor is $RA_k R^{\dagger}$ according to the previous equations for vectors.

2.2.3 Geometric Calculus

Differentiation We consider a finite-dimensional vector space V with quadratic form Q and a multivector-valued function $F : U \subset V \longrightarrow Cl(V, Q)$. It comes of no surprise that the directional derivative of F in direction r is simply:

$$F_r(x) = \lim_{s \rightarrow 0} \frac{F(x + s\,r) - F(x)}{s}\,.$$

This expression will be most of the time written $(r * \nabla)F$ instead of F_r, expressing the idea of a scalar product between r and the operator ∇, seen as a vector, as will be clearer below. The linearity in r is straightforward; the sum, the geometric product and the grade are preserved. If we want to differentiate a multivector-valued function $F : U \subset V \longrightarrow Cl(V, Q)$ directly relative to the variable, we consider a base $\{e_1, \cdots, e_m\}$ of V and the coordinate functions of the vector x in this basis $x = \sum_{i=1}^{m} x^i e_i$. The directional derivatives along the basis directions, $(e_i * \nabla) = \frac{\partial}{\partial_{x_i}}$, combine into a total change operator[6] as:

$$\nabla = \sum_{i=1}^{m} e^i\,(e_i * \nabla) \quad \text{meaning} \quad \nabla\,F(x) = \sum_{i=1}^{m} e^i\,\frac{\partial F(x)}{\partial_{x_i}}\,.$$

Note that we also have to define the differentiation from the right, because of the non-commutativity: for a function F, $F(x)\,\nabla = \sum_{i=1}^{m} (e_i * \nabla)F(x)\,e^i = \sum_{i=1}^{m} \frac{\partial F(x)}{\partial_{x_i}}\,e^i$. Thanks to the geometric product, we can write ∇ as follows: $\nabla\,F = \nabla \wedge F + < \nabla, F >$. In the case of a vector-valued function F, we have the usual definitions of the divergence and curl operators:

$$\text{curl}(F) := \nabla \wedge F = \frac{1}{2}(\nabla\,F - F\,\nabla) \quad \text{and} \quad \text{div}(F) := < \nabla, F > = \frac{1}{2}(\nabla\,F + F\,\nabla)\,.$$

[6] This explains the notation $F_r = (r * \nabla)F$ for the directional differentiation.

To write the product rule, accents are necessary to specify on what factor the differentiation acts: $\nabla(F\,G) = \grave{\nabla}\grave{F}\,G + \grave{\nabla}F\,\grave{G}$. The definition of a differentiation with respect to a multivector, for a function $F : U \subset Cl(V,Q) \longrightarrow Cl(V,Q)$, is quite straightforward, given a reciprocal frame for the whole space $Cl(V,Q)$.

Integration Consider again a multivector-valued function F; the line integral is

$$\int_C F(x)\,dx = \lim_{n\to\infty}\sum_{j=1}^{n}\overline{F}^j\,\Delta x^j\,, \quad \text{with} \quad \overline{F}^j = \frac{1}{2}\,(F(x_i) + F(x_{i-1}))$$

where the chords $\Delta x^i = x_i - x_{i-1}$ correspond to a subdivision of the curve C. The measure dx is said to be a *directed measure*, since it is vector-valued. The product between $F(x)$ and dx is the geometric product. If F is vector-valued,

$$\int_C F(x)\,dx = \int_C < F(x), dx > + \int_C F(x) \wedge dx\,.$$

Similarly, if $D \subset \mathbb{R}^2$ is a triangulated planar domain, $\overline{F}^k$ is the average of F over the k-th simplex,

$$\int_D F(x)\,dx = \lim_{n\to\infty}\sum_{k=1}^{n}\overline{F}^k\,\Delta x^k\,.$$

The surface measure of the k-th simplex given by vertices x_0, x_1, x_2 is

$$\Delta x^k = \tfrac{1}{2}\,(x_0 \wedge x_1 + x_2 \wedge x_0 + x_1 \wedge x_2)\,.$$

This integral definition can be generalized to higher dimensions [5].

The fundamental theorem states:

$$\oint_{\partial V} F\,dS = \int_V \grave{F}\,\grave{\nabla}\,dX\,, \quad \text{for a function } F \text{ defined over a volume } V\,.$$

2.2.4 Clifford Convolution and Clifford Fourier Transform

For F and H two multivector-valued functions $F, H : \mathbb{R}^m \to Cl_{p,q}$, the left- and the right-Clifford Convolution of the functions write respectively:

$$(H *_\ell F)(x) = \int_{\mathbb{R}^m} H(x')F(x-x')|dx'|\,, \quad (H *_r F)(x) = \int_{\mathbb{R}^m} H(x-x')F(x')|dx'|\,.$$

The quantity $|dx|$ is used to make the integral grade-preserving since dx is a vector within Clifford algebra. Modifying $x - x'$ into $x + x'$, we get the left- and right-*Clifford correlations* [11]. The Clifford convolutions generalize the known convolution of scalar-valued functions.

A vector field F can be smoothed through convolution with a scalar field, for example a Gaussian kernel. In the case of two vector fields, the formula for the geometric product leads to the integration of a scalar function $< H(x-x'), F(x') >$ and a bivector function $H(x-x') \wedge F(x')$ [29].

In the case of a multivector-valued function $F : \mathbb{R}^3 \to Cl_{3,0}$, it is possible to define the *Clifford Fourier Transform* (CFT) of F and its inverse as follows:

$$\mathscr{F}\{F\}(u) = \int_{\mathbb{R}^3} F(x)\, e^{-2\pi \mathrm{I}_3 <x,u>} |dx| \,, \quad \mathscr{F}^{-1}\{F\}(x) = \int_{\mathbb{R}^3} F(u) e^{2\pi \mathrm{I}_3 <x,u>} |du| \,.$$

The function $e^{-2\pi \mathrm{I}_3 <x,u>} = \cos(2\pi < x, u >) + \mathrm{I}_3 \sin(2\pi < x, u >)$ is often called *Clifford Fourier kernel.*

The convolution theorem is also valid for the Clifford Fourier Transform and Clifford convolutions as defined here. For example, using the left convolution,

$$\mathscr{F}\{H *_\ell F\}(u) := \mathscr{F}\{H\}(u)\, \mathscr{F}\{F\}(u) \,.$$

As mentioned before, the reader willing to get a deeper understanding of the mathematical basics about Clifford algebras and geometric algebras is referred to [16, 19, 22]. In the next section, we will focus on the analysis of vector fields in the context of GA.

3 Vector Fields in Geometric Algebra

Our main focus in this paper is the analysis of vector fields, more precisely of steady, linear and non-linear vector fields in Euclidean space and on manifolds. One is classically interested in streamlines, critical points with their classification, separatrices, leading to the topological graph of a vector field. We will show how the analysis of vector fields can benefit from the richer context of geometric algebra.

3.1 Vector Fields on Domains of Euclidean Space

Classically, vector fields are mappings of the form $v : U \subset \mathbb{R}^n \to \mathbb{R}^n$, where U is an open set. Using the notions of GA defined in Sect. 2.2, if $Cl^1_{n,0}$ is the set of 1-vectors of $Cl_{n,0}$, a map $v : \mathbb{R}^n \longrightarrow Cl^1_{n,0}$ is also a vector field. This definition can be easily extended to bivector, trivector, or spinor fields, for example.

The identification of vector fields (satisfying the Lipschitz regularity condition) with ordinary differential equations ${}^{dx}/_{dt} = v(x)$ can also be transposed from the

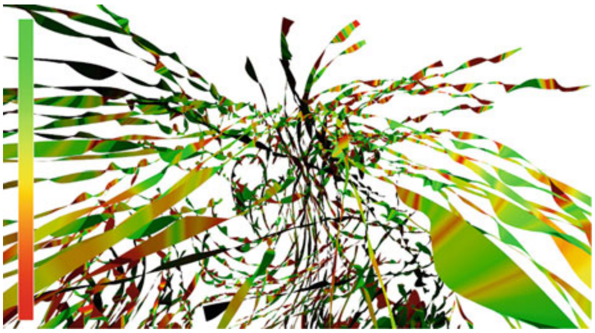


Fig. 2 Frenet ribbons constructed from a discrete vector field. The colour encodes the torsion (reprinted from [4])

classical to the GA setting. The Lipschitz continuity condition can be written in this frame, i.e., there exists a scalar constant $K \geq 0$ such that

$$\|v(x_1) - v(x_2)\|_n \leq K\|x_1 - x_2\|_n \quad \text{for all} \quad x_1, x_2 \in U\,.$$

Furthermore, the defined derivation and integration make it possible to state the existence of an unique solution (*streamline* or *integral curve*) through a given point, exactly like in the classical frame. In 2D and 3D, drawing the streamlines is a very classical way to represent a vector field. In order to avoid occlusions and empty areas, departure points (*seeds*) for these curves are to be placed efficiently.

Let us consider a small example. To a given classical vector field, we can associate curvature and torsion scalar fields: the curvature (resp. torsion) in a point is the curvature (resp. torsion) of the unique streamline in this point [35]. The curvature field associated to a vector field can be used for the seeding, or can be displayed as a further scalar value in the form of isosurfaces or by color coding. In the GA settings, instead of scalar fields, a curvature bivector field and a torsion trivector field can be defined. Visualizing the curvature bivector along a streamline, we get what is called the *Frenet ribbon* [4], see Fig. 2 for such a representation of the vector field.

3.2 Vector Fields on Differential Manifolds

Now we turn to vector fields on differential manifolds, having in mind to embed the differential geometry formalism into geometric calculus. For a more detailed presentation of this combined approach, see [22, 31].

In differential geometry, if M is a smooth manifold, a tangent vector in $p \in M$ is a derivation, i.e., a linear operator D on C_p^∞ (the algebra of germs of smooth functions at p) satisfying the Leibniz rule $D(f \cdot g) = f \cdot D(g) + g \cdot D(f)$. The tangent space of M in p is T_pM, the set of such derivations. A vector field is a function assigning to every $p \in M$ an element of T_pM.

In $\mathbb{R}^3$, in a more intuitive way, we can imagine giving in each point p of a surface S a vector tangent $v(p)$ to the surface. The link between this v and the associated derivation D_v is the derivative $D_v(f)(p) = Df(p)(v(p))$. The operator point of view makes it easier to manipulate vector fields and compose them with other operators. Furthermore a discretization can be made without working with coordinates [1].

To translate this definition into GA, we give the tangent spaces a Clifford algebra structure. Taking a chart (U, ϕ) around $p \in M$, the derivations e_i^p defined by

$$e_i^p(f) = \frac{\partial}{\partial x_i}(f \circ \phi^{-1})|_{x=\phi(p)}$$

form a basis for T_pM. Forming the blades of these basis vectors, we can build a geometric algebra structure on T_pM.

With a little more abstraction, a vector field can classically be seen as a *section* of the *tangent bundle*, a particular *vector bundle*: Taking TM to be the disjoint union of tangent spaces on M, $TM = \sqcup_{x \in M} T_xM$, and $\pi : TM \to M$ defined by $\pi(v) = x$ for $x \in T_{xM}$, we can see M as the base space, TM as the total space and π as the projection, these three elements defining a fibre bundle called the tangent bundle. The section is a continuous map s with $\pi \circ s = id_M$, meaning $s(x) \in T_xM$, hence what we understand as a vector field. The adding of a geometric algebra structure can be done in the general case of a vector bundle on a manifold with some metrics, using a construction very similar as the one made in Sect. 2.1.1: quotienting a tensor algebra with a two-sided ideal.

Scalar fields, vector fields, bivector fields, spinor fields on surfaces, for example, are natural extensions of this definition of vector fields (or can be seen as sections of the Clifford tangent bundle, see above), and, as long as M is simply connected, it is also the case for rotation fields $r : M \to SO(n)$, since they can be lifted to spinor fields. Since every differentiable manifold is locally diffeomorphic to an Euclidean space (via the charts), the existence and uniqueness of streamlines is also granted on manifolds, within or outside GA context.

3.3 Critical Points, Orbits and Topological Graph

The topological graph is an important tool of analysis: it goes one step further than the streamline representation and decomposes the vector field domain into regions of similar behavior. The critical points and closed orbits (with their type, like defined below) and the separatrices (streamlines or surfaces between areas of different behavior) form the topological graph of the vector field, that eventually describes the underlying structure of the field in a more efficient way as a collection of streamlines. Such a graph does not take into account the norm of the vector field [12].

The classification of critical points finds its origin in the theory of dynamical systems. For regular critical points, i.e., for critical points with an invertible Jacobian matrix, a linear approximation of the field can be considered. Studying eigenvalues and eigenvectors of the Jacobian matrix makes the classification possible, provided none of the eigenvalues is pure imaginary. The so-called hyperbolic critical points, satisfying this condition, are isolated and are structurally stable: a small local perturbation does not modify the topology. This justifies the use of the linear approximation to describe the field's behavior around this point. In two dimensions for example, the hyperbolic critical points are sources, sinks, saddles and spirals. Unstable critical points are centers. A similar classification can be done for orbits, according to the derivative of the Poincaré map [2]. For non-linear critical points, said to be of *higher order*, the non-invertibility of the first derivative leads one to consider a higher order Taylor expansion. For the isolated ones, the index[7] can help discriminate critical points of different types. Sometimes this proves insufficient, since two critical points with same index can be of different types. The GA formalism provides an elegant alternative for the computation of the index: for example, in 3D,

$$ind(c) = \frac{1}{8\,\pi\,\mathrm{I}_3} \int_{B(c)} \frac{v \wedge dv}{|v|^3}\,,$$

for v the vector field, c the critical point, $B(c)$ an arbitrary small ball around c [25]. A corresponding discrete computation will be introduced in Sect. 4.2. Unlike the index, the ordered list of all different behavior sectors (i.e., *elliptic*, *hyperbolic* and *parabolic sectors*) makes an unambiguous classification possible [14,18,33,34] (see Fig 3).

Next, we turn to a more practical view of geometric algebras, as this chapter is also intended for engineers and practitioners. In particular, we will briefly explain how GA can be implemented and the potential advantages of using Clifford algebra when, for example, dealing with rotations in spaces of high dimensions.

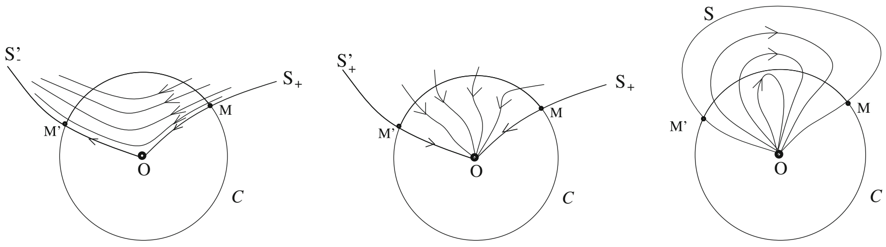


Fig. 3 Classification of sectors: hyperbolic, parabolic and elliptic sectors (reprinted from [33])

[7]In 2D, the index corresponds the number of turns the field makes around a critical point.

4 Geometric Algebra for Computer Graphics and Visualization of Vector Fields

Nowadays, geometric algebra is mostly recognized as a promising mathematical concept and is beginning to find broader application. Emerging computer architectures (multicore, many-core, parallel) lead us to believe that the language of GA may find a new playground and evolve towards what Hildenbrand calls Geometric Algebra Computing (GAC) [21]. However, GA is not yet a widespread method in engineering applications, mainly because of two reasons, academic and practical [26]. On the one hand, GA combines many mathematical concepts that were developed separately over the years and are taught as such in curriculum. On the other hand, most engineering applications in three-dimensional space can be dealt using standard vector and matrix algebra tools. The goal of this last section is to introduce how GA can be used advantageously in computer graphics applications and vector field analysis and visualization.

4.1 Geometric Algebra for Computer Graphics

Computer graphics is surely the most obvious field of application of GA. In geometrical applications, operations and transformations are applied on primitives that are combined to represent an object (model). Linear geometric transformations are usually represented using matrices, vectors and scalars. But while 3×3 matrices encode the 3D-rotations about an axis through the origin, quaternions are better suited instead, because they are easier to interpret.

The quaternion representation of a rotation is a nearly minimal parametrization that requires only four scalars. Given a quaternion, one can easily read off the axis and angle of the rotation, it is not the case with the Euler angles representation. The composition of rotations in quaternion form is faster and more stable numerically (the renormalization is more efficient than with matrices). Furthermore, the interpolation in the set of quaternions $\mathbb{H}$ (for example to get an animated view of a rotated object) consists in defining a path on S^3 which is mapped to $SO(3)$. The Euler angles parametrization, from the 3-torus to $SO(3)$ is not a local diffeomorphism: problem known as the *gimbal lock*. This is why, in graphic libraries such as OpenGL, rotations are given in terms of a rotation axis and a rotation angle and converted internally into rotation matrices.

Note that $\mathbb{H}$ forms a subalgebra of a geometric algebra [27]: it is trivially isomorphic to the even algebra $Cl^{+}_{3,0}$ (the set of even multivectors of $Cl_{3,0}$). We can identify the unit and the basis elements $\mathbf{i}, \mathbf{j}, \mathbf{k}$ of $\mathbb{H}$ with the unit and the products $e_1 e_2$, $e_1 e_3$ and $e_2 e_3$ of $Cl_{3,0}$ [19]. Thus, the aforementioned quaternion representation of a rotation (of angle θ around unit axis u):

$$q = e^{\frac{1}{2}\theta(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} = \cos\tfrac{1}{2}\theta + (u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})\sin\tfrac{1}{2}\theta \quad \text{with} \quad \mathscr{R}(x) = q\,xq^{-1}\,,$$

can be seen as a rotor of $Cl_{3,0}$. The products and the conjugation on $\mathbb{H}$ and on $Cl^+_{3,0}$ are defined likewise. Similarly, identifying the pseudoscalar I of $Cl_{2,0}$ with the imaginary unit i of $\mathbb{C}$, we have an algebra isomorphism between $Cl^+_{2,0}$ and $\mathbb{C}$.

Clearly, geometric algebra exhibits structural simplicity in the formulations, but its naive implementation may be far less efficient than classical analytical geometry implementations, especially for high dimensions. Fortunately, GA expressions can benefit from compilation (e.g. operator overloading) and parallelization techniques (including GPUs) [15, 21].

In practice, GA has been implemented in two ways. The *additive approach* encodes each multivector of $Cl_{n,0}$ with its 2^n coordinates. It leads typically to a $O(2^{2n})$ time complexity for linear operations and for products (inner, outer and geometric), and to a storage complexity in $O(2^n)$. The *multiplicative approach*, restricted to blades, stores the coordinates of the unit orthogonal factors in a matrix and the magnitude using a scalar. Although the storage complexity is smaller than in the additive approach, there is still no available strategy for an efficient implementation of addition. Nevertheless, factorization and simplification operations allow a trivial implementation of "meet" and "join" operations.

As suggested by its name, the conformal model $Cl_{4,1}$ of $\mathbb{R}^3$ can be used to represent various angle-preserving geometries. In this model, all conformal transformations can be represented by versors, especially the ones preserving the Euclidean distances.

To define the conformal model $Cl_{4,1}$ of $\mathbb{R}^3$, two vectors e_+ and e_- are adjoined to the basis vectors e_1, e_2, e_3 of $\mathbb{R}^3$ embedded in $\mathbb{R}^5$. They are chosen to form an orthogonal basis, with ${e_+}^2 = Q(e_+) = 1$ and ${e_-}^2 = Q(e_-) = -1$. If we define respectively $n_0 = {}^1\!/\!_{\sqrt{2}}(e_- + e_+)$ and $n_\infty = {}^1\!/\!_{\sqrt{2}}(e_- - e_+)$, the new basis $\{e_1, e_2, e_3, n_0, n_\infty\}$ is not orthogonal ($< n_0, n_\infty >= -1$), but makes intuitive definitions for the model possible. The representation p of a point $p_b \in \mathbb{R}^3$ in the conformal model is defined by the following mapping:
$p = F(p_b) = p_b + n_0 + \frac{1}{2}\, p_b^2 n_\infty$.

The element n_0 has the same translation role as the origin vector e_0 in the homogeneous model. The vector n_∞ represents the point at infinity and the axis of symmetry of the *horosphere*, the set of elements defined by this equation. The Euclidean distance between two points $p_b, q_b \in \mathbb{R}^3$ is directly proportional to the squared root of the inner product $< F(p_b), F(q_b) >$ of their representations in the model $Cl_{4,1}$. The horosphere is formed of *null vectors*, i.e., vectors of zero norm, as consequence of the fact that $p^2 =< p, p >$ is proportional to $(p_b - p_b)^2 = 0$.

The spheres, planes, circles and lines of $\mathbb{R}^3$ can be expressed in the conformal model space $Cl_{4,1}$ with two different conditions, using the inner or the outer product. For the example of the sphere:

- A sphere $S(a_b, r)$ centered in a_b, with radius r corresponds to: $\tilde{S}(a_b, r) = F(a_b) - {}^1\!/\!_2 r^2 n_\infty \in Cl_{4,1}$ with $p_b \in S(a, r) \Longleftrightarrow < F(p_b), \tilde{S}(a_b, r) >= 0$.

– The sphere containing the four points $a_b, b_b, c_b, d_b \in \mathbb{R}^3$ corresponds to the element: $S = F(a_b) \wedge F(b_b) \wedge F(c_b) \wedge F(d_b) \in Cl_{4,1}$ with $p_b \in S \iff F(p_b) \wedge S = 0$.

Since any vector $x \in Cl_{4,1}$ can be written $x = F(a_b) \pm {}^1\!/_2 r^2 n_\infty$, for an $a_b \in \mathbb{R}^3$ and a $r \in \mathbb{R}$, the building blocks of $Cl_{1,4}$ are spheres, points (spheres with radius zero) and imaginary spheres (spheres with imaginary radius). The reflection in an hyperplane corresponds to a conjugation by a vector in $Cl_{4,1}$. To the other transformations, translations, rotations and scalings, correspond rotors in exponential form (e.g. $T = e^{-{}^1\!/_2 t_b n_\infty}$ for the translation of vector t_b). All orthonormal transformations can be expressed by rotors, since translations enjoy this property.

4.2 Geometric Algebra for the Visualization of Vector Fields

For the sake of clarity, we restrict ourselves here to 2D and 3D vector fields or vector fields defined on surfaces embedded in $\mathbb{R}^3$. The objective is to show that GA allows one to perform the local analysis of the fields using differential geometry in a rather classical way, but offers more flexibility and efficiency when identifying the global structures.

With vector data defined at the vertices of a simplicial triangulation T_h or of a regular sampling (Cartesian grid), discrete equivalents of geometric and topological entities (e.g. curve, ball) are needed, as well as interpolations, giving vector values at arbitrary locations. This can be achieved in several ways but requires special attention to avoid ambiguous or non-conformal situations [23].

To compute the topological index in 2D, we recast the formulation given in Sect. 3.3 in a discrete setting [18]. Let $B(c)$ denote a closed polygonal curve around the critical point. For every couple of neighbor vertices (p_1, p_2), form the bivector ${}^1\!/_2(\tilde{v}(p_1) \wedge \tilde{v}(p_2))$ with the values of the normalized vector field $\tilde{v} = {}^v\!/_{\|v\|}$. The sum of all bivectors, divided by the volume of the unit disk π, will give an approximation of the winding number of v on the curve, which is in turn an approximation of the index of v in c.

It can be shown that two closed polygonal curves discretizing the same underlying continuous curve lead to the same winding number, as long as they are ϵ-dense (i.e., any point of the continuous curve between two neighbors will be within ϵ-distance of both neighbors). In a continuous setting, the index of a critical point is well defined as the winding number of every circle containing this only critical point, since a nonvanishing vector field v in the interior of a closed path γ implies a zero winding number of v on γ.

A similar computation can be done for 3D vector fields, on a triangulated surface around the critical point (see Fig. 1). For a triangle of neighbor vertices (p_1, p_2, p_3) on this surface, form the trivector ${}^1\!/_6(\tilde{v}(p_1) \wedge \tilde{v}(p_2) \wedge \tilde{v}(p_3))$ with the values of the normalized vector field. The normalization factor is $\frac{4}{3}\pi$ for the volume of the unit ball [7, 25]. For a vector field on a surface, the computation is less straightforward than in 2D, since vectors should be projected on a plane, before the sum is computed.

A common aforementioned technique in visualization is to integrate the vector field along a curve, the *integral line* (or streamline in a fluid). Given a Lipschitz continuous vector field v defined on an open subset $U \subset \mathbb{R}^m$, one defines curves $\gamma(t)$ on U such that for each t in an interval I, $\gamma'(t) = v(\gamma(t))$. Picard's theorem states that there exists a unique C^1-curve γ_x for each point x in U, so that $\gamma_x(0) = x$, and $\gamma_x'(t) = v(\gamma_x(t))$ for $t \in (-\epsilon, +\epsilon)$. These curves partition the set U into equivalence classes.

Numerically, the discretization of streamlines relies on an integration method; Euler or Runge-Kutta methods are the most common schemes to advance a point along the integral curve given its previous location and a time step δt. Any such method requires to interpolate the field vector at a new location x. The interpolation, defined on classical vector fields using barycentric coordinates, can be written exactly the same way for GA vector fields $v : \mathbb{R}^m \rightarrow Cl_{m,0}$ ($m = 2, 3$). For example, if x is contained in a simplex then the linear interpolate reads: $v(x) = \sum_{i=1}^{l} \lambda_i v_i$, where v_i (resp. λ_i) denotes the values of v at the simplex vertices (resp. corresponding barycentric coordinates). Note that the interpolation of a vector field v defined on a triangulated surface S is not straightforward, since the interpolated vectors need to be defined in the tangent planes.

Not every characteristic of the field lies in the topological graph: features such as vortices, shear zones, shock waves, attachment lines or surfaces are not captured in this description and are very important elements to specify the structure of a vector field. The computation methods reviewed in [28] to extract features in vector fields are presented in the classical frame but can be extended naturally to the GA frame. Several scalar fields deliver information on the presence of vortices: the vorticity magnitude, the helicity magnitude, the pressure for example. For instance, the vorticity is exactly half of the curl defined in GA.

In some specific situations, the vector field may exhibit local patterns with repetitions over the domain. Their localization would help to apprehend the overall structure of the field. For example, in 2D, we could look for the repetition of singularities like monkey saddles, zones with axis drain, or S-shaped zones. The following approach is inspired by image processing.

Correlation. Given a 2D (resp. 3D) pattern, i.e., a vector field defined on a small square (resp. cubic) domain, we can compute the Clifford correlation (introduced in Sect. 2.2.4) between this pattern and a vector field. At each point of the domain, this function gives the similarity of the vector field (in the neighborhood of this point) with the given pattern [10]. The correlation implies a convolution (quadratic complexity), which can be replaced, via Clifford Fourier Transform, by a multiplication (linear complexity) in the frequency domain. Furthermore, since the 3D CFT can be written as a sum of four complex Fourier transforms through the identification of the pseudoscalar I_3 with the imaginary unit i, Fast Fourier Transforms can be used. However, the main drawback of this method is related to the necessity to check the presence of a given pattern in all positions, for many scales and in many orientations, or the search of the pattern will not be complete.

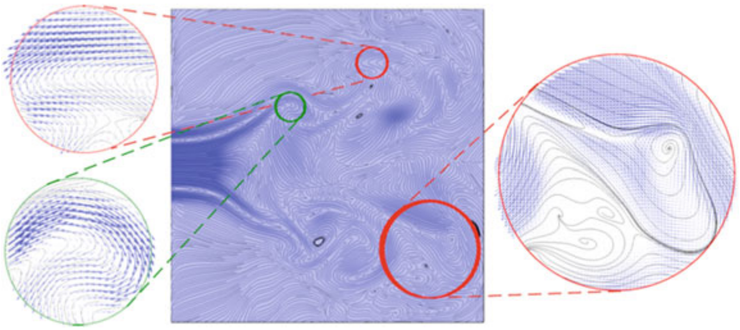


Fig. 4 Search for an S-shaped pattern in a 2D swirling jet flow dataset. The original pattern is shown in a *green circle*, whereas the found occurrences are shown in *red circles*, overlapping for different scales. The method used is the comparison of computed values for a family of moment invariants on the dataset, comparing with the tabulated values for the pattern (reprinted from [30])

Invariants. Suppose that we have again a particular feature (*patch*) we want to identify in a given vector field. Let us attribute values to the different patches through a mapping. Such a mapping, if it exhibits rotation, translation and scale invariance is called shortly *RTS-invariant*. If it is, for example, not rotation invariant, then its value has to be computed for all rotated variants of the patch of interest.

A family of RTS-invariants and non-redundant *moments* of order $\leq d$ [30] can be built for 2D scalar and vector fields, using the complex numbers to get a nice formulation of the rotation invariance in the equations. In Fig. 4, showing a 2D swirling jet flow dataset, the occurrences of a given S-shaped pattern can be seen, as obtained by this method. For 3D scalar functions, one of the ways of defining such moments is to use the spherical harmonic functions as building bricks. To extend to 3D vector fields, complex numbers are no help anymore, and quaternions generate a dimension 4 algebra. If the nice formulation of rotations in Clifford algebra and the existence of a product of vectors seems to pave the way for this generalization, the defining of building bricks (perhaps with the spherical vectorial harmonics) for the moments is the first difficulty, followed by the formulation of a rotation invariance condition. To our knowledge, the extension has not been written yet.

Several alternatives to moments as RTS-invariants are defined in literature. For example, the *harmonic power spectrum* and *harmonic bispectrum* defined in [13] for 3D vector fields rely on spherical vectorial harmonics. The theory is explained in the classical frame, using representation theory, but possibly further invariants could be defined and a substantial gain of clarity could be achieved if using GA.

Heat equation. On a Riemannian manifold M, consider the Clifford bundle obtained from the tangent bundle. The Riemannian metric $g_{ij}(p) =< e_i^p, e_j^p >$, since positive definite, leads to Euclidean tangent spaces. Let us define now a connection on the manifold ∇^E compatible with the metric (for example the Levi-Cevita connection)

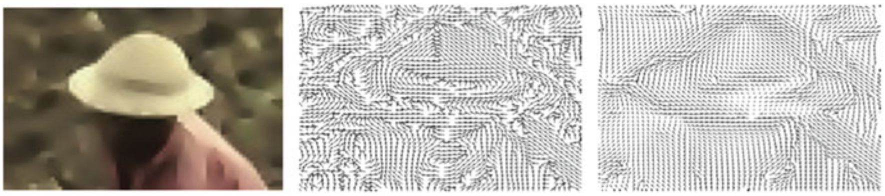


Fig. 5 A color image (*left*) with the corresponding unit vector field of edge orientations (*middle*) and a Clifford-Beltrami regularization of this vector field (reprinted from [3])

and extend it as ∇^C to the Clifford space such that it preserves the graduation, we define a generalized Laplacian as follows:

$$\Delta^C = \sum_{ij} g_{ij}(\nabla^C_{e_i}\nabla^C_{e_j} - \sum_k \Gamma^k_{ij}\nabla^C_{e_k}) .$$

Considering the heat equation $\frac{\partial s_t}{\partial t} + \Delta^C s_t = 0$, with initial condition $s_0 = s$, associated with these operators, the solution is a regularization of the section s. It can be approximated through the convolution with the heat kernel. Varying the operators (Clifford-Hodge, Clifford-Beltrami), different flows are obtained, leading to different regularizations. This approach was introduced in [3], and was applied to reducing noise in color images, see Fig. 5, but not yet, to the best of our knowledge, as a global approach tool for vector fields.

In addition to regularization, heat kernel signatures, like they are defined for scalar fields [32], could be used to define signatures of vector field patches.

Acknowledgements This work undertaken (partially) in the framework of CALSIMLAB is supported by the public grant ANR-11-LABX-0037-01 overseen by the French National Research Agency (ANR) as part of the "Investissements d'Avenir" program (reference: ANR-11-IDEX-0004-02).

References

1. Azencot, O., Ben-Chen, M., Chazal, F., Ovsjanikov, M.: An operator approach to tangent vector field processing. In: Computer Graphics Forum, vol. 32, pp. 73–82. Wiley Online Library (2013)
2. Asimov, D.: Notes on the topology of vector fields and flows. Technical Report, NASA Ames Research Center (1993)
3. Batard, T.: Clifford bundles: A common framework for image, vector field, and orthonormal frame field regularization. SIAM J. Imag. Sci. **3**(3), 670–701 (2010)
4. Benger, W., Ritter, M.: Using geometric algebra for visualizing integral curves. GraVisMa (2010)
5. Bromborsky, A: An introduction to geometric algebra and calculus (2010)
6. Cartan, E: Leçons sur la théorie des spineurs..., vol. 643. Hermann & cie (1938)

7. Dorst, L., Fontijne, D., Mann, S: Geometric Algebra for Computer Science (Revised Edition): An Object-Oriented Approach to Geometry. Morgan Kaufmann (2009)
8. Dieudonné, J.: Sur les groupes classiques. Hermann & cie (1948)
9. Doran, C., Anthony, A., Lasenby, N.: Geometric algebra for physicists. Cambridge University Press, Cambridge (2003)
10. Ebling, J., Scheuermann, G.: Clifford fourier transform on vector fields. IEEE Trans. Vis. Comput. Graph. **11**(4), 469–479 (2005)
11. Ebling, J., Scheuermann, G.: Clifford convolution and pattern matching on irregular grids. In: Scientific Visualization: The Visual Extraction of Knowledge from Data, pp. 231–248. Springer (2006)
12. Ebling, J., Wiebel, A., Garth, C., Scheuermann, G.: Topology based flow analysis and superposition effects. In: Hauser, H., Hagen, H., Theisel, H (eds) Topology-Based Methods in Visualization, Mathematics and Visualization, pp. 91–103. Springer (2007)
13. Fehr, J.: Local rotation invariant patch descriptors for 3d vector fields. In: 20th International Conference on Pattern Recognition (ICPR), pp. 1381–1384. IEEE (2010)
14. Firby, P.A., Gardiner, C.F.: Surface Topology, 2nd edn. Ellis Horwood Series in Mathematics and Its Applications. Horwood (1991)
15. Fontijne, D.: Efficient implementation of geometric algebra. PhD thesis, University Amsterdam (2007)
16. Gilbert, J.C.: Clifford Algebras and Dirac Operators in Harmonic Analysis. Cambridge Studies in Advanced Mathematics (1991)
17. Grosshans, F.D., Rota, G.-C., Stein, J.A.: Invariant Theory and Superalgebras. Number 69. American Mathematical Soc. (1987)
18. Henle, M.: A Combinatorial Introduction to Topology. Books in Mathematical Sciences. Freeman (1979)
19. Hestenes, D.: New Foundations for Classical Mechanics. Springer (1999)
20. Hildenbrand, D., Fontijne, D., Perwass, Ch., Dorst, L.: Geometric algebra and its application to computer graphics. In: Tutorial Notes of the EUROGRAPHICS Conference (2004)
21. Hildenbrand, D.: Conformal geometric algebra. In: Foundations of Geometric Algebra Computing, pp. 27–44. Springer (2013)
22. Hestenes, D., Sobcyk, G.: Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics. Kluwer Academic (1987)
23. Linial, N.: Finite metric spaces, combinatorics, geometry and algorithms. In: Proceedings of the ICM, Beijing,, vol. 3, pp. 573–586 (2002)
24. Mahmoudi, M.G.: Orthogonal symmetries and clifford algebras. In: Indian Academy of Sciences Proceedings-Mathematical Sciences, vol. 120, p. 535 (2011)
25. Mann, S., Rockwood, A.: Computing singularities of 3d vector fields with geometric algebra. In: Visualization, 2002 (VIS 2002), pp. 283–289. IEEE (November 2002)
26. Perwass, C.: Geometric Algebra with Applications in Engineering, vol. 4. Springer (2009)
27. Porteous, I.R.: Clifford Algebras and the Classical Groups, vol. 50. Cambridge University Press, Cambridge (1995)
28. Post, F.H., Vrolijk, B., Hauser, H., Laramee, R.S., Doleisch, H.: The state of the art in flow visualisation: Feature extraction and tracking. In: Computer Graphics Forum, vol. 22, pp. 775–792. Wiley Online Library (2003)
29. Reich, W., Scheuermann, G.: Analyzing real vector fields with clifford convolution and clifford–fourier transform. In: Bayro-Corrochano, E., Scheuermann, G. (eds) Geometric Algebra Computing, pp. 121–133. Springer, London (2010)
30. Schlemmer, M., Heringer, M., Morr, F., Hotz, I., Bertram, M.-H., Garth, C., Kollmann, W., Hamann, B., Hagen, H.: Moment invariants for the analysis of 2d flow fields. IEEE Trans. Vis. Comput. Graph. **13**(6), 1743–1750 (2007)
31. Snygg, J.: A New Approach to Differential Geometry Using Clifford's Geometric Algebra. Springer (2011)

32. Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. In: Computer Graphics Forum, vol. 28, pp. 1383–1392. Wiley Online Library (2009)
33. Scheuermann, G., Tricoche, X.: 17 - topological methods for flow visualization. In: Hansen, C.D., Johnson, C.R. (eds) Visualization Handbook, pp. 341–356. Butterworth-Heinemann (2005)
34. Theisel, H., Rössl, C., Weinkauf, T.: Topological representations of vector fields. In: Floriani, L., Spagnuolo, M. (eds) Shape Analysis and Structuring, Mathematics and Visualization, pp. 215–240. Springer, Berlin (2008)
35. Weinkauf, T., Theisel, H.: Curvature measures of 3d vector fields and their applications. J. WSCG **10**(2), 507–514 (2002)

Computing Accurate Morse-Smale Complexes from Gradient Vector Fields

Attila Gyulassy, Harsh Bhatia, Peer-Timo Bremer, and Valerio Pascucci

1 Introduction

As scientific data is becoming increasingly complex, domain experts are turning to more sophisticated techniques for its analysis. The Morse-Smale (MS) complex summarizes the gradient flow behavior of a scalar function and is recognized as a useful structure in such analyses. For example, it has been used to analyze Rayleigh-Taylor instabilities [15], to study the core structure of porous materials [9], to analyze burning regions in turbulent combustion simulations [1], to identify vortical structures [13], and to compute the filamentary structure of the cosmic web [27], to name just a few.

With the expanding use of the MS complex in analysis, improving the geometric accuracy of computed results is increasingly important. This paper extends previous work by Gyulassy et al. [11] in this domain. In the previous approach, the discrete gradient field was computed by identifying the probability of flowing to any given critical cell. The algorithm relies on processing cells in order of increasing function value to compute these probabilities. In contrast, the algorithm proposed in this paper works directly on a rotation-free vector field, with no knowledge of the underlying scalar values. This is achieved by first converting the vector field into a dependency graph, and reformulating the probability computation algorithm to work on a traversal of this graph. We show the effectiveness of this approach for several two-dimensional examples, first comparing the result against a similar construction

A. Gyulassy (✉) • H. Bhatia • V. Pascucci
School of Computing and SCI Institute, University of Utah, Salt Lake City, UT, USA
e-mail: jediati@sci.utah.edu; hbhatia@sci.utah.edu; pascucci@sci.utah.edu

P.-T. Bremer
Lawrence Livermore National Laboratory, Livermore, CA, USA
e-mail: bremer5@llnl.gov

on a scalar field, and then computing a segmentation of two rotation-free vector fields.

2 Related Work

Despite the fact that the initial ideas were already discussed more than a century and a half ago [2, 18] the first practical algorithm to compute MS complexes of two dimensional, piecewise linear functions was introduced fairly recently [7]. In Sect. 2.1 we discuss the most successful approaches for computing MS complexes based on discrete Morse theory. In Sect. 2.2 we review relevant topology-based analysis of vector fields.

2.1 Discrete Morse Theory

The main mathematical tool for computing MS complexes has been discrete Morse theory, as proposed by Forman [8], a formulation that has given rise to many efficient combinatorial algorithms. Lewiner [17] presented the first technique for constructing a discrete gradient field that agrees with the flow behavior of a scalar function. King et al. [14] introduced a technique to generate discrete Morse functions on simplicial complexes by modifying the Hasse diagram. Gyulassy et al. [10] introduced an algorithm that assigns cells in order of increasing function value and increasing dimension, using the ordering to avoid acyclicity checks. Reininghaus et al. [20, 22] presented an approach for generating discrete vector fields at multiple scales by computing matchings of a cell graph using the Hungarian method. Robins et al. [24] present an embarrassingly parallel technique that computes the discrete gradient on the lower star of a vertex. Shivashankar et al. [25, 26] also present an embarrassingly parallel technique for computing the gradient by carefully constructing a discrete Morse function. One commonality of all these techniques is that they pair cells into gradient arrows in the direction of steepest descent. This local optimization leads to significant error, and the geometric accuracy of the MS complex is further affected by the mesh orientation.

A randomized approach was independently introduced by Reininghaus et al. [23], for two-dimensional simplicial complexes, and by Gyulassy et al. [11], for volumetric meshes. This technique was shown to produce a better geometric reconstruction, and proven to converge to the smooth flow features under mesh refinement [11]. In a related field, Sousbie et al. [28] introduced a technique that integrated the probabilities of a randomized selection to produce watershed transforms with accurate geometry. The most geometrically accurate algorithm for computing MS complexes to date was presented by Gyulassy et al. [11], using a similar construction to Sousbie's, but in the context of discrete Morse theory.

2.2 Topological Analysis of 2D Vector Fields

An important research thrust of the past two decades is concerned with computing vector field topology. In their seminal paper Helman and Hesselink [12], define and compute the topological skeleton of a vector field. This structure segments the domain of a 2D field using streamlines, traced from each saddle of the field along its eigenvector directions. The nodes of the skeleton are critical points of the vector field and arcs are the separatrices connecting them. Since then, a number of techniques have been proposed to extract the topological skeleton in a stable and efficient manner.

For example, Chen et al. [4] create a triangle level graph by determining how the image of a triangle is advected and deformed by the PL flow. This triangle level graph can then be used to decompose the domain into topological structures. Szymczak and Zhang [29] focus on piecewise constant (PC) vector fields, and extract a similar graph, that subsequently can be used to extract topological structures. Recently, Levine et al. [16] computed the topological structure for PL flow by quantizing the flow along the edges of the triangulation, leading to a provably consistent streamline computation. This enables combinatorial and consistent extraction the topological structure of the field. Recent works of Reininghaus and Hotz [20, 22] construct a combinatorial vector field based on Forman's discrete Morse theory [8]. Using combinatorial fields allows the extraction of a consistent topological structure, however these techniques typically suffer from poor geometric accuracy due to the discretization of the flow.

3 Background

We present necessary background in discrete Morse theory and the algorithm to compute a geometrically accurate MS complex.

3.1 Morse Functions and the MS Complex

Let f be a real-valued smooth map $f : \mathbb{M} \to \mathbb{R}$ defined over a compact d-manifold $\mathbb{M}$. A point $p \in \mathbb{M}$ is critical when $|\nabla f(p)| = 0$, i.e. the gradient is zero, and is non-degenerate when its Hessian (matrix of second partial derivatives) is non-singular. The function f is a *Morse function* if all its critical points are non-degenerate and no two critical points have the same function value. In this case the *Morse Lemma* states that there exists local coordinates around p such that f has the following *standard form*: $f_p = \pm x_1^2 \pm x_2^2 \cdots \pm x_d^2$. The number of minus signs in this equation gives the *index* of critical point p. In three-dimensional functions, minima are index-0, 1-saddles are index-1, 2-saddles are index-2, and maxima are index-3.

An integral line in f is a path in $\mathbb{M}$ whose tangent vector agrees with the gradient of f at each point along the path. The integral line passing through a point p is the solution to $\frac{\partial}{\partial t}L(t) = \nabla f(L(t)), \forall t \in \mathbb{R}$, with initial value $L(0) = p$. Each integral line has an origin and destination at critical points of f. *Ascending* and *descending* manifolds are obtained as clusters of integral lines having common origin and destination respectively. The descending manifolds of f form a cell complex that partitions $\mathbb{M}$; this partition is called the *Morse* complex. Similarly, the ascending manifolds also partition $\mathbb{M}$ in a cell complex. A Morse function f is a *Morse-Smale function* if ascending and descending manifolds of its critical points only intersect transversally. An index-i critical point has an i-dimensional descending manifold and a $(d - i)$-dimensional ascending manifold. The cell complex formed by the intersection of the ascending and descending manifolds of a Morse-Smale function is the *Morse-Smale complex.*

3.2 Discrete Morse Theory

Discrete Morse theory is at the heart of current techniques for efficiently computing Morse-Smale complexes. We provide a brief overview with basic definitions from Forman [8], and we refer the reader to this introductory work for an intuitive description. A d-cell is a topological space that is homeomorphic to a Euclidean d-ball $B^d = \{x \in \mathbb{E}^d : |x| \leq 1\}$. For cells α and β, $\alpha < \beta$ means that α is a *face* of β and β is a *co-face* of α. If $dim(\alpha) = dim(\beta) - 1$, we say α is a *facet* of β, and β is a *co-facet* of α, and denote this $\alpha \dot{<} \beta$. When necessary to clarify the discussion, we may denote the dimension of a d-cell α with $\alpha^{(d)}$. The *star* of a cell α, denoted $St(\alpha)$, is the set of co-faces of α. The *link* of α is the closure of the star, minus the star itself, $Lk(\alpha) = \overline{St(\alpha)} - St(\alpha)$.

Let K be a regular complex that is a mesh representation of $\mathbb{M}$. The *barycenter* of a cell α is the image of a map $B : K \to \mathbb{M}$ from cells of K to their center of mass, points in $\mathbb{M}$. A function $F : K \to \mathbb{R}$ that assigns scalar values to every cell of K is a *discrete Morse function* if for every $\alpha^{(d)} \in K$, its number of co-facets $|\{\beta^{(d+1)} \dot{>} \alpha \ |F(\beta) \leq F(\alpha)\}| \leq 1$, and its number of facets $|\{\gamma^{(d-1)} \dot{<} \alpha \ |F(\gamma) \geq F(\alpha)\}| \leq 1$. A cell $\alpha^{(d)}$ is critical if its number of co-facets $|\{\beta^{(d+1)} \dot{>} \alpha \ |F(\beta) \leq F(\alpha)\}| = 0$ and its number of facets $|\{\gamma^{(d-1)} \dot{<} \alpha | F(\gamma) \geq F(\alpha)\}| = 0$, and has *index of criticality* equal d. The *lower star* of α, denoted $St^-(\alpha)$ is subset of $St(\alpha)$ where each element has lower function value. Similarly, the *lower link* of α is $Lk^-(\alpha) = \overline{St^-(\alpha)} - St^-(\alpha)$.

A *vector* in the discrete sense is a pairing of cells $\langle \alpha^{(d)}, \beta^{(d+1)} \rangle$, where $\alpha \dot{<} \beta$. We say that an arrow points from $\alpha^{(d)}$ to $\beta^{(d+1)}$. The *direction* of the arrow relates the combinatorial notion of the pairing to the geometric interpretation of the flow, and is given by $B(\beta^{(d+1)}) - B(\alpha^{(d)})$. Intuitively, this vector simulates a direction of flow.

A *discrete vector field* V on K is a collection of pairs $\langle \alpha_i^{(d)}, \beta_i^{(d+1)} \rangle$ of cells of K such that each cell is in at most one pair of V. A critical cell is unpaired. Given a discrete vector field V on K, a *V-path* is a sequence of cells

$$\alpha_0^{(d)}, \beta_0^{(d+1)}, \alpha_1^{(d)}, \beta_1^{(d+1)}, \alpha_2^{(d)}, \ldots, \beta_r^{(d+1)}, \alpha_{r+1}^{(d)}$$

such that for each $i = 0, \ldots$, r, the pair $\langle \alpha_i^{(d)}, \beta_i^{(d+1)} \rangle \in V$, and $\alpha_i^{(d)}$ and $\alpha_{i+1}^{(d)}$ are distinct facets of $\beta_i^{(d+1)}$. A V-path is the discrete equivalent of a streamline in a smooth vector field. A discrete vector field in which all V-paths are monotonic in F and do not contain any loops is a *discrete gradient field*, denoted G, of the discrete Morse function F. When constructing a discrete gradient field, we say that G is *valid* if these two conditions are met. One interpretation of discrete equivalent of *flow* in a continuous gradient is taking a *step* in a V-path, i.e., we say that $\alpha_{i+1}^{(d)}$ is one step from $\alpha_i^{(d)}$. The critical cell $\alpha_{r+1}^{(d)}$ at the end of a V-path is the *destination* of the V-path. We also say $\alpha_{r+1}^{(d)}$ *terminates* the V-path.

Ascending and descending manifolds of a critical cell α are obtained as the collection of V-paths that originate and terminate at α. The 1-skeleton of the discrete MS complex is composed of the critical cells and the V-paths that connect them.

3.3 Computing Morse-Smale Complexes with Accurate Geometry

Here we briefly review the algorithm presented by Gyulassy et al. [11] for computing discrete gradient fields with accurate geometry, and refer the reader to the original work for a more complete description. The algorithm is based on the observation that if gradient arrows are paired randomly in a downwards direction with appropriate probabilities, the discrete gradient field produced will have convergent geometry. The probability for selecting a potential gradient arrow is given by the alignment of the arrow with the gradient, normalized over all possible arrows. Specifically, the alignment is given by the dot product of the (continuous) gradient evaluated at the barycenter of the cell, and the "displacement" implied by the gradient arrow in the discrete flow, i.e., the difference in the coordinate locations of the cells at the head and tail of a gradient arrow. The function f is a continuous function, typically obtained by interpolating samples on the vertices of the mesh K.

$$w(\langle \alpha, \beta_i \rangle) = max\{(B(\alpha) - B(\beta_i)) \cdot (-\nabla f(B(\alpha)),\ 0\} \tag{1}$$

Using these alignment weights, the probability of picking $\langle \alpha, \beta_i \rangle$ is defined as

$$Pr(\langle \alpha, \beta_i \rangle) = w(\langle \alpha, \beta_i \rangle) / \sum_{\beta_j \in St^-(\alpha)} w(\langle \alpha, \beta_j \rangle). \tag{2}$$

Although V-paths in a purely randomized discrete gradient field constructed using these probabilities converges to integral paths in f under mesh refinement, the rate of convergence is slow. This motivated a second algorithm, that computed the exact probability that a cell would belong to the ascending manifold of any critical cell. The probability that α^d flows to any critical cell κ^d is given by a map $\mu : K \to \mathbb{R}$, called a *membership distribution*. In the following, let β be a co-facet of α^d, and $Lk^-_\beta(\alpha)$ denote the set of d-cells in the lower link of α that are also facets of β.

$$\mu_\alpha(\kappa) = \begin{cases} 0 & \text{if } dim(\alpha) \neq dim(\kappa) \\ 1 & \text{if } \alpha \text{ is critical and } \kappa = \alpha \\ 0 & \text{if } \alpha \text{ is critical and } \kappa \neq \alpha \\ \sum\limits_{\beta \in St^-(\alpha)} Pr(\langle\alpha,\beta\rangle) \sum\limits_{\rho \in Lk^-_\beta(\alpha)} \dfrac{\mu_\rho(\kappa)}{|Lk^-_\beta(\alpha)|} & \text{otherwise.} \end{cases} \tag{3}$$

This recurrence relation is solved by processing d-cells in order of increasing function value, with ties broken by simulation of simplicity [5]. Through this ordering, it is guaranteed that the membership distributions of every d-cell in the lower link of α has already been computed. Gradient arrows between d-cells and $(d+1)$-cells are assigned to minimize the likelihood that a new gradient arrow crosses the boundary between any two ascending manifolds.

When pairing α, each cell in the lower link $\gamma_i \in Lk^-(\alpha)$ has already been paired. In fact, γ_i is part of V-paths that do not change below γ_i with any subsequent gradient arrow assignments, and we can find their terminating critical cells. It is well-known that in dimensions higher than one, V-paths can split, therefore, let D_{γ_i} be the set of critical cells that terminate V-paths flowing through γ_i. The weight of $\langle\alpha, \beta_i\rangle$ is defined as:

$$w(\langle\alpha,\beta_i\rangle) = \sum_{\gamma \in Lk^-(\alpha), \gamma \dot{<} \beta_i} \left(\sum_{\kappa \in D_\gamma} \mu_\alpha(\kappa) \right) \tag{4}$$

Intuitively, this weight represents the likelihood that α belongs to the same ascending manifolds as the facets of β. Therefore, higher weights indicate potential pairs that are less likely to cross boundaries of ascending manifolds.

4 Approach

The algorithm described in Sect. 3.3 relies on sorting the d-cells of a mesh according to function values. However, these may not be available when given only a continuous gradient field. Although it is possible to create a valid scalar field from any gradient field, we instead present an algorithm that bypasses this step and works directly with the vector values. Our algorithm first computes a directed graph that encodes the dependencies in Eq. (3). For a scalar field this graph should be acyclic, however due to numerical instability as well as sampling artifacts, cycles occur in practice, and we present a technique to break these cycles. Next, the algorithm constructs the membership distributions and gradient arrows by processing elements in order of traversal of the dependency graph.

4.1 Dependency Graph from Vector Field

Let V be a vector-valued function defined on a manifold $\mathbb{M}$, and let K be a mesh representing $\mathbb{M}$. We construct a *dependency graph* that encodes the ordering relationship needed to evaluate membership distributions. We start with the Hasse Diagram H of K, the directed graph where each cell of K is represented as a node with arrows pointing to each of its facets. Next, we modify H to H' by reversing all arrows that are aligned (positive dot product) with the vectors of V at the tail of the arrow. Now H' represents flow in a "downwards" direction, where the processing of a cell at the tail of an arrow depends on the prior processing of the cell at the head of an arrow. This procedure is illustrated in Fig. 1.

The algorithm for assigning gradient arrows assigns all d-cells before all $(d+1)$-cells, therefore we must further modify H' to ensure no deadlocks occur. This is partly ensured by reversing the lowest weight arrow for any cell having only incoming arrows from its facets, where the weight is defined as magnitude of alignment with the vector value. These local changes modifying H' to H'' now ensure that all the faces of a cell will be assigned prior to the cell itself.

Finally, in order to represent the dependency graph of a scalar function, H'' must be acyclic. In practice, H'' can admit cycles where numerical error or sampling artifacts simulate rotation in V. These cycles can be found efficiently by first flood-filling H'' from sinks in the graph, following arrows in a head-to-tail manner,

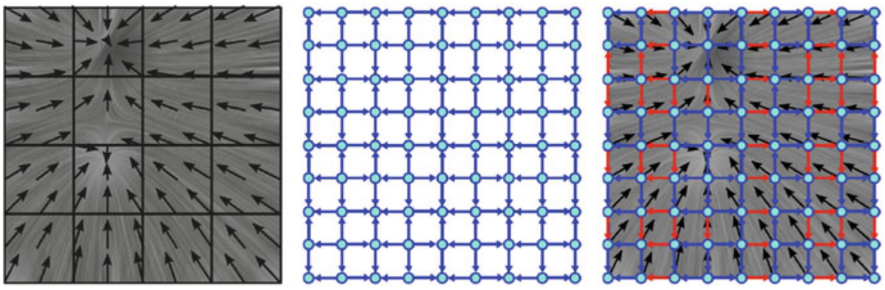


Fig. 1 The vector field is sampled at the barycenter of each cell (*left*). The Hasse Diagram H of the grid encodes facet relationships (*center*). H is modified to H' by reversing any *arrow* that is aligned with the underlying vector field (*right*)

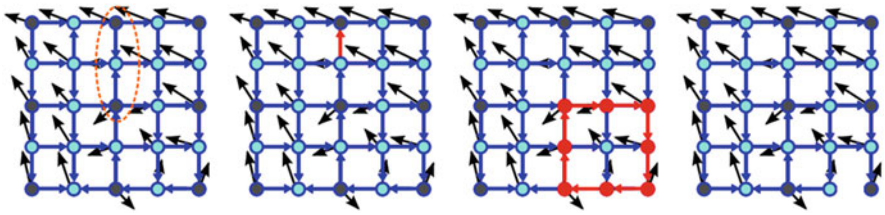


Fig. 2 An edge in the modified Hasse Diagram H' has two vertices depending on it (*far left*) due to numerical instability. This dependency is broken by reversing one arrow (*middle left*). A cycle is detected in H'' (*middle right*), and broken by removing the lowest weight arrow (*far right*)

and marking cells as visited. Any cell not marked in this flood fill depends at some point on a cell that is part of a cycle. This cycle can be found by depth-first search from an unmarked cell. A cycle is broken by deleting its arrow in H'' that has the lowest weight, where the weight is the magnitude of the dot product of the arrow with the vector value. Breaking all cycles results in modifying H'' to the final dependency graph G. Converting H' to H'' to G is illustrated in Fig. 2.

If the scalar values are known, the construction of the dependency graph is simplified. We construct the modified Hasse diagram H' as before by reversing arrows aligned with the gradient. However, cycles are removed by deleting any arrows where the scalar value at the tail is lower than the scalar value at the head. Simulation of simplicity ensures consistency in the comparison and eliminates flat regions. This embarrassingly parallel technique produces a cycle-free dependency graph.

4.2 Accurate Algorithm

In this section we present a modified version of the accurate geometry algorithm [11] that uses vector values and our dependency graph G. As in the previous approach, discrete gradient arrows are assigned in rounds of increasing dimension, first by assigning arrows that maximize the weights based on membership distributions, and then by simple homotopy expansions. However, in each case, the sorted ordering is replaced by a traversal of the dependency graph G, and operations such as lower star St^{-} and lower link Lk^{-} are replaced by a lookup of dependencies in G. In particular, define $D(\alpha)$ to be the set of cells reachable by one tail-to-head arrow in G, replacing $St^{-}(\alpha)$. In the following, let K be the mesh, V be the vector-valued function, and Γ be the discrete gradient field.

```
ComputeGradient(K, V) :
$\Gamma$ = {}
for $d \in [0, D]$ do
    $\Gamma$ = AssignArrows(d, K, V, $\Gamma$)
    $\Gamma$ = HomotopyExpand(K, V, $\Gamma$)
end for
return $\Gamma$
```

The algorithm to assign gradient arrows for d-cells, *AssignArrows*, uses a queue Q to traverse the dependency graph. In the following *SeedQueue*(d, G, Q) traverses G and inserts into Q all d-cells having no dependencies, i.e. no outward pointing arrows in G. The *UpdateQueue*(d, G, Q) inserts d- and $(d+1)$-cells into Q whose dependencies have been resolved, those cells for which every neighbor at the head of a downwards pointing arrow in G has been processed by the algorithm.

```
AssignArrows(d, K, V, $\Gamma$) :
$Q = \emptyset$
SeedQueue($d, G, Q$)
while not $Q.empty()$ do
    $\alpha = Q.top(); Q.pop()$
    if dim($\alpha, K) = d$ then
        C = $\{\beta \in D(\alpha)\}$
        if $C = \emptyset$ then
            $\Gamma = \Gamma \cup \alpha \leftarrow critical$
        else
            $\beta_j = argmax_{\beta_i \in C}(w(\langle\alpha, \beta_i\rangle))$
            $\Gamma = \Gamma \cup \langle\alpha, \beta_j\rangle$
        end if
    end if
    $process(\alpha)$
    $UpdateQueue(d, G, Q)$
end while
return $\Gamma$
```

```
HomotopyExpand(d, K, V, $\Gamma$) :
$H = \{\alpha \in K |$ # unassigned facets in $\Gamma$ is 1$\}$
while $H \neq \emptyset$ do
    $\alpha$ = PopFirst(H)
    $\beta$ = unassigned facet of $\alpha$
    if $\alpha \in D(\beta)$ then
        $\Gamma = \Gamma \cup \langle\alpha, \beta\rangle$
        update($H$)
    end if
end while
return $\Gamma$
```

Let $K^n \subseteq K$ be the subcomplex of assigned cells of K after selecting n pairs. Assigning a gradient arrow adds exactly two cells to this subcomplex and marking a cell as critical adds one. Each subcomplex K^n either adds a new critical d-cell, or attaches a $(d+1)$-cell to d-cell pair to K^{n-1}. An assignment of a gradient arrow is a *simple homotopy expansion* if K^n is homotopic to K^{n+1}. In practice, a d-cell to $(d+1)$-cell arrow can be inserted without changing the homotopy type of the subcomplex when (1) all faces of the d cell are assigned, and (2) the d-cell is the only unassigned face of the $(d+1)$-cell. The only time we prohibit this expansion is when the arrow would point "uphill", i.e., the $(d+1)$-cell depends on the d-cell in the dependency graph G.

The above algorithm produces accurate geometry for the boundaries of ascending manifolds, and to obtain accurate geometry for descending manifolds as well we perform a second pass. The second pass operates on the dual of the grid, with vector values negated, and the added constraint that each cell can only be paired with other cells of the same dimensional ascending manifold, as identified in the first phase.

5 Results

We present results for two-dimensional examples, both for gradient fields derived from scalar functions, and also for rotation-free vector fields. We also discuss limitations of the current approach.

5.1 Comparison of Constructions

In many instances the gradient of a scalar function is known, for instance in scientific simulations. Often, this gradient is of much higher quality than the local approximation obtained by interpolating and comparing scalar values. We compare the result of working directly with the gradient field to the result of the scalar approximation in Fig. 3. Here, the first technique uses difference in scalar values to determine the steepness of an edge, while the second technique uses a central difference approximation of the gradient. While working directly with the gradient produces slightly smoother results, the central difference creates many additional critical points along flat ridge-like structures.

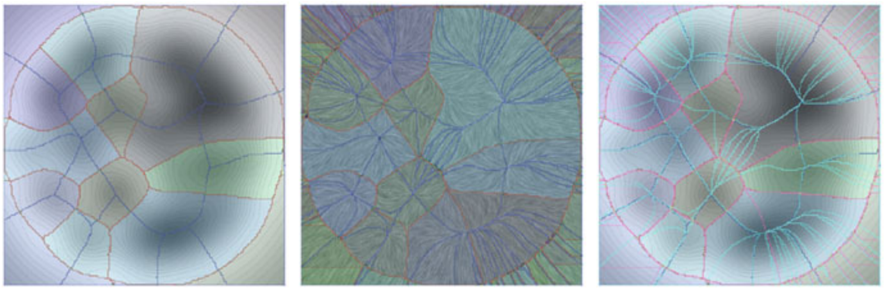


Fig. 3 The scalar-value based algorithm [11] is applied to a scalar field (*left*). A central difference formula is used to find the gradient of this function, and the complex is computed with our vector-value based algorithm (*middle*). The two results are overlaid for comparison (*right*)

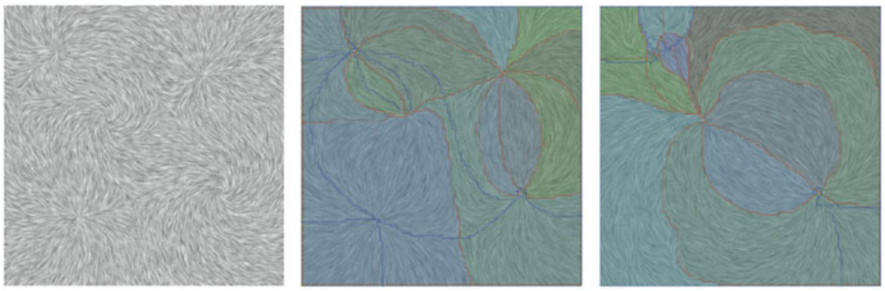


Fig. 4 A two-dimensional vector field (*left*) is decomposed into a rotation free (*middle*), and $\pi/2$-rotated divergence-free (*right*) components. These derived fields are segmented with our new approach

5.2 Rotation-Free Vector Fields

Another source of gradient-like vector fields comes from the Helmholtz-Hodge decomposition (HHD). We restrict the discussion here to two-dimensional simply connected domains. The HHD [3] decomposes a smooth vector field (V) into its curl-free ($\mathbf{r}$) and divergence-free ($\mathbf{d}$) component fields. It is an important result, and assists in the analysis of complex flows in terms of their simpler components [19, 30, 31]. In two-dimensions, the components $\mathbf{d}$ and $\mathbf{r}$ can be represented as the gradient and co-gradient of two scalar potentials respectively. If J is defined as the $\pi/2$-rotation operator, then both $\mathbf{d}$ and $J\mathbf{r}$ are rotation-free fields. Reininghaus and Hotz [21] showed that the topology of a 2D divergence-free vector field could be recovered by the scalar topology of a $\pi/2$-rotation, using the HHD to remove the divergent vector field component. For the purpose of our examples, we compute the HHD of synthetic vector fields defined on a regular grid to obtain the rotation-free fields. We illustrate the results of this segmentation in Fig. 4.

5.3 Known Limitations

The proposed technique is based on the ability to detect to which ascending manifold a cell is most likely to belong, and produces a gradient field that avoids crossing boundaries between them. This produces artifacts in practice in two instances: first, in the case of a *strangulation*, i.e. where both sides of a separating line belong to the same ascending manifold, such as in the interior of a volcano-shaped region; and second, where new critical points are added due to geometric constraints in the second pass of the algorithm. Since these critical points are never detected in the first pass, accurate boundaries are never computed for their ascending manifolds. In the scalar field construction, these spurious critical points have low persistence and are immediately removed during simplification [6]. However, the

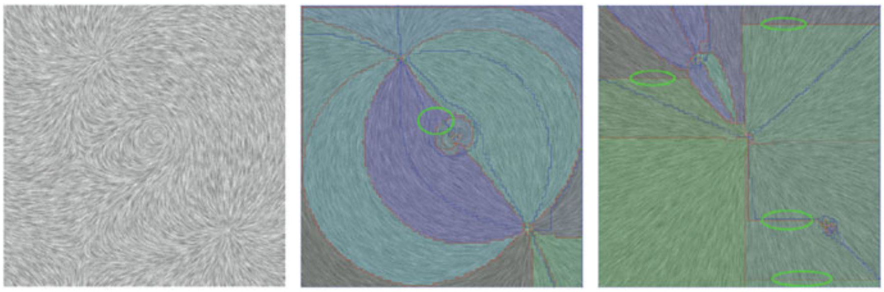


Fig. 5 The limitations for the approach are highlighted for a two-dimensional vector field. When both "sides" of a separatrix belong to the same ascending or descending manifold, the geometric accuracy of the separatrix is poor (*middle*). When critical points are added by the second pass of the algorithm, the boundaries of their ascending manifolds have poor geometric quality (*right*)

vector-valued construction does not assign scalar values to the nodes of the resulting Morse-Smale complex, and therefore an additional procedure, such as integrating alignment with the vector field along the path, must be applied to remove them, as proposed by Reininghaus et al. [20]. Figure 5 illustrates these artifacts.

Conclusions and Future Work

We have modified the algorithm presented by Gyulassy et al. [11] to operate on rotation-free vector fields. Our approach generated a dependency graph, resolving numerical problems such as cycles, and converted the sorted order of the scalar-valued algorithm into a more general dependency-graph traversal. We presented the results of our algorithm for a gradient field derived from a scalar field, as well as for rotation-free vector fields derived from Helmholtz-Hodge decomposition.

One of the main advantages of this approach is the more relaxed ordering of computation provided by the dependency graph. Indeed, many portions of the discrete gradient field can be computed independently from one another, which provides a potential avenue for parallelizing this construction. In particular, we will use this dependency graph constructed for scalar functions as a basis for a block-parallel approach to handle large-scale data.

Acknowledgements This work is supported in part by NSF OCI-0906379, NSF OCI-0904631, DOE/NEUP 120341, DOE/MAPD DESC000192, DOE/LLNL B597476, DOE/Codesign P01180734, and DOE/SciDAC DESC0007446.

References

1. Bremer, P.-T., Weber, G., Pascucci, V., Day, M., Bell, J.: Analyzing and tracking burning structures in lean premixed hydrogen flames. IEEE Trans. Vis. Comput. Graph. **16**(2), 248–260 (2010)
2. Cayley, A.: On contour and slope lines. Lond. Edinb. Dublin Philos. Mag. J. Sci. **XVIII**, 264–268 (1859)
3. Chorin, A.J., Marsden, J.E.: A Mathematical Introduction to Fluid Mechanics. Springer, Berlin (1993)
4. Chen, G., Mischaikow, K., Laramee, R.S., Zhang, E.: Efficient Morse decompositions of vector fields. IEEE Trans. Vis. Comput. Grap. **14**(4), 848–862 (2008)
5. Edelsbrunner, H., Mücke, E.P.: Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. ACM Trans. Graph. **9**, 66–104 (1990)
6. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, FOCS '00, pp. 454–463. IEEE Computer Society, Washington (2000)
7. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. Discrete Comput. Geom. **30**, 87–107 (2003)
8. Forman, R.: A user's guide to discrete Morse theory. In: Séminare Lotharinen de Combinatore, vol. 48 (2002)
9. Gyulassy, A., Duchaineau, M., Natarajan, V., Pascucci, V., Bringa, E., Higginbotham, A., Hamann, B.: Topologically clean distance fields. IEEE Trans. Comput. Graph. Vis. **13**(6), 1432–1439 (2007)
10. Gyulassy, A., Bremer, P.-T., Pascucci, V., Hamann, B.: A practical approach to Morse-Smale complex computation: scalability and generality. IEEE Trans. Vis. Comput. Graph. **14**(6), 1619–1626 (2008)
11. Gyulassy, A., Bremer, P.T., Pascucci, V.: Computing Morse-Smale complexes with accurate geometry. IEEE Trans. Vis. Comput. Graph. **18**(12), 2014–2022 (2012)
12. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. IEEE Comput. **22**(8), 27–36 (1989)
13. Kasten, J., Reininghaus, J., Hotz, I., Hege, H.-C.: Two-dimensional time-dependent vortex regions based on the acceleration magnitude. IEEE Trans. Vis. Comput. Graph. **17**(12), 2080–2087 (2011)
14. King, H., Knudson, K., Neza, M.: Generating discrete Morse functions from point data. Exp. Math. **14**(4), 435–444 (2005)
15. Laney, D., Bremer, P.-T., Mascarenhas, A., Miller, P., Pascucci, V.: Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. IEEE Trans. Vis. Comput. Graph. **12**(5), 1052–1060 (2006)
16. Levine, J.A., Jadhav, S., Bhatia, H., Pascucci, V., Bremer, P.-T.: A quantized boundary representation of 2D flow. Comput. Graph. Forum (EuroVis Proc.) **31**(3pt1), 945–954 (2012)
17. Lewiner, T.: Constructing discrete Morse functions. Master's thesis, Department of Mathematics, PUC-Rio (2002)
18. Maxwell, J.C.: On hills and dales. Lond. Edinb. Dublin Philos. Mag. J. Sci. **XL**, 421–427 (1870)
19. Polthier, K., Preuß, E.: Identifying vector fields singularities using a discrete Hodge decomposition. In: Hege, H.C., Polthier, K. (eds.) Mathematical Visualization III, pp. 112–134. Springer Verlag (2003)
20. Reininghaus, J., Hotz, I.: Combinatorial 2d vector field topology extraction and simplification. In: Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications. Mathematics and Visualization, pp. 103–114. Springer, Berlin (2011)

21. Reininghaus, J., Hotz, I.: Computational discrete Morse theory for divergence-free 2d vector fields. In: Peikert, R., Hauser, H., Carr, H., Fuchs, R. (eds.) Topological Methods in Data Analysis and Visualization II. Mathematics and Visualization, pp. 3–14. Springer, Berlin/Heidelberg (2012)
22. Reininghaus, J., Lowen, C., Hotz, I.: Fast combinatorial vector field topology. IEEE Trans. Vis. Comput. Graph. **17**, 1433–1443 (2011)
23. Reininghaus, J., Gunther, D., Hotz, I., Weinkauf, T., Seidel, H.-P.: Combinatorial gradient fields for 2d images with empirically convergent separatrices, arxiv, 1(1208.6523) (2012)
24. Robins, V., Wood, P., Sheppard, A.: Theory and algorithms for constructing discrete Morse complexes from grayscale digital images. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1646–1658 (2011)
25. Shivashankar, N., Natarajan, V.: Parallel computation of 3d Morse-Smale complexes. Comput. Graph. Forum **31**(3pt1), 965–974 (2012)
26. Shivashankar, N., Natarajan, V.: Parallel computation of 3D Morse-Smale complexes. Comp. Graph. Forum **31**(3pt1), 965–974 (2012). doi:10.1111/j.1467-8659.2012.03089.x
27. Sousbie, T.: The persistent cosmic web and its filamentary structure - i. Theory and implementation. Mon. Not. R. Astron. Soc. **414**(1), 350–383 (2011)
28. Sousbie, T., Colombi, S., Pichon, C.: The fully connected n-dimensional skeleton: probing the evolution of the cosmic web. Mon. Not. R. Astron. Soc. **393**(2), 457–477 (2009)
29. Szymczak, A., Zhang, E.: Robust Morse decompositions of piecewise constant vector fields. IEEE Trans. Vis. Comput. Grap. **18**(6), 938–951 (2012)
30. Tong, Y., Lombeyda, S., Hirani, A.N., Desbrun, M.: Discrete multiscale vector field decomposition. ACM Trans. Graph. **22**(3), 445–452 (2003)
31. Wiebel, A.: Feature detection in vector fields using the Helmholtz-Hodge decomposition. Diplomarbeit, University of Kaiserslautern (2004)

Part V
High-Dimensional Data Analysis: Exploration of High-Dimensional Models

Exercises in High-Dimensional Sampling: Maximal Poisson-Disk Sampling and k-d Darts

Mohamed S. Ebeida, Scott A. Mitchell, Anjul Patney, Andrew A. Davidson, Stanley Tzeng, Muhammad A. Awad, Ahmed H. Mahmoud, and John D. Owens

1 Introduction

1.1 Maximal Poisson-Disk Sampling (MPS) Definition

Sample points are called *well-spaced* if they have a limited ratio between the maximum distance between any domain point and its nearest sample point, and the minimum distance between two sample points. A well-spaced sampling is efficient at exploring a space. The maximum distance ensures that the domain is adequately covered and reduces interpolation error. The minimum distance ensures that we do not waste time or generate noise with samples that provide similar information to nearby samples. Typically well-spaced is defined locally, by stating that the aspect ratio of Voronoi cells is bounded. Samples that have random positions are often preferred because they do not introduce directional bias in the estimates.

M.S. Ebeida • S.A. Mitchell (✉)
Computing Research, Sandia National Laboratories, Albuquerque, NM, USA
e-mail: msebeid@sandia.gov; samitch@sandia.gov

A. Patney • A.A. Davidson • J.D. Owens
Department of Electrical and Computer Engineering, University of California, Davis, CA, USA
e-mail: anjul.patney@gmail.com; aaldavidson.ucdavis@gmail.com; jowens@ece.ucdavis.edu

S. Tzeng
Department of Computer Science, University of California, Davis, CA, USA
e-mail: solgoodguy@gmail.com

M.A. Awad • A.H. Mahmoud
Department of Naval Architecture and Marine Engineering, Alexandria University, Egypt
e-mail: muhammad.a.awad91@student.alx.edu.eg; eng.a.hassen@gmail.com

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_13

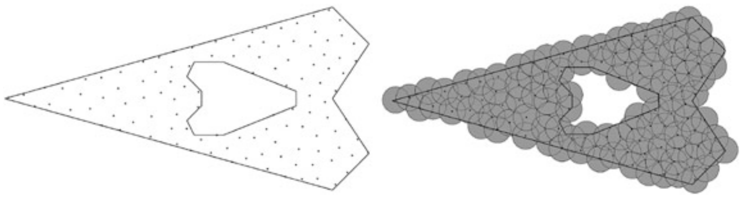


Fig. 1 A maximal Poisson disk sample over a non-convex domain and a uniform sizing function

Maximal Poisson-disk Sampling (MPS) is a process that selects a random set of points, $X = \{x_i\}$, from a given domain, $\mathscr{D}$, in some d-dimensional space. The samples are at least a minimum distance apart, satisfying an empty disk criterion: Eq. (2). For simplicity we focus on the uniform case, where the disk radius, r, is constant regardless of location or iteration. Inserting a new point, x_i, defines a smaller domain, $\mathscr{D}_i \subset \mathscr{D}$, available for future insertions, where $\mathscr{D}_o = \mathscr{D}$; see Eq. (1). The *maximal condition*, Eq. (3), requires that the sample disks overlap, in the sense that they cover the whole domain leaving no room to insert an additional point. This property identifies the termination criterion of the associated sampling process. *Bias-free* or *unbiased* means that the likelihood of the next sample being inside any remaining subdomain is proportional to the area of the subdomain; see Eq. (1). This is uniform sampling from the uncovered area, equivalent to uniform sampling over the entire domain, and rejecting already-covered points. See Fig. 1 for an example MPS over a non-convex domain. Extending MPS disks to squares, or ellipses and rectangles in anisotropic spaces, is natural, yet unexplored.

$$\text{Bias-free: } \forall x_i \in X, \forall \Omega \subset \mathscr{D}_{i-1} : P\,(x_i \in \Omega) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathscr{D}_{i-1})} \tag{1}$$

$$\text{Empty disk: } \forall x_i, x_j \in X, x_i \neq x_j : ||x_i - x_j|| \geq r \tag{2}$$

$$\text{Maximal: } \forall x \in \mathscr{D}, \exists x_i \in X : ||x - x_i|| < r \tag{3}$$

1.2 Applications of MPS

Random point distributions, including MPS, have found widespread use in computer graphics [17, 18]. Most applications are in dimensions below 6. Rendering Sect. 5.2 makes use of sampling light rays. The global illumination problem is concerned with computing indirect lighting, light from sources that are reflected off surfaces before illuminating an object in a scene. Computing the light in a scene exactly is intractable because of the number of combinations of light sources, ray directions,

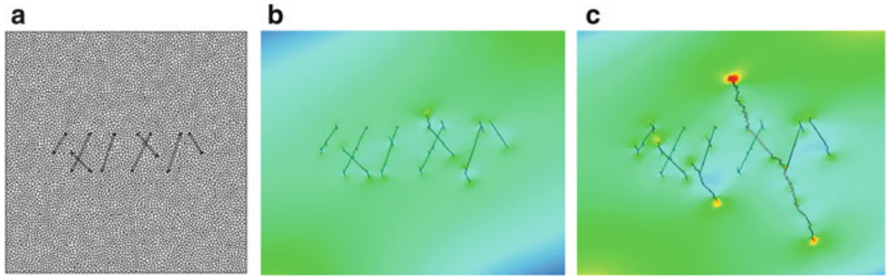


Fig. 2 Fracture after injecting CO_2 below caprock. The *color* represents maximum principal stress. The initial fracture joints are sealed, but opened and spread due to the high injection pressure. (**a**) Initial Voronoi mesh. (**b**) Early time. (**c**) Late time

surface reflection angles, and observer positions. This is made worse if these quantities are unknown a priori, such as the location of an observing character in a game. An approximate solution makes use of MPS points. These points sample the scene space, and we can precompute the approximate contribution of each sample point, the inter-sample transmissions. Then, in real time, we may magnify by light source intensity and interpolate and combine with other information. A similar MPS sampling and workflow is done for texture synthesis. A small piece of a texture is placed at each MPS sample point, and the boundaries between patches are blended so as not to stand out to an observer.

In these graphics applications and some others, all three properties of MPS are desired: maximality ensures the accuracy of the sampled approximate solution; empty disk ensures efficiency by avoiding nearby, redundant points; and bias-free avoids artificial visual artifacts, repeating patterns, that a deterministic regular spacing produces. Humans are expert at detecting patterns, even imagining them where none exists. The distribution of retinal cells in our eyes has a Fourier spectrum much like that of MPS, which may help explain why MPS works so well.

MPS points are well-spaced, which leads to meshes with well-shaped elements. Other meshing algorithms generate well-spaced points, but MPS is also random. For simulating fracture, e.g. for carbon sequestration in Fig. 2, meshes with both properties are required [2, 7]. The mesh randomness models some of the natural material strength variability. Generating different meshes for the same geometry using the same sizing function provides a useful tool to study the sensitivity of the solution to the mesh, complementary to refinement studies; see Fig. 6a.

1.3 Potential Ties to Computational Topology

We speculate that some MPS techniques and applications may be useful for analysis of data for computational topology. For example, as in global illumination, MPS could be used as cluster centers as a form of resampling the domain data, to reduce the size of a computational topology point cloud. MPS processes could also be used to generate the initial data points, perhaps modified to place more points where they are topologically significant. In particular, many computational topology algorithms rely on a mesh. Our sampling techniques can be used to create such a mesh. Going further, we may circumvent the need for a mesh by using the structure implied by the disks. For example, we are investigating implicit Voronoi diagrams for traversing a point cloud to construct the Morse-Smale complex.

One recent use of MPS points in high dimensions is real-time robot motion planning, where the space is the configuration space of the robot and its obstacles [19]. The challenges for motion planning are similar to those for some discrete computational topology calculations. Both rely on an imperfect representation of the space by point clouds. Algorithms for both problems typically suffer from the curse of dimensionality, and the point clouds are typically high dimensional. In motion planning the goal is to find one path between two points, which has similarities to finding Morse-Smale or Reeb graph paths, and contrasts to trying to characterize the entire space in homology calculations. Recently a realtime robot motion planning problem was solved using our MPS sampling of its 23-dimensional configuration space, so there is hope that large spaces from topology might become computationally tractable as well. Conversely, it is possible that computational topology techniques for finding smooth and short paths, e.g. homology generators and Morse-Smale paths, could be used to smooth and shorten a robot path. Smooth and short paths mimic human motions and are more efficient.

2 MPS Algorithms in Low Dimensions

2.1 Algorithmic Challenges

Poisson-disk sampling is defined as a serial statistical process of rejection sampling: generate a disk uniformly at random and reject it if its center lies inside a prior disk. A maximal sampling is defined as achieving the limit distribution. A straightforward implementation of the statistical process is called "dart throwing," where a dart is synonymous with a candidate disk center point. In dart throwing, most darts are accepted in the beginning of the process, but the likelihood of accepting the next point is proportional to the volume fraction of the domain left uncovered by a disk, which typically becomes vanishingly small as the process continues.

A process definition is not the same as an output definition, nor necessarily the most efficient way of achieving that output. (Consider defining "sorted order" as the output of the bubble sort process. Only after "sorted order" is defined independent of the process that produced it do we have the chance to discover quicksort.) Unfortunately we (the community) do not have a precise closed-form description of the MPS output distribution, nor are we sure that it is the ideal for random well-spaced points [14]. One branch of research has modified the MPS process in order to achieve efficiency [5]. Our (the authors') first solutions are in a different research branch, where we propose algorithms that produce equivalent outputs to dart throwing, but more efficiently.

2.2 *Our Algorithmic Solutions*

In particular, to find an efficient equivalent process, we rely on the observation that the probability of introducing the next disk center in any uncovered subregion is proportional to the area of the subregion. We use a background grid subdivision of the domain to track a superset of the remaining uncovered subregions. Cells that are completely covered by a single disk are discarded. Efficiency follows from the superset being not much bigger than the uncovered region. The background grid is uniform with cell diagonals of length r. This size ensures a cell can have at most one point, whose disk completely covers the cell. The background grid also speeds up retrieving nearby disks, to check if a sample point is inside one. We now summarize two variations that use this grid [11, 12].

2.2.1 Efficient MPS by Polygon Tracking

The first MPS algorithm we developed has two phases [12]. The first phase is dart throwing, but each dart is selected within a grid cell, not the entire domain. After a number of dart throws proportional to the number cells, we switch to the second phase. We further refine each cell by constructing a polygon that is closer to the uncovered region inside the cell, but is still a superset; see Fig. 3. Now we do dart throwing inside the polygons: we select a polygon uniformly by area, then a dart uniformly inside it. If the dart is uncovered we accept it and update the polygons. The chance of a dart being uncovered is provably large, which leads to a provable expected run-time of $\mathbb{E}(n \log n)$. (There is typically no deterministic time bound, e.g. $O(n \log n)$, for these types of algorithms because of the random decisions they

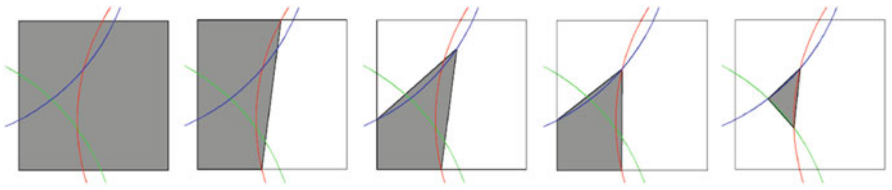


Fig. 3 Generating a tight polygonal superset of the uncovered regions. From *left* to *right*, we start with the cell, then subtract disks. We use the chords instead of the arcs between intersection vertices

must make.) Unbeknownst to us there was a prior method with the same guarantees of maximality, bias-free, and $\mathbb{E}(n \log n)$ time [15]. It uses the evolving Delaunay triangulation to keep track of the remaining void. Ours [12] was the first method based on grids with these guarantees; grids are preferred in some settings due to their locality, simplicity, and speed. Our method was easily parallelized on a GPU. However, keeping track of polygon intersections is both cumbersome and requires a large amount of memory. If we increase the domain dimension the memory consumption explodes, restricting this method to low-dimensional spaces. The next method we developed addresses these shortcomings for slightly higher dimensions, and we prefer it even for lower dimensions.

2.2.2 Simple MPS by Implicit Quad-Trees

The second method [11] we developed, Simple MPS, maintains all of the desirable qualities of our previous method, Efficient MPS [12], namely maximality and bias-free, with the added benefits of being simpler to code; using less memory; faster run-time even in low dimensions and scalable to higher dimensions in practice (but without a run-time proof). It starts with dart throwing in the background grid; described in Sect. 2.2.1 and identical to the first algorithm [12]. However, instead of proceeding to polygons, we simply subdivide all the remaining child cells. Covered child cells are discarded, and we repeat the algorithm on the remaining cells. Since the cells are all the same size, it is easy to represent them by indices, and we do not need the overhead of a tree as in a true quadtree.

The key to this strategy's efficiency is that the *collection* of active cells is a close approximation to the *entire* uncovered area, even if one particular cell is much larger than the uncovered area it encloses. In practice the number of cells decreases geometrically per refinement, which helps both runtime and memory, and allows us to reach maximality by refining down to machine precision. We were able to maximally sample 6d domains on a CPU. On the GPU, we sampled at an impressive rate of 1 M samples/s in 2d and 75 K samples/s in 3d. More details on how to parallelize, and proofs of maximality and bias-free, can be found in our paper [11].

2.3 Variable Radii MPS

We define two useful variations of MPS [16]. In the first version, the size of the disks varies spatially over the domain; see Fig. 5b for a classic stippling application. We still get locally well-spaced points if this variation is slow, e.g. the disk sizing function satisfies a Lipschitz condition. The quality of the meshes generated from

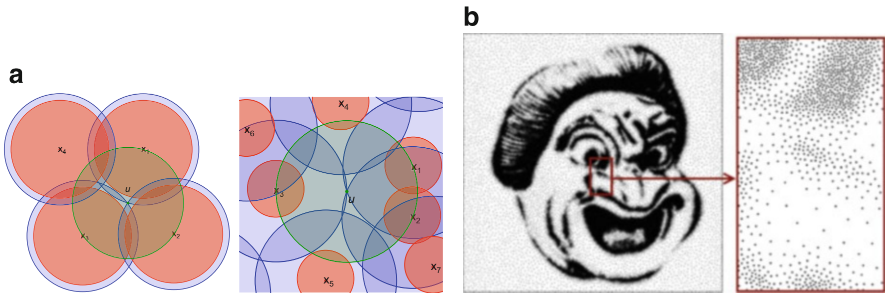


Fig. 4 Variable radii MPS. (**a**) Two-radii, $r_c > r_f$. (**b**) Spatially varying radii

those points degrades smoothly as the variation increases, up to some critical threshold after which there are no guarantees. The run-time also increases with the variation, because proximity checks must search a larger neighborhood. Spatially varying radii had been considered previously by other authors [3], but we appear to be the first to quantify these conditions and their effects [16].

If the radii of two disks differ, there are several ways of defining "conflict," the conditions under which a dart is rejected because it fails to satisfy a version of Eq. (2). There are variations based on size and generation order. Each variation has advantages and disadvantages. Defining conflict as the smaller disk containing the center of the larger disk provides the best quality and it can tolerate the largest Lipschitz constant (< 1), but generates the largest point sets. If we define a conflict as a candidate disk center lying inside a previously accepted disk, then this is the easiest to implement, as it is a minor change to Simple MPS and other algorithms. However, it has the biggest restriction on the Lipschitz constant ($< 1/2$) and provides the weakest output quality guarantees. We have also explored *sifted disks* for reducing the discrete density of a maximal point set, by removing and relocating points, while still maintaining the MPS conditions [9].

Our second useful variation of MPS [16] is to use two radii for each point, two-radii MPS; see the concentric blue and red disks in Fig. 4. We decouple the maximality (coverage, blue) and conflict (empty-disk, red) conditions, by using different disk radii for each. I.e. we replace the disk-free "r" in Eq. (2) with r_f, and replace the coverage "r" in Eq. (3) with r_c. One benefit of a *larger* coverage radius r_c is a smoother noise spectrum [20], defined by the Fourier transform of all pairwise point distances; see Fig. 5. We have also explored adapting a point set to obtain a *smaller* coverage radius, which improves interpolation error and mesh quality, by a method we call opt-beta, locally optimizing the position of nodes [6].

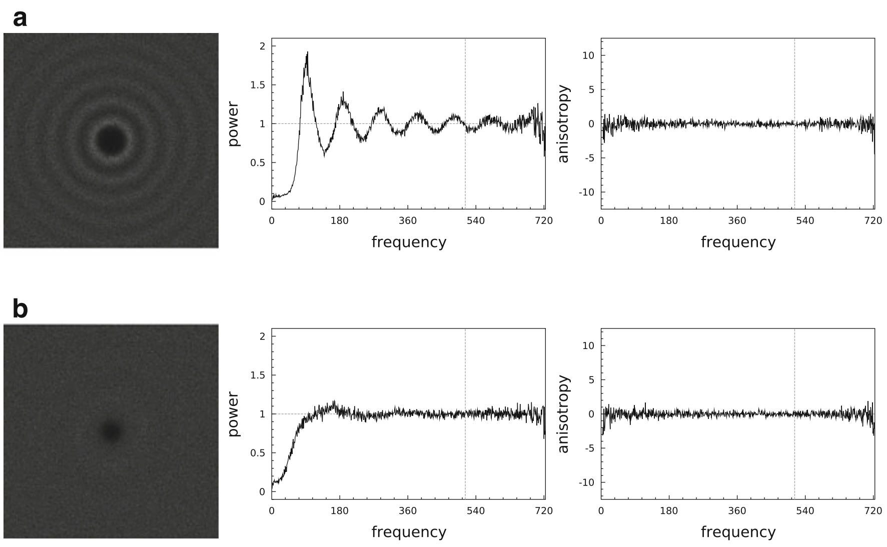


Fig. 5 Spectral properties [20] of single-radii and two-radii MPS. (**a**) Single radius MPS. (**b**) Two radii, $r_c = 2r_f$

3 Meshing Algorithms Based on MPS

We consider two types of meshes: constrained Delaunay triangulations [10] and Voronoi polyhedra [8]. See Fig. 6 for triangles and Fig. 8 for poyhedra. MPS produces well-spaced points, which can lead to well-shaped elements immediately, without the need for post-processing such as smoothing or edge swapping; see Fig. 7b. For Delaunay triangulations we must place points exactly on the domain boundary to get a conforming mesh. We must place boundary points more densely than in the interior to ensure good quality triangles. In Voronoi meshing, the MPS points are the Voronoi cell seeds, and it is better if the points lie strictly interior to the domain, and the cells are clipped by the domain boundary. Using interior points, cells have better aspect ratios, and the Voronoi mesh is smaller; see Fig. 8 cut-away.

Delaunay refinement [4] is the most popular method for well-spaced points and well-shaped triangles. It creates triangles first, and adds points to remove bad-quality triangles. In contrast, MPS generates well-spaced points first, and only forms triangles at the end. The theoretical guarantees about the outputs, the numbers of points and elements' qualities, are nearly identical. In practice, freedom from the vagaries of intermediate triangles appears to allow MPS to change mesh size more quickly [1], and more closely adhere to a sizing function, for the same quality

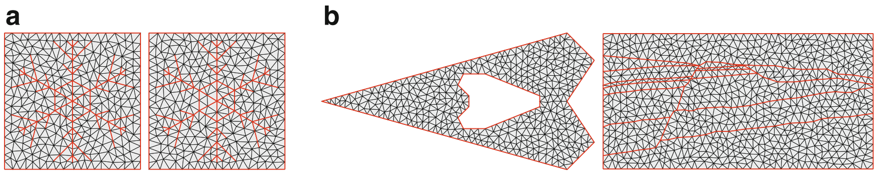


Fig. 6 Delaunay triangulations via Poisson-disk sampling. (**a**) Two random meshes with the same radius and domain. (**b**) Random meshes of non-convex domains. *Red internal interfaces* are represented in the mesh

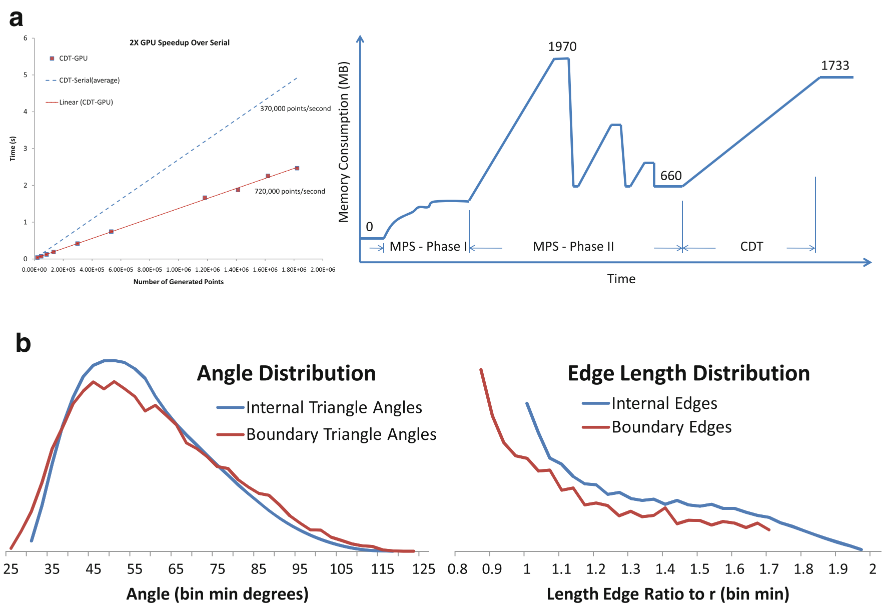


Fig. 7 Algorithm performance and mesh quality for our MPS triangle mesher. (**a**) Near-linear run-time (*left*) and memory (*right*). (**b**) Provably-good angle and edge length quality

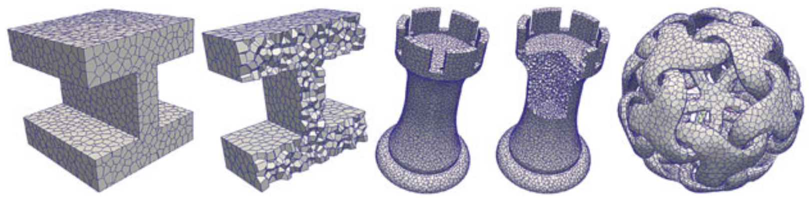


Fig. 8 MPS Voronoi meshes. The boundary elements are nearly as large and well-shaped as the interior elements. Operations are local so it is easy to handle complicated global topology

requirements. The local nature of MPS operations leads to near-linear time in serial; see Fig. 7. It also leads to easy parallelism with little communication [10]. In contrast Delaunay refinement must build very large intermediate triangles between distant points.

4 Sampling in High Dimensions

4.1 Algorithmic Challenges

Although the definition of MPS is dimension-independent, some properties of the distribution change as the dimension increases, and the algorithms that work well in low dimensions take more memory and time. High-dimensional spaces present several challenges. As dimension increases, the volume of a sphere relative to its bounding box decreases, and relative to its inscribed box it increases. A unit box can contain an exponential-in-d number of unit disks. The box becomes a worse approximation of a sphere, and this dooms grid-based methods as expensive in time, memory, or both. Regardless of the methods used to generate the sampling, each disk can have exponentially more nearby disks as the dimension increases. This increases the combinatorial complexity of computing intersections of disks, or disks with grids.

If we take a step back from MPS, a typical underlying goal is to sample the space in a way that gives unbiased estimates, with low variance, of some quantity. Typically this still means our sampling should be random. However, points are just one way of sampling, a zero-dimensional way. MPS points are chosen uniformly at random *by volume* without regard to the shape of the domain. Thus it is hard to hit narrow regions with point samples, even though some of their dimensions might be large. In particular, in standard MPS algorithms it is hard to tell if maximality has been reached and the domain is covered by disks. More generally, it is hard to track narrow regions of interest. In uncertainty quantification, the domain may be the parameter space of a simulation, and the region of interest is where the simulation returns a value below a threshold. Typically the simulation is more sensitive to some parameters than others, so this subregion is narrow in those dimensions, but large in the insensitive parameter directions.

4.2 Algorithmic Solution: k-d Darts

To address such a scenario, rather than evaluating a function at a single point, we consider higher-dimensional evaluations along k-dimensional hyperplanes or "flats" [13]; see Fig. 9. Initially, we defined a "k-d dart" as a set of axis-aligned hyperplanes spanning all combinations of k free and $d - k$ fixed coordinates. How-

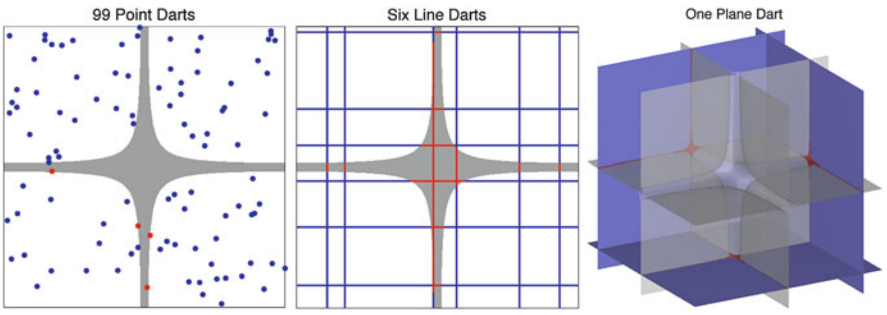


Fig. 9 Sampling a narrow region with hyperplanes increases the amount of information we capture. While the number of random points landing in the gray area approaches zero as its thickness decreases, each random line is destined to capture a portion of it

ever, further analysis and experimentation showed that just picking k-dimensional hyperplanes with random free-coordinate indices is simpler and works just as well. This is all predicated on the assumption that it is possible to evaluate a function along a flat. A particularly nice situation is when the function is analytic, and we can substitute the flat's fixed coordinates for its parameters. As a last resort, one could estimate the evaluation of a function along a flat numerically. In general the cost of evaluating a higher-dimensional flat is typically larger than a 0-d flat (i.e. a point). However, the amount of information gained and the ability to find narrow regions is often worth the cost. We get faster convergence, and higher quality in our examples.

5 High-Dimensional Algorithms Using k-d Darts

We next show the utility of darts over several algorithmic examples: generating relaxed maximal Poisson-disk samples, approximating depth of field blur [13, 21], and volume estimation [13].

5.1 Relaxed Maximal Poisson-Disk Sampling (RMPS)

In a maximal Poisson-disk sample, the coverage disks overlap to cover the entire domain, leaving no room to increase the sample size. At maximality, the achieved coverage radius r_c is at most the prescribed conflict radius r_f; recall Sect. 2.3. Achieving this condition is extremely hard in high dimensions (e.g. $d > 6$). A relaxed version of this condition allows the coverage disks to be slightly larger than the conflict disks, quantified by their ratio, the *distribution aspect ratio* $\beta = \frac{r_c}{r_f} > 1.0$. Increasing the allowable upper bound on β makes solving the problem easier.

k-d darts [13] utilize line darts to capture the narrow voids between existing conflict disks while simultaneously estimating the volume of the remaining void. Line darts are much faster than point darts for RMPS, because it is fast and easy to subtract a set of spheres from an axis-aligned line. The remaining segments are uncovered, and we introduce a new sample along them.

Our k-d darts RMPS method produces nearly-maximal distributions whose spectral properties (randomness) are nearly identical to those of MPS. It uses much less memory than previous methods, allowing us to examine larger and higher-dimensional domains. We sample domains of up to six dimensions on a CPU, and three dimensions on a GPU.

5.2 *Depth of Field by Line Darts (DoF)*

Optical camera lenses focus at a single distance from the camera, so captured images typically exhibit blur effects at other distances. Computer-generated images, on the other hand, by default are in focus at all distances. Simulating the depth-of-field effects of a real camera is helpful in adding realism to computer graphics. Using one-dimensional darts or line darts or line samples [13, 21], one can produce high-quality low-noise images with depth-of-field effects. Compared to traditional point sampling, line darts are able to capture more information per sample. Although each line dart is computationally more expensive than a point dart, in practice line-darts reduce the overall run time to produce an image of comparable quality.

Depth of field involves sampling the image in 4-d (x, y, u, v) space, where (x, y) is screen space and (u, v) is lens space. In k-d darts [13], each dart sample consists of four orthogonal lines, one spanning each of the 4-d coordinates. In *wagon wheel* [21], each dart sample consists of a radial line in (u, v) space, passing through the center of the lens; see Fig. 10. The remainder of this section elaborates on the wagon wheel approach.

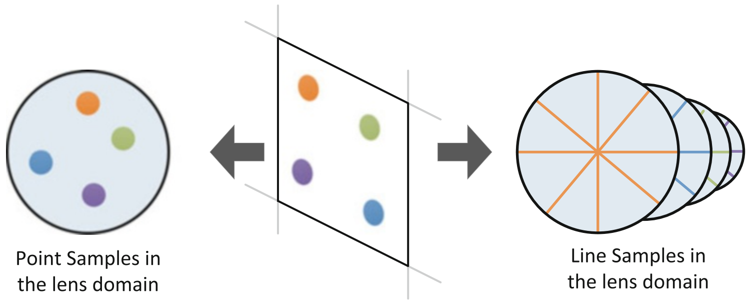


Fig. 10 Depth of field line sampling vs. point sampling. We have four colored samples in scene space (*middle*). For each one, point sampling generates one point in lens space (*left*), while our method generates multiple line samples (*right*)

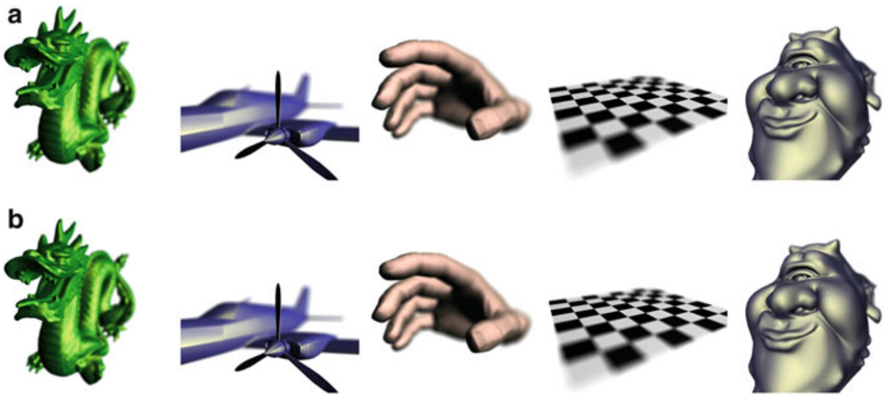


Fig. 11 256 point samples (*top*) and 16 wagon-wheel line samples (*bottom*) per pixel produce nearly the same image, but line samples are about four times faster. (**a**) Example 800 × 800 pixel scenes rendered using 256 *point* samples. (**b**) Example 800 × 800 pixel scenes rendered using 16 *line* samples

For each pixel we generate several sampling locations (x_p, y_p) within the pixel and then perform line sampling along the lens (u, v) space. Rendering starts by computing intersections between line darts and incoming primitives that represent the scene. Primitives that intersect the line samples then generate colored line segments, whose contribution is aggregated in to the final color of the pixel. How line samples are placed along the lens plays a crucial role in the final image quality.

A wagon wheel line sampling pattern (Fig. 10) has several advantages when compared to alternatives. First, it has uniform line sample lengths. This helps when implementing the algorithm on highly parallel architectures, such as a GPU. Second, line samples passing through the origin can be expressed in a simple slope-intercept form $v = mu$, simplifying the math. Lastly, the bias that is associated with such a pattern can be easily removed by a reweighting during filtering. An alternative would be a grid-like pattern, as the ridges of a waffle, using k-d darts in just (u, v) space.

Figure 11 shows the results of several scenes rendered with 256 point samples versus 16 wagon wheel line samples, run in parallel on a NVIDIA GTX 580 GPU. Also, for k-d darts, 16 line-darts produce a better picture, more quickly, than 1024 points. Using either wagon wheels or k-d darts, line sampling demonstrates a clear win in terms of both quality and performance.

5.3 Volume Estimation by Hyperplane Darts

We study the accuracy of high-dimensional sampling using k-d darts, on the classical problem of estimating the volume of an object [13]. We seek to experimentally quantify the effects of the object orientation, dart orientation, object surface area to volume ratio, and dimensions of the object and dart. We create a d-dimensional hyper-ellipsoid object as follows.

– Start with a unit d-ball (d-dimensional unit sphere);
– Scale along the x-axis by a factor squish s to generate an elliptical profile; and
– Perform r random Givens rotations to randomly orient it.

We compute the volume of the ellipsoid analytically, for the ground-truth. For comparison, we estimate the volume of the ellipsoid using $k = 0$ darts, i.e. classical Monte Carlo point sampling: sample random points from the ellipsoid's bounding box and count the fraction inside the ellipsoid. The accuracy decreases as d increases.

For k-d darts, we can choose to throw line darts, plane darts, or darts of any dimension $k < d$. Figure 12 show how k-d darts consistently outperform point darts. Their advantage increases as k increases.

Darts may be axis-aligned or arbitrarily oriented. We recommend axis-aligned darts for three reasons. First, it is easy to distribute aligned darts uniformly, which ensures that the expected mean of the function estimates is accurate. Second, it is easiest to implement aligned darts, since it involves simply fixing coordinate values. Third, in many cases it is most efficient because we may obtain an expression for the underlying function along a dart by substituting in the fixed coordinate values. We compare the accuracy of aligned and unaligned darts in the top row of Fig. 12. For

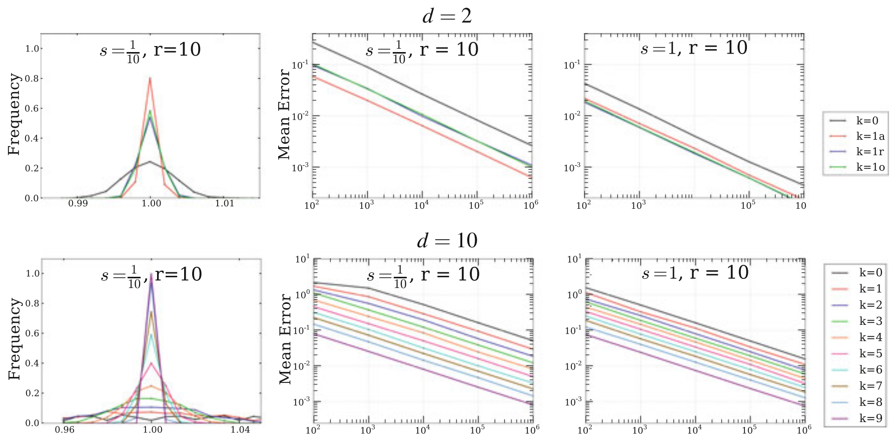


Fig. 12 d-Dimensional hyper-ellipsoid volume estimation, for varying squish factor s scaling the main axis, dart dimension k, and fixed number of random rotations $r = 10$ of the ellipsoid. Top uses axis-aligned (1a), randomly oriented (1r), and orthogonal pairs of randomly oriented darts (1o). Bottom uses axis-aligned darts in 10-d. *Left* shows the ratio of the estimated to true volume by frequency for a fixed number of samples, n. *Right* shows $|\text{mean} - \text{true}|/\text{true volume}$ by the the number of darts, n

squished ellipsoids, aligned darts are slightly more accurate, but the prior reasons are more significant.

6 Summary

We have highlighted the main research results of our sampling group from 2011 to 2013. We consider MPS, or more generally point sets that are both well-spaced and random, to be a very rich area, crossing many fields and applications. Over the course of our research, we have explored many modifications to our algorithms and used the output for many different applications. Uses and features of our sample generation and modification algorithms are summarized in Tables 1 and 2. We have compared and contrasted these to the works of others. These variations [6, 9], and their tradeoffs, led us to the conceptual framework for sampling illustrated in Fig. 13.

Table 1 Our sample generation and modification algorithms' application context

Name	Goals	Mesh	Nodes	Beta	Dim
Opt-beta	Reduce beta	Ok	Move	Tune < 1	2
Steiner reduction	Fewer points, preserve mesh angles	Yes	(Re)move	≥ 1	2
Sifted disks	Fewer points, preserving beta $= 1$	Ok	(Re)move	1	2
k-d darts	Create points, integration, in high dimensions		Create	$\rightarrow 1$	2–23
Variable radii MPS	Create points with sizing function or spacing	Yes	Create	Tune ≥ 1	2
VorMesh MPS	Create Voronoi polyhedral mesh	Yes	Create	1–2	2–3
DelMesh MPS	Create Delaunay triangle mesh	Yes	Create	1–2	2–3
Simple MPS	Create points in moderate dimensions	Ok	Create	1	2–6
Efficient MPS	Create points in two dimensions	Ok	Create	1	2

Table 2 Our algorithms' performance and features

Name	GPU	Time	Memory	Features	Ref.
Opt-beta		Numeric	n	Local position optimization, as smoothing	[6]
Steiner reduction		$\approx n$	n	2-to-1 node replacement	[1]
Sifted disks		n	n	2-to-1 node replacement	[9]
k-d darts		n^2d^2	nd	High-d, global hyperplanes, many applications	[13]
Variable radii MPS		$\approx n$	n	Provable qualities	[16]
VorMesh MPS		$\approx n$	n	Bounded domains, provable quality	[8]
DelMesh MPS	GPU	$\approx n$	n	Bounded domains, provable quality	[10]
Simple MPS	GPU	$\approx n2^d$	$n2^d$	Efficient flat quadtree tracking voids	[11]
Efficient MPS	GPU	$n \log n$	n	Provable runtime, polygon approx. voids	[12]

Many [8, 10, 16] use a form of Simple MPS [11]

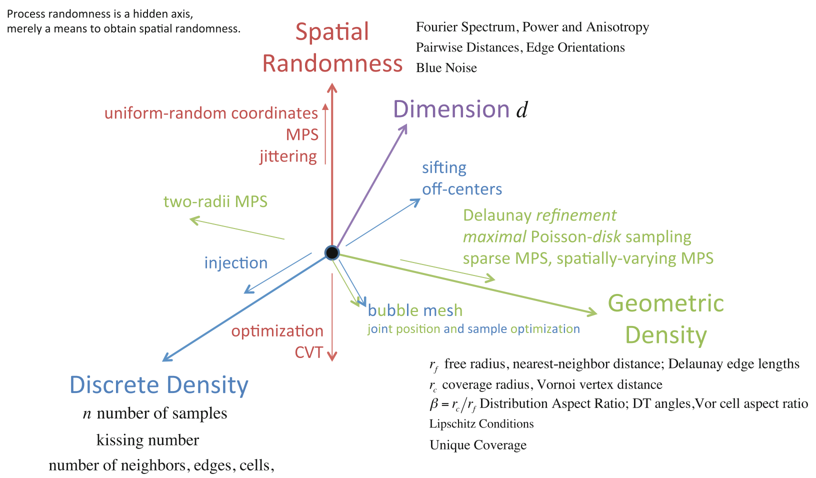

Process randomness is a hidden axis, merely a means to obtain spatial randomness.
Spatial Randomness
Fourier Spectrum, Power and Anisotropy
Pairwise Distances, Edge Orientations
Blue Noise
uniform-random coordinates
MPS
jittering
Dimension d
sifting
off-centers
two-radii MPS
Delaunay *refinement*
maximal Poisson-*disk* sampling
sparse MPS, spatially-varying MPS
injection
bubble mesh
joint position and sample optimization
optimization
CVT
Geometric Density
r_f free radius, nearest-neighbor distance; Delaunay edge lengths
r_c coverage radius, Vornoi vertex distance
$\beta = r_c/r_f$ Distribution Aspect Ratio; DT angles,Vor cell aspect ratio
Lipschitz Conditions
Unique Coverage
Discrete Density
n number of samples
kissing number
number of neighbors, edges, cells,

Fig. 13 A conceptual space parameterizing the output of any sampling method. Note methods such as jittering add randomness, while optimization methods tend to remove randomness. Each axis may be varied, but the axes are not independent. For example, two-radii MPS places points further apart, and hence produces fewer of them, and their positions are more random. Achieving maximality tends to add regularity and improve mesh quality. Bubble mesh interleaves changing the position and number of points; again optimizing positions leads to greater regularity

Acknowledgements The UC Davis authors thank the National Science Foundation (grant # CCF-1017399), Sandia LDRD award #13-0144, UC Lab Fees Research Program Award #12-LR-238449, NVIDIA and Intel Graduate Fellowships, and the Intel Science and Technology Center for Visual Computing for supporting this work.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. Abdelkader, A., Mitchell, S.A., Ebeida, M.S.: Steiner point reduction in planar Delaunay meshes. In: ACM Symposium on Computational Geometry. Young Researchers Forum (2014) http://www.computational-geometry.org/YRF/cgyrf2014.pdf and http://www.cs.sandia.gov/~samitch/bibliography_2007.html#steiner-reduction
2. Bolander Jr., J.E., Saito, S.: Fracture analyses using spring networks with random geometry. Eng. Fract. Mech. **61**(5–6), 569–591 (1998)
3. Bowers, J., Wang, R., Wei, L.-Y., Maletz, D.: Parallel Poisson disk sampling with spectrum analysis on surfaces. In: SIGGRAPH Asia '10, pp. 166:1–10 (2010)
4. Chew, L.P.: Guaranteed-quality mesh generation for curved surfaces. In: Proceedings of the Ninth Annual Symposium on Computational Geometry, pp. 274–280 (1993)
5. Dunbar, D., Humphreys, G.: A spatial data structure for fast Poisson-disk sample generation. ACM Trans. Graph. **25**(3), 503–508 (July 2006)
6. Ebeida, M.S., Awad, M.A., Ge, X., Mahmoud, A.H., Mitchell, S.A., Knupp, P.M., Wei, L.-Y.: Improving spatial coverage while preserving the blue noise of point sets. Computer-Aided Design **46**, 25–36 (2014)
7. Ebeida, M.S., Knupp, P.M., Leung, V.J., Bishop, J.E., Martinez, M.J.: Mesh generation for modeling and simulation of carbon sequestration process. In: DOE Scientific Discovery Through Advanced Computing (SciDAC) (2011)
8. Ebeida, M.S., Mitchell, S.A.: Uniform random Voronoi meshes. In: 20th International Meshing Roundtable, pp. 273–290. Springer, Berlin (October 2011)
9. Ebeida, M.S., Mahmoud, A.H., Awad, M.A., Mohammed, M.A., Mitchell, S.A., Rand, A., Owens, J.D.: Sifted disks. Comput. Graph. Forum **32**(2 Pt 4), 509–518 (2013)
10. Ebeida, M.S., Mitchell, S.A., Davidson, A.A., Patney, A., Knupp, P.M., Owens, J.D.: Efficient and good Delaunay meshes from random points. Comput. Aided Design **43**(11), 1506–1515 (2011)
11. Ebeida, M.S., Mitchell, S.A., Patney, A., Davidson, A.A., Owens, J.D.: A simple algorithm for maximal Poisson-disk sampling in high dimensions. Comput. Graph. Forum **31**(2), 785–794 (May 2012)
12. Ebeida, M.S., Patney, A., Mitchell, S.A., Davidson, A., Knupp, P.M., Owens, J.D.: Efficient maximal Poisson-disk sampling. ACM Trans. Graph. **30**(4), 49:1–49:12 (July 2011)
13. Ebeida, M.S., Patney, A., Mitchell, S.A., Dalbey, K.R., Davidson, A.A., Owens, J.D.: k-d darts: Sampling by k-dimensional flat searches. ACM Trans. Graph. **33**(1), 3:1–3:16 (2014). doi: 10.1145/2522528. http: //doi.acm.org/10.1145/2522528
14. Heck, D., Schlömer, T., Deussen, O.: Blue noise sampling with controlled aliasing. ACM Trans. Graph. (TOG) **32**(3), 25 (2013)
15. Jones, T.R.: Efficient generation of Poisson-disk sampling patterns. J. Graph. Tools **11**(2), 27–36 (2006)
16. Mitchell, S.A., Rand, A., Ebeida, M.S., Bajaj, C.: Variable radii Poisson-disk sampling. In: Canadian Conference on Computational Geometry, vol. 24, pp. 185–190 (2012)
17. Nehab, D., Hoppe, H.: A fresh look at generalized sampling. Found. Trends Comput. Graph. Vis. **8**(1), 1–84 (2014)

18. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann (2010)
19. Park, C., Pan, J., Manocha, D.: RealTime GPU-based motion planning for task executions. In: IEEE International Conference on Robotics and Automation Workshop on Combining Task and Motion Planning (May 2013)
20. Schlömer, T.: PSA Point Set Analysis. Version 0.2.2. http://code.google.com/p/psa/ (2011)
21. Tzeng, S., Patney, A., Davidson, A., Ebeida, M.S., Mitchell, S.A., Owens, J.D.: High-quality parallel depth-of-field using line samples. In: Proceedings of High Performance Graphics, pp. 23–31 (June 2012)

Realization of Regular Maps of Large Genus

Faniry Razafindrazaka and Konrad Polthier

1 Introduction

Tiling of closed surfaces into non-overlapping faces is one of the central topics in surface topology and computer graphics. Either the surface is given and a nice tiling of this surface has to be found or the tiling is given and the surface on which this tiling is the most symmetric has to be found. This paper explores the later case but restricts the tiling scheme to the class of regular maps.

The concept of map was first introduced by Coxeter and Moser [2]. A *map* is a family of polygonal faces such that any two faces share an edge or a vertex, or are disconnected; each edge belongs precisely to two faces; the faces containing a given vertex form a single cycle of adjacent faces; between any two faces is a chain of adjacent faces. In other words, it is a closed 2-manifold, without boundaries obtained by glueing topologically equivalent polygonal faces. If the map has p-gonal faces and q-gonal vertex-figures (number of faces around at a vertex), then it is said to be of type $\{p, q\}$.

A *regular map* is a map which is flag transitive. It means that, on the surface, if a vertex or an edge or a face is mapped to another vertex an edge or a face, then the map is one to one and preserves all adjacency properties between the vertices, edges and faces. Regular maps can be viewed as generalization of the Platonic solids to higher dimension. They define regular tilings of closed surface and their group structure can be used to move along the surface which behaves like an “hyperbolic” parameterization.

F. Razafindrazaka (✉) • K. Polthier
Freie Universität Berlin, Berlin, Germany
e-mail: faniry.razafi@fu-berlin.de; konrad.polthier@fu-berlin.de

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_14

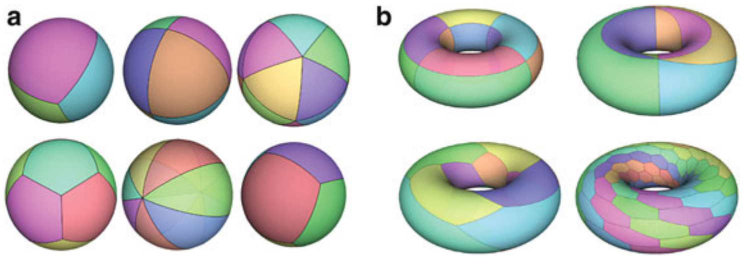


Fig. 1 Example of genus 0 and genus 1 regular maps, where the first set corresponds to the Platonic solids and the second set to tilings of the torus. (**a**) Spherical regular maps. (**b**) Genus 1 regular maps

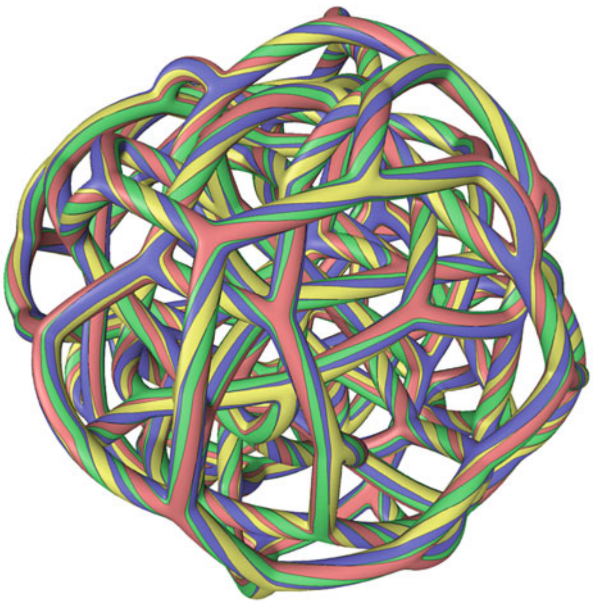


Fig. 2 Visualization of a genus 61 surface tiled with 480 hexagons following the regularity of the map R61.1′{6, 4}

Figure 1 illustrates some examples of low genus regular maps which include the Platonic solids, the Hosohedron and some tiling of the torus.

In this report, we give realizations of regular maps of genus > 20 that we classify to be *large* genus regular maps. In Fig. 2 is an example of a genus 61 regular map which is the highest genus regular map ever visualized so far. Our main contribution is the targetless tubification described in Sect. 4.2. All the images generated in this paper are produced by our algorithm.

The chapter is organized as follows: first, we will give theoretical backgrounds on regular map including geometric and algebraic characterization; second, we will describe the basic procedure to obtain, with maximal symmetry, large genus surfaces; and last, we will give a description of the method used to produce tilings on these large genus surfaces.

2 Related Works

Up to now, there is no general method to visualize regular maps but a lot is already known about their symmetry group, see for example Conder [1]. The problem is two-fold, understanding the symmetry group of the regular map and finding a suitable space model for it. Jack van Wijk [3], in his Siggraph paper, suggested a generic approach which gives interesting visualization of some of the lower genus regular maps up to genus 29. He succeed to handle about 50 cases by using a brute force computer search. However, his method is too restrictive and cannot realize even some of the simplest cases. Séquin's investigations [8–10] are also a huge source of inspiration. He uses physical modelling techniques, including sketches, paper models and Styrofoam, to finally obtain a computer generated model. Some cases have been solved by his method from genus 2 to genus 5 but each regular map is handled separately. Sequin's approach are useful for a better understanding of the structure of regular maps but too primitive to handle the large ones. In our early work [5,6], We uses the same approach as van Wijk but we added a relaxation procedure to obtain more symmetrical and smooth tubular geometry. We use this relaxation scheme as a second step of the targetless tubification algorithm.

In this paper, we aim at surfaces having more than two junctions and with rich structure to accommodate regular maps. We are not interested in the hosohedral kind of surface. These are the surfaces obtained by taking the tubular neighborhood of Hosohedra.

3 Background Notions

3.1 On Standard Geometry

Isometric realization of tilings depends on the ambient spaces where they are embedded. These are: the Sphere, the Euclidean plane and the Hyperbolic plane. For spherical and hyperbolic geometry, we use [7]. Examples of spherical isometric tilings are the Platonic solids; the Euclidean plane can be isometrically tiled by checker-board patterns; and the Hyperbolic plane are tiled by p-gons for large p's. A closed 3D realization of a sub-tiling of the Euclidean plane is a torus (see Fig. 2 in Chapter 6). The closeness of the torus is topologically derived from a parallelogram in the Euclidean plane wrapped in 3D by identifying opposite sides. In this 3D realization, isometry is lost but the topology of the tiling is still preserved. We then only talk about combinatorial transitivity.

Similar 3D realizations can also be done from the Hyperbolic plane. A genus $g > 1$ surface is derived by taking a $4g$-gon in the Hyperbolic plane and identifying pairwise edges. Hence, any tiling of the Hyperbolic plane can be realized as 3D surfaces by finding a $4g$-gon partitioning this tiling with the correct identification at the boundary. Special case of these tilings are regular maps.

3.2 *On Regular Map*

A *finitely generated group* is a group of the form $\langle \mathscr{G} \mid \mathscr{R} \rangle$, where $\mathscr{G}$ is a set of generators and $\mathscr{R}$ is a set of relations. If $R_i \in \mathscr{R}$, then $R_i = I$ which is the identity of the group.

A regular map is a finitely generated group of the following form

$$\mathrm{Sym}(M_S) = \left\langle R, S, T \,|\, R^p, S^q, T^2, (RS)^2, (ST)^2, (RT)^2, \mathscr{R}_1, \ldots, \mathscr{R}_m \right\rangle, \tag{1}$$

where R is a rotation of $2\pi/q$; S is a rotation of $2\pi/p$ and T is a reflection. They are transformations acting on a fundamental triangle with corner angles $\pi/p, \pi/q$ and $\pi/2$ (see Fig. 3). Depending on p and q, they can be euclidean motions, special orthogonal matrices (for spherical) or Moëbius transformations (for hyperbolic). $\mathscr{R}_1, \ldots, \mathscr{R}_m$ are extra relations making the group finite. The expression 1 is called the symmetry group of the map. It is the set of all automorphisms of the regular map [2].

The symmetry group of the cube (a regular map of type $\{4, 3\}$) is defined by

$$\mathrm{Sym(Cube)} = \left\langle R, S, T \,|\, R^4, S^3, T^2, (RS)^2, (ST)^2, (RT)^2 \right\rangle, \tag{2}$$

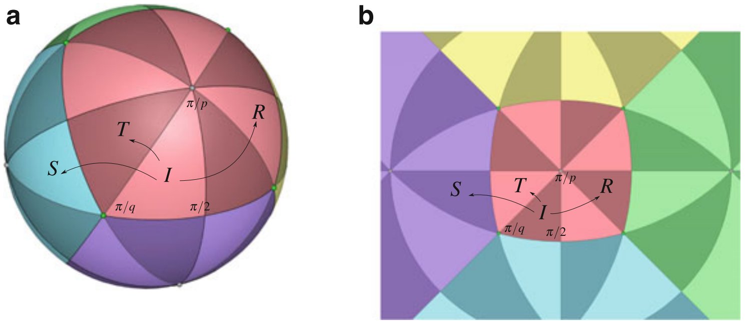


Fig. 3 Two different representations of the Cube using: (**a**) a Sphere and (**b**) stereographic projection into the plane

Sym(Cube) can be realized on a blown up Cube, taking as fundamental triangle a spherical triangle with corner angles $\pi/4, \pi/3$ and $\pi/2$ (Fig. 3a). It can also be visualized as a 2D surface using stereographic projection which is only conformal (Fig. 3b) but not isometric.

Orientable regular maps are denoted in Conder [1] by R$g.i\{p,q\}$, which is the ith−reflexible orientable map of genus g. $\{p,q\}$ is the Schläfi symbol of the tiling. Reflexible means that the transformation T in Eq. 1 is also an automorphism of the map. Analogously, the dual map is represented by R$g.i'\{q,p\}$. Conder [1] listed all reflixible regular maps of genus 2 to 302. They are given as symmetry groups and used as input to our algorithm.

4 Generating Large Genus Surfaces

In this section, we explore in depth techniques to generate and visualize large genus surfaces. Our aim is not only to generate some genus g surface but also a surface with rich topological structure and nice looking shape.

4.1 Tubification Process

A genus g surface can be generated by taking a sphere and drill non intersecting g tunnels on it. Another approach, very used for teaching, is the sphere with g handles. It consists mainly of taking Tori and glueing them on a Sphere to form handles. It is then unclear where the Tori should be placed and if the resulting surface can be used to visualize symmetric tilings.

A better approach is the use of a *tubification* process. It consists of taking a tiling of a surface, turning its edges into tubes, its vertices into junctions and its faces into tunnels. For example, a genus 2 surface can be derived from a tubified Hosohedron $\{2,3\}$ as illustrated in Fig. 4b. Surfaces with rich structure or regular surfaces can be derived by taking regular tilings having more than two vertices. The Platonic solids are direct examples of these. Even more, we can take any regular maps and apply

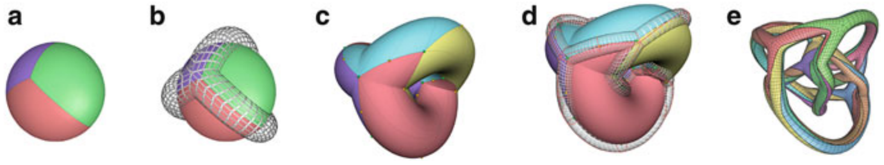


Fig. 4 Recursive tubification starting from: (**a**) hosohedron $\{2,3\}$, (**b**) tubification of its edge graph, (**c**) regular map R2.8′$\{8,3\}$, (**d**) tubification of its edge graph and (**e**) regular map R9.3′$\{6,4\}$

the tubification process to derive large genus surfaces. In Fig. 4c, d, we show an example of a tubification of the regular map R2.8′$\{8, 3\}$.

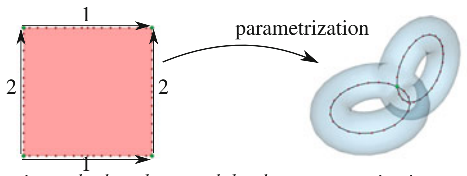
As in [3], a pairing of source and target map is used to generate higher genus surfaces. The source map is the actual regular map that we want to realize and the target map is the regular map which after tubification gives a space model for the source map. More precisely, let $(M_i)_{i \in I}$ be a finite sequence of regular maps such that a space model of M_{i+1} is the tubification of M_i. For a given n, if for all $i < n$, the M_i's are realized, then a tubification of M_{n-1} is a space model of M_n. Otherwise, we cannot give a space model for M_n. This become now a classical method used to visualize successfully large class of regular maps. In the next section, we show that, in fact, the sequence of M_i's is not needed, only the pairing of source-target map is enough.

4.2 Targetless Tubification

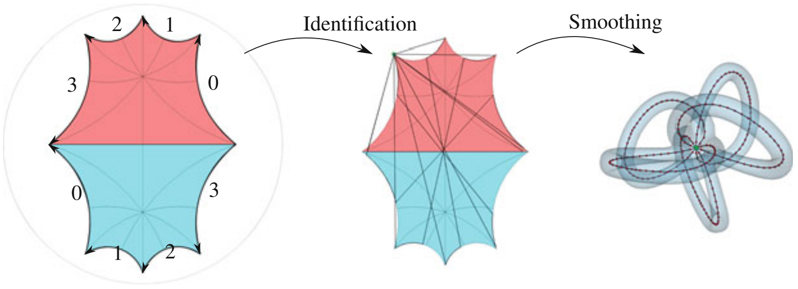
The tubification of an existing regular map has a critical issue since it needs an actual realization of the target regular map. Hence, if the target regular map does not have a 3D embedding, then the tubification cannot be applied and thus no higher genus surface is generated.

We give a solution to this restriction by taking advantage of the planar representation of the target regular map. We call the process a *targetless tubification*. Targetless in the sense that no actual 3D embedding of the target regular map is needed but only an embedding of its edge graph is sufficient.

We generalize the torus case to Hyperbolic space. More precisely, suppose we have a tiling of a flat torus with its 2D edge graph. This edge graph can be mapped to 3D using the usual parameterization function and hence a tubular surface is derived naturally. In this process, only the edge graph is needed to be embedded in space, not the 2D tiling. We do also the same process in Hyperbolic space but since we do not have an explicit parameterization, we do as follows: first, we identify explicitly boundary edges of the map and second, we apply the constrained relaxation procedure described in [5] to get symmetry and smoothness. All the higher genus surfaces in this paper were produced by this simple procedure. An example of a genus 5 surface obtained by the edge graph of the regular map R2.4$\{5, 10\}$ is illustrated in Fig. 5. In this illustration, we start with the hyperbolic realization of R2.4 with the identification at the boundary (represented by the arrows). We then match the boundary edges having the same label, head to head and tail to tail. This results in a 2D connected graph which has the same combinatorics has the edge graph of the underlying regular map. This 2D graph is then smoothed using spring energy.



Mapping only the edge graph by the parameterization



Mapping the edge graph by identifying explicitely the edge greaph

Fig. 5 Construction of a high genus surface by embedding directly the edge graph of the target regular map

Notice that no actual embedding of the regular map R2.4 is needed (see [10] for a 3D realization of this map). Our technique can be applied to any planar representation of a regular map to generate a 3D tubular surface obtained from its edge graph. Below are some examples of large genus regular map generated by the above method.

5 Topological Group Structure

In this section, we define a topological group structure on the surface generated previously that we denote S_g.

5.1 *Partition by Tube Elements*

The recursive tubfication procedure derived in Sect. 4.1 allows us to choose a tiling of S_g. We have for example a tiling with quarter-tubes, with half-tubes, with tube junctions, with full tubes or with multiple quarter-tubes (see Fig. 6). We call one of these a *fundamental domain* of S_g, and as for every group, we can cover the surface by copies of this fundamental patch. These tilings are induced naturally from the underlying regular map used to derive the tubular surface.

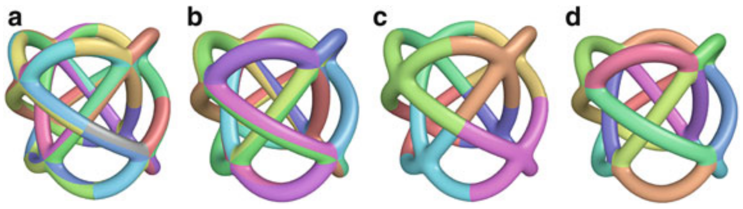


Fig. 6 Example of a partition of S_g with some elements of the tubes. (**a**) Quarter-tubes; (**b**) half-tubes; (**c**) tube junctions; (**d**) full tubes

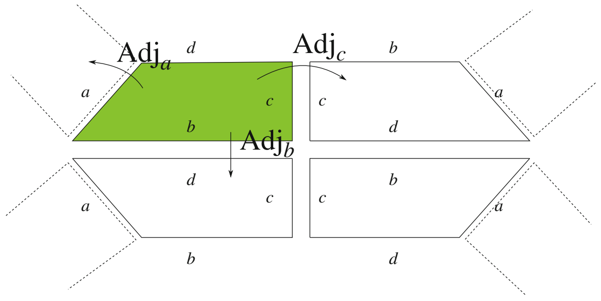


Fig. 7 The adjacency operator defined on the set of quarter-tubes

The next step is now to define a group structure induced by the tube element in order to define a parameterization of S_g. This parameterization will be then used to map other regular maps as described in Sect. 6.

5.2 *Deriving the Symmetry Group*

We restrict our construction to the case of a tiling with quarter-tubes as in [3]. The other cases can be handled analogously.

Let $\mathscr{Q}$ be the set containing all colored quarter-tubes of S_g. We label the edge of a quarter-tube by a, b, c and d, where a is the one at the junction and b, c, d are the next counter-clockwise edges. The orientation is defined by the normal of the surface at each quarter-tube (Fig. 7).

We define a basic operation Adj_x on $\mathscr{Q}$ which takes a quarter-tube Q and returns the quarter-tube adjacent to Q at edge x:

$$\begin{aligned} \mathrm{Adj}_x\colon \mathscr{Q} &\to \mathscr{Q} \\ Q &\mapsto \text{quartertube adjacent to } Q \text{ at edge } x \end{aligned}$$

For example, $(\mathrm{Adj}_b)^2$ is the identity since making two quarter-tube steps around a tube get back to the start. $(\mathrm{Adj}_a)^2$ is also the identity. Let Q_I be a fundamental domain of $\mathcal{Q}$ (it can be any quarter-tube of $\mathcal{Q}$). We define three operations A, B and C on $\mathcal{Q}$ as follows:

- A shifts Q_I two positions positively around a hole, more precisely $Q_A = \mathrm{Adj}_a(\mathrm{Adj}_c(Q_I))$;
- B rotates Q_I around the junction, $Q_B = \mathrm{Adj}_d(\mathrm{Adj}_a(Q_I))$;
- C shifts Q_I one position down, $Q_C = \mathrm{Adj}_b(Q_I)$.

Here, Q_M denotes the quarter-tube obtained by applying a transformation M to Q_I.

We can see A as a transformation moving a tube around a hole, B switches from one hole to another hole and C enables to reconstruct a full tube from a quarter of a tube. Using Adj_x , we can derive the following relation

$$(CBA)^2 = (BA)^2 = (CB)^2 = I$$

where, I denotes the identity transformation. Using the underlying symmetry group of the tiling used to build S_g we can define a symmetry group of S_g as

$$\mathrm{Sym}(S_g) = \langle A, B, C \mid A^{p_t}, B^{q_t}, C^2, (CB)^2, (BA)^2, (CBA)^2,$$
$$g_1(A, B, CB), \ldots, g_n(A, B, CB)\rangle ,$$

where, the g_i's are the extra relations of the symmetry group of the underlying regular map.

A group structure on the genus 5 surface shown in Fig. 8 is given by

$$\mathrm{Sym}(M_5) = \langle A, B, C \mid A^4, B^3, C^2, (BA)^2, (CB)^2, (CBA)^2\rangle .$$

This group has exactly 12×4 quarter-tubes highlighted in Fig. 8. Once the group structure is introduced on S_g, we can unfold this surface in hyperbolic space to embed a regular map on it.

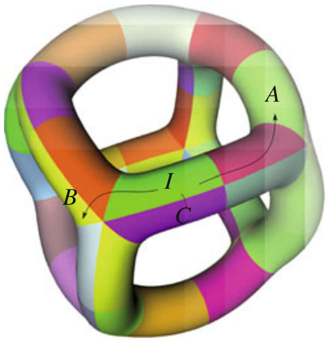


Fig. 8 Three adjacency operators acting on the quarter-tube tiling: A shifts Q_I two positions positively around a hole; B rotates Q_I around the junction; C shifts Q_I one position down

6 Tilings

6.1 *Matching Symmetry Groups*

In this section, we give a brief description of the use of the symmetry group introduced on S_g to realize a regular map. The regular map is defined with its symmetry group $\mathrm{Sym}(S_{\mathrm{map}})$ realized as planar tiling in Hyperbolic space.

The first step is to make an hyperbolic parameterization of S_g. This process is similar to the torus case (space model for genus 1 regular map) where the parameterization is done onto the unit square. For high genus surfaces, the parameterization is done by choosing a suitable fundamental quadrilateral hQ_I in hyperbolic space and set it as fundamental domain of $\mathrm{Sym}(S_g)$. The idea here is to make a 2D realization of $\mathrm{Sym}(S_g)$ using another fundamental domain. Once the parameterization is done, the regular map can be naturally mapped using the inverse mapping. An overview of the algorithm is illustrated in Fig. 9. The remaining problem is then on the construction of hQ_i and the hyperbolic transformations corresponding to the elements of $\mathrm{Sym}(S_g)$.

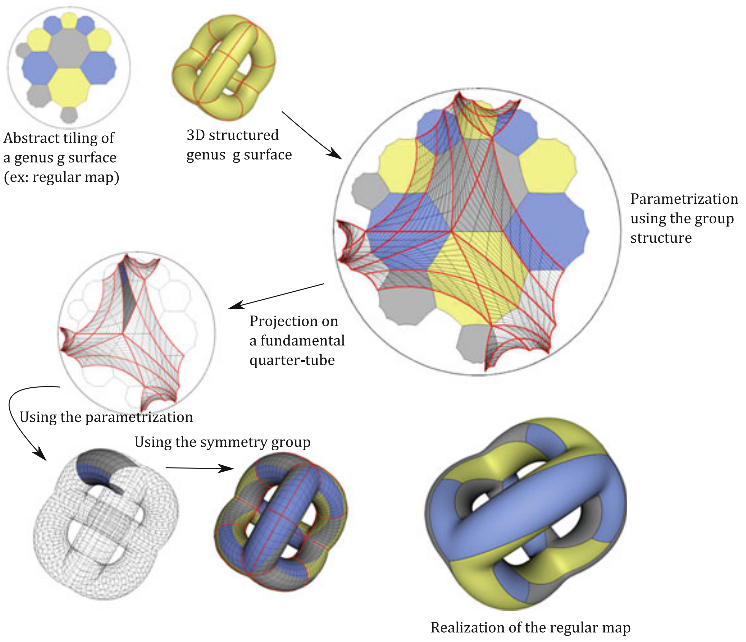


Fig. 9 Pipeline to visualize a regular map on a structured genus g surface

The construction of hQ_I depends on the matching between Sym(S_g) and Sym(S_{map}). These matchings are heuristics which check if there exists a partition of Sym(S_{map}) by Sym(S_g). A necessary condition is that the order of Sym(S_g) should dives the order of Sym(S_{map}) and a sufficient condition is the existence of a subgroup of Sym(S_{map}). In the successful case, hQ_I can be constructed, otherwise S_g is not a suitable space model for Sym(S_{map}) and the mapping cannot be done. Matchings between regular maps are generated using van Wijk [3] heuristics. It consists of a pairing of source map and target map where the second is a lower genus regular map which after tubification gives a space model the first. His heuristic also provides the exact position of the four points of hQ_I in hyperbolic space.

In his lists, there are several mappings which cannot be visualized since the target map does not have a 3D realization. This is mainly the case for the large genus regular maps which depend on several low genus ones. These cases are handled by the targetless tubification in Sect. 4.1.

7 Geometric Construction

In this section, we assume that the target map has a 3D realization and describe the geometrical tools used to construct the tubes. As stated in Sect. 4, the tubes are derived from the edges of an existing regular map. These edges define a control skeleton $\mathscr{S}$ of the tubular surface and is used to deform the shape of the tube.

The normals along $\mathscr{S}$ are induced from the surface. Using these normals, we construct the tube from four quarter tubes, highlighted in Fig. 10. Each quarter tube consists of half ellipses connected to each other which starts at the center of an edge and rotates from 0 to $\phi/2$ when approaching a junction, where ϕ is the angle with the adjacent edge at the junction. This is illustrated in Fig. 10.

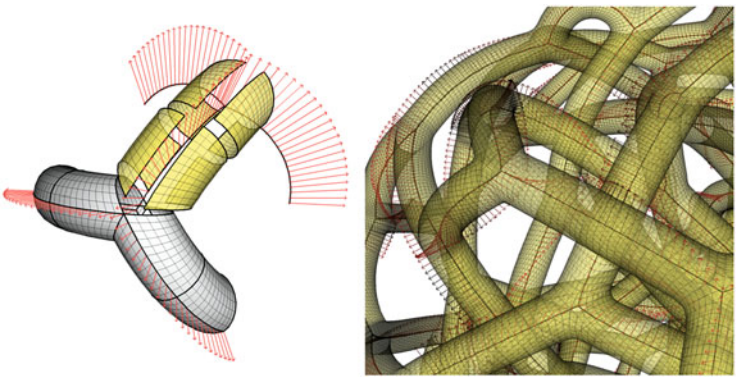


Fig. 10 Generating tubes from quarter tubes. The *red arrow* represent the normals along the edge of the tiling, induced from the surface (*left*). At the junction, each quarter tube meets with the correct angle and then identified to form a connected surface (*right*)

We did not write the mathematical construction of the tubes here, this is extensively studied in [5]. For the targetless tubification, the normals are obtained from the local orientation at the node junctions. These local orientations are derived from the group structure of the underlying regular map which tells us that it must be an orientable surface and hence the normals along its edge-graph must be continuous (take a normal and parallel transport it along any cycle of the graph, then it should give the same normal when it comes back). Normals along an edge are generated by interpolating the two normals at the two endpoints of its junction.

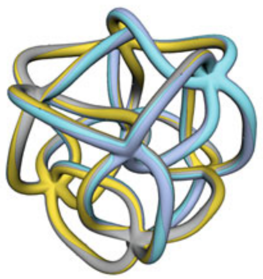
For a better smoothness at the junctions, we do a Catmull-Clark smoothing [4]. New points are not inserted but are only used as mask to relax the points. This is similar to a laplacian smoothing for quad meshes.

8 Examples of High Genus Regular Map

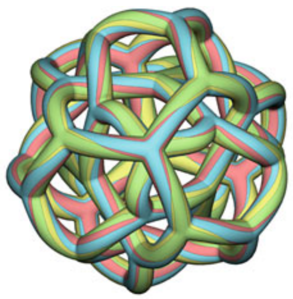
In Fig. 11, few examples of large genus regular maps generated by our targetless tubification are illustrated. The arrows are the matchings between source and target map. These are found using the heuristics presented in [3]. We choose especially large genus regular maps which are closed to spherical and euclidean tilings. Namely, maps for which the integer distance between p and q is not so big.

We succeed to generate all the regular maps missing in [3] which are more than two-fold increases of the current results. In this paper, we emphasis on the visualization of large genus regular maps with self-intersection free even for very high genus surfaces.

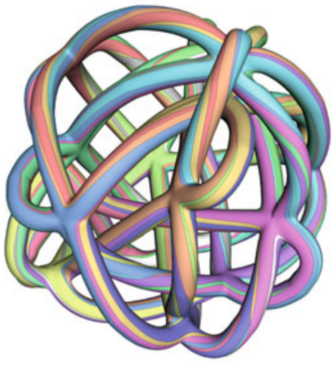
The choice of the tube radius is crucial in this process but it is closely related to the spring energy parameter (attraction and repulsion). Hence, we leave it interactive and modified by visual inspection. In all of our experiment, only few adjustment is needed to have a non self-intersecting surface.



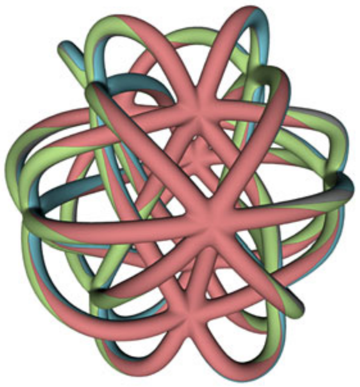
$R17.11'\{12,4\} \rightarrow R6.8'\{8,6\}$



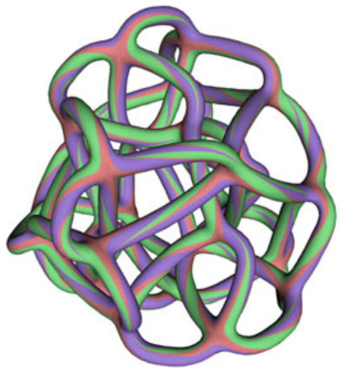
$R21.3'\{4,6\} \rightarrow R5.2'\{10,3\}$



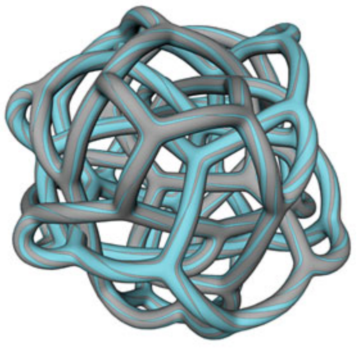
$R25.7'\{10,4\} \rightarrow R5.9\{5,5\}$



$R25.8'\{12,4\} \rightarrow R10.16'\{12,6\}$



$R28.6'\{6,4\} \rightarrow R10.10'\{12,4\}$



$R31.3'\{6,4\} \rightarrow R10.3'\{15,3\}$

Fig. 11 Some large genus regular maps generated by the targetless method

Conclusion
We presented a method to generate large genus regular maps. Regular maps are generalization of the Platonic solids into higher genus surface. These are realized by using a new targetless tubification procedure which does not require any actual embedding of a target shape to generate a genus g surface.

Regular map is an intriguing surface and having a nice visualization of them remains an interesting and unsolved problem. So far, we did not find any practical application of those shapes if not for symmetric tiling of closed surface. They can also be good models used for teaching and understanding how symmetry group work.

Similar to the Platonic solids, regular maps are the most symmetric tiling we can use for high genus surfaces but we need to find the correct space model where these symmetries can be appreciated.

What is not described in this paper is an automatic algorithm which gives the identified edges in the planar representation of the target maps. We will leave this detail as future work.

Acknowledgements We thank Jack van Wijk for material support. This research was supported by the DFG-Collaborative Research Center, TRR109, “Discretization in Geometry and Dynamics”.

References

1. Conder, M.: Regular Maps. http://www.math.auckland.ac.nz/~conder/OrientableRegularMaps301.txt (August 2012)
2. Coxeter, H., Moser, W.: Generators and Relations for Discrete Groups Springer (1957)
3. van Wijk, J.J.: Symmetric tiling of closed surfaces: visualization of regular maps. Proc. SIGGRAPH **49**, 1–12 (2009)
4. van Wijk, J.J., Cohen, A.M.: Visualization of seifert surfaces. IEEE Trans. Vis. Comput. Graph. 12 **4**, 485–496 (2006)
5. Razafindrazaka, F.: Visualization of high genus regular maps. Master’s thesis, Freie Universität Berlin (2012)
6. Razafindrazaka, F., Polthier, K.: Regular map smoothing. In: DGCI 2013, Seville (2013)
7. Reid, M., Szendro, B.: Geometry and Topology. Cambridge University Press, Cambridge (2005)
8. Séquin, C.H.: Patterns on the Genus-3 Klein quartic. In: Proceedings of the BRIDGES 2006 Conference, London, pp. 245–254
9. Séquin, C.H.: Symmetric embedding of locally regular hyperbolic tilings. In: Proceedings of the BRIDGES 2007 Conference, San Sebastian, pp. 379–388
10. Séquin, C.H.: My search for symmetrical embeddings of regular maps. In: Proceedings of the BRIDGES 2010 Conference, Hungary, pp. 85–94

Part VI
High-Dimensional Data Analysis: Analysis of Large Systems

Polynomial-Time Amoeba Neighborhood Membership and Faster Localized Solving

Eleanor Anthony, Sheridan Grant, Peter Gritzmann, and J. Maurice Rojas

Dedicated to Tien-Yien Li, in honor of his birthday.

1 Introduction

As students, we are often asked to draw (hopefully without a calculator) real zero sets of low degree polynomials in few variables. As scientists and engineers, we are often asked to count or approximate (hopefully with some computational assistance) real and complex solutions of arbitrary systems of polynomial equations in many variables. If one allows sufficiently coarse approximations, then the latter problem is as easy as the former. Our main results clarify this transition from hardness to easiness. In particular, we significantly speed up certain queries involving

E. Anthony
Mathematics Department, University of Mississippi, Hume Hall 305, P.O. Box 1848, MS 38677-1848, USA
e-mail: ecanthon@go.olemiss.edu
Partially supported by NSF REU grant DMS-1156589.

S. Grant
Mathematics Department, 640 North College Avenue, Claremont, CA 91711, USA
e-mail: sheridan.grant@pomona.edu
Partially supported by NSF REU grant DMS-1156589.

P. Gritzmann
Fakultät für Mathematik, Technische Universität München, 80290 München, Germany
e-mail: gritzmann@tum.de
Work supported in part by the German Research Foundation (DFG).

J.M. Rojas (✉)
Mathematics Department, Texas A&M University, TAMU 3368, College Station, TX 77843-3368, USA
e-mail: rojas@math.tamu.edu
Partially supported by NSF MCS grant DMS-0915245.

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_15

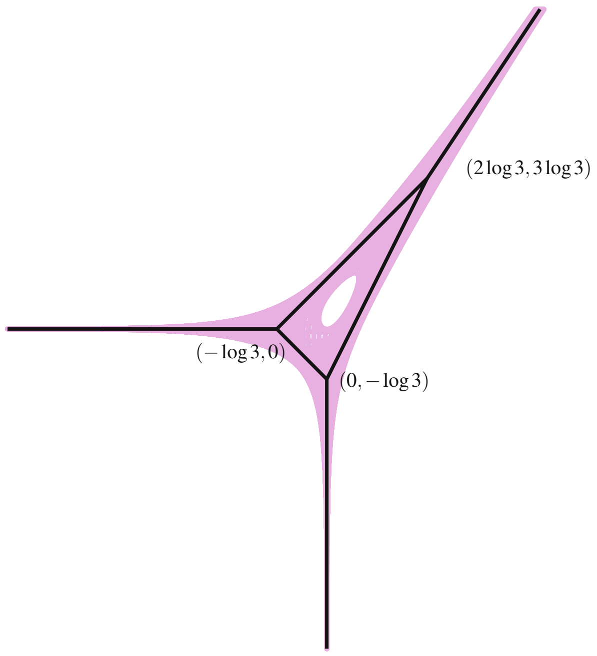
distances between points and complex algebraic hypersurfaces (see Theorems 1.4–1.6 below). We then apply our metric results to finding specially constructed start systems—dramatically speeding up traditional homotopy continuation methods—to approximate, or rule out, roots of selected norm (see Sect. 3).

Polynomial equations are ubiquitous in numerous applications, such as algebraic statistics [29], chemical reaction kinetics [42], discretization of partial differential equations [28], satellite orbit design [47], circuit complexity [36], and cryptography [10]. The need to solve larger and larger equations, in applications as well as for theoretical purposes, has helped shape algebraic geometry and numerical analysis for centuries. More recent work in algebraic complexity tells us that many basic questions involving polynomial equations are **NP**-hard (see, e.g., [13, 52]). This is by no means an excuse to consider polynomial equation solving hopeless: Computational scientists solve problems of near-exponential complexity every day.

Thanks to recent work on *Smale's 17th Problem* [8, 14], we have learned that randomization and approximation can be the key to avoiding the bottlenecks present in deterministic algorithms for solving hard questions involving complex roots of polynomial systems. Smale's 17th Problem concerns the *average-case* complexity of approximating *a single* complex root of a random polynomial system and is well-discussed in [54–58, 60, 61]. Our ultimate goal is to extend this philosophy to the harder problem of *localized solving*: estimating how far the *nearest* root of a given system of polynomials (or intersection of several zero sets) is from a given point. Here, we start by first approximating the shape of a single zero set, and then in Sect. 3 we outline a tropical-geometric approach to localized solving. Toward this end, let us first recall the natural idea (see, e.g., [65]) of drawing zero sets on log-paper. In what follows, we let $\mathbb{C}^*$ denote the non-zero complex numbers and write $\mathbb{C}\left[x_1^{\pm 1}, \ldots, x_n^{\pm 1}\right]$ for the ring of Laurent polynomials with complex coefficients, i.e., polynomials with negative exponents allowed. Also, for any two vectors $u := (u_1, \ldots, u_N)$ and $v := (v_1, \ldots, v_N)$ in $\mathbb{R}^N$, we use $u \cdot v$ to denote the standard dot product $u_1v_1 + \cdots + u_Nv_N$.

Definition 1.1 We set $x := (x_1, \ldots, x_n)$ and $\mathrm{Log}|x| := (\log|x_1|, \ldots, \log|x_n|)$, and, for any $f \in \mathbb{C}\left[x_1^{\pm 1}, \ldots, x_n^{\pm 1}\right]$, we define $\mathrm{Amoeba}(f)$ to be the set $\{\mathrm{Log}|x| \ : \ f(x) = 0\ ,\ x \in (\mathbb{C}^*)^n\}$. We call f an *n-variate t-nomial* when we can write $f(x) = \sum_{i=1}^{t} c_i x^{a_i}$ with $c_i \neq 0$, $a_i := (a_{1,i}, \ldots, a_{n,i})$, the a_i are pair-wise distinct, and $x^{a_i} := x_1^{a_{1,i}} x_2^{a_{2,i}} \cdots x_n^{a_{n,i}}$ for all i. When f is not the zero polynomial, we define the *Archimedean tropical variety of f*, denoted $\mathrm{ArchTrop}(f)$, to be the set of all $w \in \mathbb{R}^n$ for which $\max_i |c_i e^{a_i \cdot w}|$ is attained for at least *two* distinct indices i. Finally, we define $\mathrm{ArchTrop}(0)$ to be $\mathbb{R}^n$. $\diamond$

In Sect. 3 we will see how amoebae and tropical varieties are useful for speeding up polynomial system solving.



Example 1.2 Taking $f(x) = 1 + x_1^3 + x_2^2 - 3x_1x_2$, an illustration of Amoeba(f) and ArchTrop(f), truncated to $[-7, 7]^2$, appears above. Amoeba(f) is lightly shaded, while ArchTrop(f) is the piecewise-linear curve. ⋄

One may be surprised that Amoeba(f) and ArchTrop(f) are so highly structured: Amoeba(f) has tentacles reminiscent of a living amoeba, and ArchTrop(f) is a *polyhedral complex*, i.e., a union of polyhedra intersecting only along common faces (see Definition 2.7 below). One may also be surprised that Amoeba(f) and ArchTrop(f) are so closely related: Every point of one set is close to some point of the other, and both sets have topologically similar complements (4 open connected components, exactly one of which is bounded). Example 2.2 below shows that we need not always have ArchTrop(f) ⊆ Amoeba(f).

To quantify how close Amoeba(f) and ArchTrop(f) are in general, one can recall the *Hausdorff distance*, denoted $\Delta(U, V)$, between two subsets $U, V \subseteq \mathbb{R}^n$: It is defined to be the maximum of $\sup_{u\in U} \inf_{v\in V} |u - v|$ and $\sup_{v\in V} \inf_{u\in U} |u - v|$. We then have the following recent result of Avendaño, Kogan, Nisse, and Rojas.

Theorem 1.3 ([4]) *Suppose f is any n-variate t-nomial. Then* Amoeba(f) *and* ArchTrop(f) *are (a) identical for $t \leq 2$ and (b) at Hausdorff distance no greater than* $(2t - 3)\log(t - 1)$ *for $t \geq 3$. In particular, for $t \geq 2$, we also have*

$$\sup_{u \in \text{Amoeba}(f)} \inf_{v \in \text{ArchTrop}(f)} |u - v| \leq \log(t - 1).$$

Finally, for any $t > n \geq 1$, there is an n-variate t-nomial f with

$$\Delta(\text{Amoeba}(f), \text{ArchTrop}(f)) \geq \log(t-1). \qquad \blacksquare$$

Note that the preceding upper bounds are completely independent of the coefficients, degree, and number of variables of f. Our upcoming examples show that Amoeba(f) and ArchTrop(f) are sometimes much closer than the bound above.

Our first two main results help set the stage for applying Archimedean tropical varieties to speed up polynomial root approximation. Recall that $\mathbb{Q}[\sqrt{-1}]$ denotes those complex numbers whose real and imaginary parts are both rational. Our complexity results will all be stated relative to the classical Turing (bit) model, with the underlying notion of input size clarified below in Definition 1.7.

Theorem 1.4 *Suppose $w \in \mathbb{R}^n$ and $f \in \mathbb{C}\left[x_1^{\pm 1}, \ldots, x_n^{\pm 1}\right]$ is a t-nomial with $t \geq 2$. Then*

$$-\log(t-1) \leq \inf_{u \in \text{Amoeba}(f)} |u-w| - \inf_{v \in \text{ArchTrop}(f)} |v-w| \leq (2t-3)\log(t-1).$$

In particular, if we also assume that n is fixed and $(f, w) \in \mathbb{Q}[\sqrt{-1}]\left[x_1^{\pm 1}, \ldots, x_n^{\pm 1}\right] \times \mathbb{Q}^n$ with f a t-nomial, then we can compute polynomially many bits of $\inf_{v \in \text{ArchTrop}(f)} |v-w|$ in polynomial-time, and there is a polynomial-time algorithm that declares either (a) $\inf_{u \in \text{Amoeba}(f)} |u-w| \leq (2t-2)\log(t-1)$ or (b) $w \notin \text{Amoeba}(f)$ and $\inf_{u \in \text{Amoeba}(f)} |u-w| \geq \inf_{v \in \text{ArchTrop}(f)} |v-w| - \log(t-1) > 0$.

Theorem 1.4 is proved in Sect. 5. The importance of Theorem 1.4 is that deciding whether an input rational point w lies in an input Amoeba(f), *even restricting to the special case $n = 1$*, is already **NP**-hard [4].

ArchTrop(f) naturally partitions $\mathbb{R}^n$ into finitely many (relatively open) polyhedral cells of dimension 0 through n. We call the resulting polyhedral complex $\Sigma(\text{ArchTrop}(f))$ (see Definition 2.7 below). In particular, finding the cell of $\Sigma(\text{ArchTrop}(f))$ containing a given $w \in \mathbb{R}^n$ gives us more information than simply deciding whether w lies in ArchTrop(f).

Theorem 1.5 *Suppose n is fixed. Then there is a polynomial-time algorithm that, for any input $(f, w) \in \mathbb{Q}\left[\sqrt{-1}\right]\left[x_1^{\pm 1}, \ldots, x_n^{\pm 1}\right] \times \mathbb{Q}^n$ with f a t-nomial, outputs the closure of the unique cell σ_w of $\Sigma(\text{ArchTrop}(f))$ containing w, described as an explicit intersection of $O(t^2)$ half-spaces.*

Theorem 1.5 is proved in Sect. 4. As a consequence, we can also find explicit regions, containing a given query point w, where f can not vanish. Let d denote the degree of f. While our present algorithm evincing Theorem 1.5 has complexity exponential in n, its complexity is polynomial in $\log d$ (see Definition 1.7 below). The best previous techniques from computational algebra, including recent advances on Smale's 17th Problem [8, 14], yield complexity no better than polynomial in $\frac{(d+n)!}{d!n!} \geq \max\left\{\left(\frac{d+n}{d}\right)^d, \left(\frac{d+n}{n}\right)^n\right\}$.

Our framework also enables new positive and negative results on the complexity of approximating the intersection of several Archimedean tropical varieties.

Theorem 1.6 *Suppose n is fixed. Then there is a polynomial-time algorithm that, for any input k and $(f_1, \ldots, f_k, w) \in \left(\mathbb{Q}[\sqrt{-1}][x_1^{\pm 1}, \ldots, x_n^{\pm 1}]\right)^k \times \mathbb{Q}^n$, outputs the closure of the unique cell σ_w of $\Sigma\left(\bigcup_{i=1}^k \mathrm{ArchTrop}(f_i)\right)$ containing w, described as an explicit intersection of half-spaces. (In particular, whether w lies in $\bigcap_{i=1}^k \mathrm{ArchTrop}(f_i)$ is decided as well.) However, if n is allowed to vary, then deciding whether σ_w has a vertex in $\bigcap\limits_{i=1}^{n} \mathrm{ArchTrop}(f_i)$ is* **NP***-hard.*

Theorem 1.6 is proved in Sect. 6. We will see in Sect. 3 how the first assertion of Theorem 1.6 is useful for finding special start-points for *Newton Iteration* and *Homotopy Continuation* that sometimes enable the approximation of just the roots with norm vector near $(e^{w_1}, \ldots, e^{w_n})$. The final assertion of Theorem 1.6 can be considered as a refined tropical analogue to a classical algebraic complexity result: Deciding whether an arbitrary input system of polynomials equations (with integer coefficients) has a complex root is **NP**-hard. (There are standard reductions from known **NP**-complete problems, such as integer programming or Boolean satisfiability, to complex root detection [21, 52].)

On the practical side, we point out that the algorithms underlying Theorems 1.4–1.6 are quite easily implementable. (A preliminary `Matlab` implementation of our algorithms is available upon request.) Initial experiments indicate that a large-scale implementation could be a worthwhile companion to existing polynomial system solving software.

Before moving on to the necessary technical background, let us first clarify our underlying input size and point out some historical context.

Definition 1.7 We define the *input size* of an integer c to be $\mathrm{size}(c) := \log(2 + |c|)$ and, for $p, q \in \mathbb{Z}$ relative prime with $|q| \geq 2$, $\mathrm{size}(p/q) := \mathrm{size}(p) + \mathrm{size}(q)$. Given a polynomial $f \in \mathbb{Q}[x_1, \ldots, x_n]$, written $f(x) = \sum_{i=1}^t c_i x^{a_i}$, we then define $\mathrm{size}(f)$ to be $\sum_{i=1}^t \left(\mathrm{size}(c_i) + \sum_{j=1}^n \mathrm{size}(a_{i,j})\right)$, where $a_i = (a_{i,1}, \ldots, a_{i,n})$ for all i. Similarly, we define the *input size of a point* $(v_1, \ldots, v_n) \in \mathbb{Q}^n$ as $\sum_{i=1}^n \mathrm{size}(v_i)$. Considering real and imaginary parts, and summing the respect sizes, we then extend the definition of input size further still to polynomials in $\mathbb{Q}\left[\sqrt{-1}\right][x_1, \ldots, x_n]$. Finally, for any system of polynomials $F := (f_1, \ldots, f_k)$, we set $\mathrm{size}(F) := \sum_{i=1}^k \mathrm{size}(f_i)$. $\diamond$

Note in particular that the size of an input in Theorem 1.6 is $\mathrm{size}(w) + \sum_{i=1}^k \mathrm{size}(f_i)$.

Remark 1.8 The reader may wonder why we have not considered the *phases* of the root coordinates and focussed just on norms. The phase analogue of an amoeba is the *co-amoeba*, which has only recently been studied [30, 46, 48]. While it is known that the phases of the coordinates of the roots of polynomial systems satisfy certain equidistribution laws (see, e.g., [35, Thm. 1 (pp. 82–83), Thm. 2 (pp. 87–88), and Cor. 3′ (p. 88)] and [2]), there does not yet appear to be a phase analogue of ArchTrop(f). Nevertheless, we will see in Sect. 3 that our techniques sometimes allow us to approximate not just norms of root coordinates but roots in full. ◇

Historical Notes *Using convex and/or piecewise-linear geometry to understand solutions of algebraic equations can be traced back to work of Newton (on power series expansions for algebraic functions) around 1676 [44].*

More recently, tropical geometry [6, 17, 32, 38, 39] has emerged as a rich framework for reducing deep questions in algebraic geometry to more tractable questions in polyhedral and piecewise-linear geometry. For instance, Gelfand, Kapranov, and Zelevinsky first observed the combinatorial structure of amoebae around 1994 [22]. ◇

2 Background

2.1 Convex, Piecewise-Linear, and Tropical Geometric Notions

Let us first recall the origin of the phrase "tropical geometry", according to [51]: the *tropical semifield* $\mathbb{R}_{\text{trop}}$ is the set $\mathbb{R} \cup \{-\infty\}$, endowed with the operations $x \odot y := x + y$ and $x \oplus y := \max\{x, y\}$. The adjective "tropical" was coined by French computer scientists, in honor of Brazilian computer scientist Imre Simon, who did pioneering work with algebraic structures involving $\mathbb{R}_{\text{trop}}$. Just as algebraic geometry relates geometric properties of zero sets of polynomials to the structure of ideals in commutative rings, tropical geometry relates the geometric properties of certain polyhedral complexes (see Definition 2.7 below) to the structure of ideals in $\mathbb{R}_{\text{trop}}$.

Here we work with a particular kind of tropical variety that, thanks to Theorem 1.3, approximates Amoeba(f) quite well. The binomial case is quite instructive.

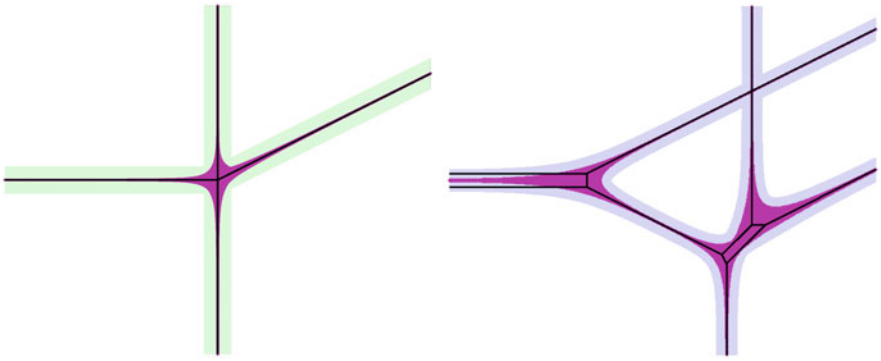
Proposition 2.1 *For any $a \in \mathbb{Z}^n$ and non-zero complex c_1 and c_2, we have*

$$\text{Amoeba}(c_1 + c_2x^a) = \text{ArchTrop}(c_1 + c_2x^a) = \{w \in \mathbb{R}^n \mid a \cdot w = \log|c_1/c_2|\}.$$

Proof If $c_1 + c_2x^a = 0$ then $|c_2x^a| = |c_1|$. We then obtain $a \cdot w = \log|c_1/c_2|$ upon taking logs and setting $w = \text{Log}|x|$. Conversely, for any w satisfying $a \cdot w = \log|c_1/c_2|$, note that $x = e^{w+\theta\sqrt{-1}}$, with $a \cdot \theta$ the imaginary part of $-c_1/c_2$, satisfies $c_1 + c_2x^a = 0$. This proves that Amoeba($c_1 + c_2x^a$) is exactly the stated affine hyperplane. Similarly, since the definition of ArchTrop($c_1 + c_2x^a$) implies that we seek w with $|c_2 e^{a\cdot w}| = |c_1|$, we see that $\mathrm{ArchTrop}(c_1 + c_2 x^a)$ defines the same hyperplane. ■

While $\mathrm{ArchTrop}(f)$ and $\mathrm{Amoeba}(f)$ are always metrically close, $\mathrm{ArchTrop}(f)$ need not even have the same homotopy type as $\mathrm{Amoeba}(f)$ in general.

Example 2.2



Letting $f := 1 + x_2^2 + x_2^4 + x_1x_2^2 + x_1x_2^4 + x_1^2x_2 + x_1^2x_2^2 + x_1^3$ and $g := 0.1 + 0.2x_2^2 + 0.1x_2^4 + 10x_1x_2^2 + 0.001x_1x_2^4 + 0.01x_1^2x_2 + 0.1x_1^2x_2^2 + 0.000005x_1^3$ we obtain the amoebae and tropical varieties (and more lightly shaded neighborhoods), restricted to $[-11, 11] \times [-9, 9]$, respectively drawn on the left and right above. The outermost shape in the left-hand (resp. right-hand) illustration is a neighborhood of $\mathrm{ArchTrop}(f)$ (resp. $\mathrm{Amoeba}(g)$).

It turns out that every point of $\mathrm{Amoeba}(f)$ (resp. $\mathrm{ArchTrop}(g)$) lies well within a distance of 0.65 (resp. 0.49) of some point of $\mathrm{ArchTrop}(f)$ (resp. $\mathrm{Amoeba}(g)$), safely within the distance $\log 7 < 1.946$ (resp. $13 \log 7 < 25.3$) guaranteed by the second (resp. first) bound of Theorem 1.3. Note also that $\mathrm{ArchTrop}(g)$ has two holes while $\mathrm{Amoeba}(g)$ has only a single hole.[1] $\diamond$

Given any f one can naturally construct a convergent sequence of polynomials whose amoebae tend to $\mathrm{ArchTrop}(f)$. This fact can be found in earlier papers of Viro and Mikhalkin, e.g., [41, 65]. However, employing Theorem 1.3 here, we can give a 5-line proof.

Theorem 2.3 *For any n-variate t-nomial f written $\sum_{i=1}^{t} c_i x^{a_i}$, and $s > 0$, define $f^{*s}(x) := \sum_{i=1}^{t} c_i^s x^{a_i}$. Then $\Delta\left(\frac{1}{s}\mathrm{Amoeba}(f^{*s}), \mathrm{ArchTrop}(f)\right) \to 0$ as $s \to +\infty$.*

Proof By Theorem 1.3, $\Delta(\mathrm{Amoeba}(f^{*s}), \mathrm{ArchTrop}(f^{*s})) \leq (2t-3)\log(t-1)$ for all $s > 0$. Since $|c_i e^{a_i\cdot w}| \geq |c_j e^{a_j\cdot w}| \iff |c_i e^{a_i\cdot w}|^s \geq |c_j e^{a_j\cdot w}|^s$, we immediately obtain that $\mathrm{ArchTrop}(f^{*s}) = s\,\mathrm{ArchTrop}(f)$. So then

[1] A *hole* of a subset $S \subseteq \mathbb{R}^n$ is simply a bounded connected component of the complement $\mathbb{R}^n \setminus S$.

$\Delta(\text{Amoeba}(f^{*s}), \text{ArchTrop}(f^{*s})) = s\Delta\big(\frac{1}{s}\text{Amoeba}(f^{*s}), \text{ArchTrop}(f)\big)$ and thus $\Delta\big(\frac{1}{s}\text{Amoeba}(f^{*s}), \text{ArchTrop}(f)\big) \leq \frac{(2t-3)\log(t-1)}{s}$ for all $s > 0$. ■

To more easily link ArchTrop(f) with polyhedral geometry we will need two variations of the classical Newton polygon. First, let Conv(S) denote the convex hull of[2] $S \subseteq \mathbb{R}^n$, $\mathbf{O} := (0, \ldots, 0)$, and $[N] := \{1, \ldots, N\}$. Recall that a *polytope* is the convex hull of a finite point set, a *(closed) half-space* is any set of the form $\{w \in \mathbb{R}^n \mid a \cdot w \leq b\}$ (for some $b \in \mathbb{R}$ and $a \in \mathbb{R}^n \setminus \{\mathbf{O}\}$), and a *(closed) polyhedron* is any intersection of finitely many (closed) half-spaces. Polytopes are exactly polyhedra that are bounded [27, 66]. The two resulting representations of polytopes—$\mathscr{V}$*-presentation* (the convex hull of a finite point set) and $\mathscr{H}$*-presentation* (an intersection of finitely many half-spaces)—are equivalent, but can be exponentially different from an algorithmic point of view. See, e.g., [24, 25].

Definition 2.4 Given any n-variate t-nomial f written $\sum_{i=1}^{t} c_i x^{a_i}$, we define its *(ordinary) Newton polytope* to be $\text{Newt}(f) := \text{Conv}\big(\{a_i\}_{i\in[t]}\big)$, and the *Archimedean Newton polytope of* f to be $\text{ArchNewt}(f) := \text{Conv}\big(\{(a_i, -\log|c_i|)\}_{i\in[t]}\big)$. Also, for any polyhedron $P \subset \mathbb{R}^N$ and $v \in \mathbb{R}^N$, a *face* of P is any set of the form $P_v := \{x \in P \mid v \cdot x \text{ is maximized}\}$. We call v an *outer normal* of P_v. The *dimension* of P, written $\dim P$, is simply the dimension of the smallest affine linear subspace containing P. Faces of P of dimension 0, 1, and $\dim P - 1$ are respectively called *vertices*, *edges*, and *facets*. (P and $\emptyset$ are called *improper faces* of P, and we set $\dim \emptyset := -1$.) Finally, we call any face of P *lower* if and only if it has an outer normal $(w_1, \ldots, w_N)$ with $w_N < 0$, and we let the *lower hull* of ArchNewt(f) be the union of the lower faces of ArchNewt(f). ◇

The outer normals of a k-dimensional face of an n-dimensional polyhedron P form the relative interior of an $(n-k)$-dimensional polyhedron called an *outer normal cone*. Note that ArchNewt(f) usually has dimension 1 greater than that of Newt(f). ArchNewt(f) enables us to relate ArchTrop(f) to linear optimization.

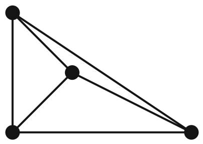
Proposition 2.5 *For any n-variate t-nomial f,* ArchTrop(f) *can also be defined as the set of all $w \in \mathbb{R}^n$ with* $\max\limits_{x\in\text{ArchNewt}(f)} \{x \cdot (w, -1)\}$ *attained on a positive-dimensional face of* ArchNewt(f).

Proof The quantity $|c_i e^{a_i \cdot w}|$ attaining its maximum for at least two indices i is equivalent to the linear form with coefficients $(w, -1)$ attaining its maximimum for at least two different points in $\{(a_i, -\log|c_i|)\}_{i\in[t]}$. Since a face of a polytope is positive-dimensional if and only if it has at least two vertices, we are done. ■

Example 2.6 The Newton polytope of our first example, $f = 1 + x_1^3 + x_2^2 - 3x_1x_2$, is simply the convex hull of the exponent vectors of the monomial terms: $\text{Conv}(\{(0,0), (3,0), (0,2), (1,1)\})$. For the Archimedean Newton polytope, we take the coefficients into account via an extra coordinate:

[2]That is, smallest convex set containing…

$\text{ArchNewt}(f) = \text{Conv}(\{(0,0,0),(3,0,0),(0,2,0),\ (1,1,-\log 3)\})$. In particular, $\text{Newt}(f)$ is a triangle and $\text{ArchNewt}(f)$ is a triangular pyramid with base $\text{Newt}(f) \times \{0\}$ and apex lying beneath $\text{Newt}(f) \times \{0\}$. Note also that the image of the orthogonal projection of the lower hull of $\text{ArchNewt}(f)$ onto $\mathbb{R}^2 \times \{0\}$ naturally induces a triangulation of $\text{Newt}(f)$, as illustrated below. $\diamond$



Our last example motivates us to consider more general subdivisions and duality. (An outstanding reference is [15].) Recall that a *k-simplex* is the convex hull of $k+1$ points in $\mathbb{R}^N$ (with $N \geq k+1$) *not* lying in any $(k-1)$-dimensional affine linear subspace of $\mathbb{R}^N$. A *simplex* is then simply a k-simplex for some k.

Definition 2.7 A *polyhedral complex* is a collection of polyhedra $\Sigma = \{\sigma_i\}_i$ such that for all i we have (a) every face of σ_i is in Σ and (b) for all j we have that $\sigma_i \cap \sigma_j$ is a face of both σ_i and σ_j. (We allow improper faces like $\emptyset$, σ_i, and σ_j.) The σ_i are the *cells* of the complex, and the *underlying space of* Σ is $|\Sigma| := \bigcup_i \sigma_i$. In particular, we define $\Sigma(\text{ArchTrop}(f))$ to be the complex whose cells are exactly the (possibly improper) faces of the closures of the connected components of $\mathbb{R}^n \setminus \text{ArchTrop}(f)$.

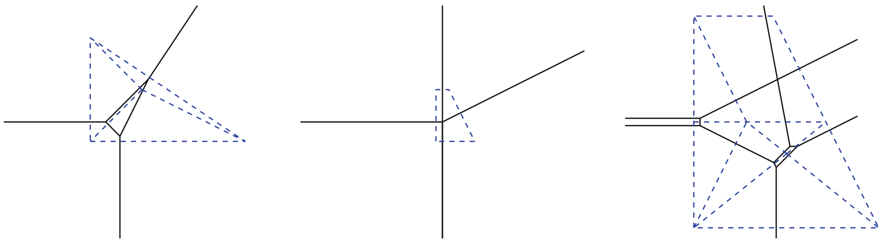
A *polyhedral subdivision* of a polyhedron P is then simply a polyhedral complex $\Sigma = \{\sigma_i\}_i$ with $|\Sigma| = P$. We call Σ a *triangulation* if and only if every σ_i is a simplex. Given any finite subset $A \subset \mathbb{R}^n$, a *polyhedral subdivision induced by* A is then just a polyhedral subdivision of $\text{Conv}(A)$ where the vertices of all the σ_i lie in A. Finally, the *polyhedral subdivision of* $\text{Newt}(f)$ *induced by* $\text{ArchNewt}(f)$, denoted Σ_f, is simply the polyhedral subdivision whose cells are $\{\pi(Q) \mid Q \text{ is a lower face of ArchNewt}(f)\}$, where $\pi : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n$ denotes the orthogonal projection forgetting the last coordinate. $\diamond$

Recall that a *(polyhedral) cone* is just the set of all nonnegative linear combinations of a finite set of points. Such cones are easily seen to always be polyhedra [27, 66].

Example 2.8 The illustration from Example 2.6 above shows a triangulation of the point set $\{(0,0),(3,0),(0,2),(1,1)\}$ which happens to be Σ_f for $f = 1 + x_1^3 + x_2^2 - 3x_1x_2$. More to the point, it is easily checked that the outer normals to a face of dimension k of $\text{ArchNewt}(f)$ form a cone of dimension $3-k$. In this way, thanks to the natural partial ordering of cells in any polyhedral complex by inclusion, we get an order-reversing bijection between the cells of Σ_f and pieces of $\text{ArchTrop}(f)$. $\diamond$

That ArchTrop(f) is always a polyhedral complex follows directly from Proposition 2.5 above. Proposition 2.5 also implies an order-reversing bijection between the cells Σ_f and the cells of $\Sigma(\text{ArchTrop}(f))$—an incarnation of *polyhedral duality* [66].

Example 2.9 Below we illustrate the aforementioned order-reversing bijection of cells through our first three tropical varieties, and corresponding subdivisions Σ_f of Newt(f):



Note that the vertices of $\Sigma(\text{ArchTrop}(f))$ correspond bijectively to the two-dimensional cells of Σ_f, and the one-dimensional cells of $\Sigma(\text{ArchTrop}(f))$ correspond bijectively to the edges of Σ_f. (In particular, the rays of Σ(ArchTrop (f)) are perpendicular to the edges of Newt(f).) Note also that the vertices of Σ_f correspond bijectively to connected components of the *complement* $\mathbb{R}^2\backslash\text{ArchTrop}(f)$. $\diamond$

2.2 *The Complexity of Linear Programming*

Let us first point out that [3, 21, 49, 59] are excellent references for further background on the classical Turing model and **NP**-completeness. The results on the complexity of linear optimization we'll use are covered at a more leisurely pace in standard monographs such as [26, 53]. See also [23].

Definition 2.10 Given any matrix $M \in \mathbb{Q}^{k\times N}$ with $i^{\underline{\text{th}}}$ row m_i, and $b := (b_1,\ldots,b_k)^\top \in \mathbb{Q}^k$, the notation $Mx \leq b$ means that $m_1\cdot x \leq b_1,\ldots,m_k\cdot x\leq b_k$ all hold. Given any $c=(c_1,\ldots,c_N)\in\mathbb{Q}^N$ we then define the *(natural form) linear optimization problem* $\mathscr{L}(M,b,c)$ to be the following: Maximize $c\cdot x$ subject to $Mx\leq b$ and $x\in\mathbb{R}^N$. We also define size$(\mathscr{L}(M,b,c))$:= size(M) + size(b) + size(c) (see Definition 1.7). The set of all $x\in\mathbb{R}^N$ satisfying $Mx\leq b$ is the *feasible region* of $\mathscr{L}(M,b,c)$, and when it is empty we call $\mathscr{L}(M,b,c)$ *infeasible*. Finally, if $\mathscr{L}(M,b,c)$ is feasible but does not admit a well-defined maximum, then we call $\mathscr{L}(M,b,c)$ *unbounded*. $\diamond$

Theorem 2.11 *Given any linear optimization problem $\mathscr{L}(M,b,c)$ as defined above, we can decide infeasibility, unboundedness, or (if $\mathscr{L}(M,b,c)$ is feasible, with bounded maximum) find an optimal solution x^*, all within time polynomial in* size($\mathscr{L}(M,b,c)$). *In particular, if $\mathscr{L}(M,b,c)$ is feasible, with bounded maximum, then we can find an optimal solution x^* of size polynomial in* size($\mathscr{L}(M,b,c)$). ■

Theorem 2.11 goes back to work of Khachiyan in the late 1970s on the *Ellipsoid Method* [34], building upon earlier work of Shor, Yudin, and Nemirovskii.

For simplicity, we will not focus on the best current complexity bounds, since our immediate goal is to efficiently prove polynomiality for our algorithms. We will need one last complexity result from linear optimization: Recall that a constraint $m_i \cdot x \leq b_i$ of $Mx \leq b$ is called *redundant* if and only if the corresponding row of M, and corresponding entry of b, can be deleted from the pair (M,b) without affecting the feasible region $\{x \in \mathbb{R}^N \mid Mx \leq b\}$.

Lemma 2.12 *Given any system of linear inequalities $Mx \leq b$ we can, in time polynomial in* size(M) + size(b)*, find a submatrix M' of M, and a subvector b' of b, such that $\{x \in \mathbb{R}^N \mid M'x \leq b'\} = \{x \in \mathbb{R}^N \mid M'x \leq b'\}$ and $M'x \leq b'$ has no redundant constraints.* ■

The new set of inequalities $M'x \leq b'$ is called an *irredundant representation* of $Mx \leq b$, and can easily be found by solving $\leq k$ linear optimization problems of size no larger than size($\mathscr{L}(M,b,\mathbf{O})$) (see, e.g., [53]).

The linear optimization problems we ultimately solve will have *irrational "right-hand sides"*: Our b will usually have entries that are (rational) linear combination of logarithms of integers. As is well-known in Diophantine Approximation [5], it is far from trivial to efficiently decide the sign of such irrational numbers. This problem is equivalent to deciding inequalities of the form $\alpha_1^{\beta_1} \cdots \alpha_N^{\beta_N} > 1$, where the α_i and β_i are integers. Note, in particular, that while the number of arithmetic operations necessary to decide such an inequality is easily seen to be $O((\sum_{i=1}^N \log|\beta_i|)^2)$ (via the classical binary method of exponentiation), taking bit-operations into account naively results in a problem that appears to have complexity *exponential* in $\log|\beta_1| + \cdots + \log|\beta_N|$. But we can in fact go much faster...

2.3 Irrational Linear Optimization and Approximating Logarithms

Recall the following result on comparing monomials in rational numbers.

Theorem 2.13 ([11, Sec. 2.4]) *Suppose $\alpha_1,\ldots,\alpha_N \in \mathbb{Q}$ are positive and $\beta_1,\ldots,\beta_N \in \mathbb{Z}$. Also let A be the maximum of the numerators and denominators of the α_i (when written in lowest terms) and $B := \max_i\{|\beta_i|\}$. Then, within*

$$O\big(N30^N \log(B)(\log\log B)^2 \log\log\log(B)(\log(A)(\log\log A)^2 \log\log\log A)^N\big)$$

bit operations, we can determine the sign of $\alpha_1^{\beta_1} \cdots \alpha_N^{\beta_N} - 1$. ■

While the underlying algorithm is a simple application of Arithmetic-Geometric Mean Iteration (see, e.g., [9]), its complexity bound hinges on a deep estimate of Nesterenko [43], which in turn refines seminal work of Matveev [40] and Alan Baker [5] on linear forms in logarithms.

Definition 2.14 We call a polyhedron P *ℓ-rational* if and only if it is of the form $\{x \in \mathbb{R}^n \mid Mx \leq b\}$ with $M \in \mathbb{Q}^{k\times n}$ and $b = (b_1, \ldots, b_k)^\top$ satisfying

$$b_i = \beta_{1,i} \log|\alpha_1| + \cdots + \beta_{k,i} \log|\alpha_k|,$$

with $\beta_{i,j}, \alpha_j \in \mathbb{Q}$ for all i and j. Finally, we set

$$\text{size}(P) := \text{size}(M) + \text{size}([\beta_{i,j}]) + \sum_{i=1}^{k} \text{size}(\alpha_i). \diamond$$

Via the Simplex Method (or even a brute force search through all n-tuples of facets of P) we can obtain the following consequence of Theorems 2.11 and 2.13.

Corollary 2.15 *Following the notation of Definition 2.14, suppose n is fixed. Then we can decide whether P is empty, compute an irredundant representation for P, and enumerate all maximal sets of facets determining vertices of P, in time polynomial in* size(P). ■

The key trick behind the proof of Corollary 2.15 is that the intermediate linear optimization problems needed to find an irredundant representation for P use linear combinations (of rows of the original representation) with coefficients of moderate size (see, e.g., [53]).

3 Tropical Start-Points for Numerical Iteration and an Example

We begin by outlining a method for picking start-points for Newton Iteration (see, e.g., [12, Ch. 8] for a modern perspective) and Homotopy Continuation [7, 31, 37, 62, 64]. While we do not discuss these methods for solving polynomial equations in further detail, let us at least point out that Homotopy Continuation (combined with Smale's α-Theory for certifying roots [7,12]) is currently the fastest, most easily parallelizable, and reliable method for numerically solving polynomial systems in complete generality. Other important methods include Resultants [18] and Gröbner Bases [20]. While these alternative methods are of great utility in certain algebraic and theoretical applications [1, 19], Homotopy Continuation is currently the method of choice for practical numerical computation with extremely large polynomial systems.

Algorithm 3.1 (Coarse Approximation to Roots with Log-Norm Vector Near a Given Query Point)

INPUT. *Polynomials* $f_1, \ldots, f_n \in \mathbb{Q}\left[\sqrt{-1}\right]\left[x_1^{\pm 1}, \ldots, x_n^{\pm 1}\right]$, *with* $f_i(x) = \sum_{j=1}^{t_i} c_{i,j} x^{a_j(i)}$ *an* n*-variate* t_i*-nomial for all* i, *and a query point* $w \in \mathbb{Q}^n$.

OUTPUT. *An ordered* n*-tuple of sets of indices* $(J_i)_{i=1}^n$ *such that, for all* i, $g_i := \sum_{j \in J_i} c_{i,j} x^{a_j(i)}$ *is a sub-summand of* f_i, *and the roots of* $G := (g_1, \ldots, g_n)$ *are approximations of the roots of* $F := (f_1, \ldots, f_n)$ *with log-norm vector nearest* w.

DESCRIPTION.

1. *Let* σ_w *be the closure of the unique cell of* $\Sigma(\bigcup_{i=1}^n \mathrm{ArchTrop}(f_i))$ *(see Definition 2.7) containing* w.
2. *If* σ_w *has no vertices in* $\bigcap_{i=1}^n \mathrm{ArchTrop}(f_i)$ *then output an irredundant collection of facet inequalities for* σ_w*, output* "`There are no roots of F in` σ_w `.`"*, and* STOP.
3. *Otherwise, fix any vertex* v *of* $\sigma_w \cap \bigcap_{i=1}^n \mathrm{ArchTrop}(f_i)$ *and, for each* $i \in [n]$, *let* E_i *be any edge of* $\mathrm{ArchNewt}(f_i)$ *generating a facet of* $\mathrm{ArchTrop}(f_i)$ *containing* v.
4. *For all* $i \in [n]$, *let* $J_i := \{j \mid (a_j(i), -\log|c_{i,j}|) \in E_i\}$.
5. *Output* $(J_i)_{i=1}^n$. ■

Thanks to our main results and our preceding observations on linear optimization, we can easily obtain that our preceding algorithm has complexity polynomial in size(F) for fixed n. In particular, Step 1 is (resp. Steps 2 and 3 are) accomplished via the algorithm underlying Theorem 1.5 (resp. Corollary 2.15).

The key subtlety then is to prove that, for most inputs, our algorithm actually gives useful approximations to the roots with log-norm vector nearest the input query point w, or truthfully states that there are no root log-norm vectors in σ_w. We leave the precise metric estimates defining "most inputs" for future work. However, we point out that a key ingredient is the $\mathscr{A}$-discriminant [22], and a recent polyhedral approximation of its amoeba [50] refining the *tropical discriminant* [16]. So we will now clarify the meaning of the output of our algorithm.

The output system G is useful because, with high probability (in the sense of *random liftings*, as in [18, Lemma 6.2]), all the g_i are *binomials*, and binomial systems are particularly easy to solve: They are equivalent to linear equations in the logarithms of the original variables. In particular, any $n \times n$ binomial system always has a *unique* vector of norms for its roots.

Recall the standard notation $\mathrm{Jac}(F) := \left[\frac{\partial f_i}{\partial x_j}\right]_{n \times n}$. The connection to Newton Iteration is then easy to state: Use any root of G as a start-point $z(0)$ for the iteration $z(n+1) := z(n) - \mathrm{Jac}(F)^{-1}|_{z(n)} F(z(n))$. The connection to Homotopy Continuation is also simple: Use the pair (G, ζ) (for any root ζ of G) to start a path converging (under the usual numerical conditioning assumptions on whatever predictor-corrector method one is using) to a root of F with log-norm vector near w. Note also that while it is safer to do the extra work of Homotopy Continuation, there will be cases where the tropical start-points from Algorithm 3.1 are sufficiently good for mere Newton Iteration to converge quickly to a true root.

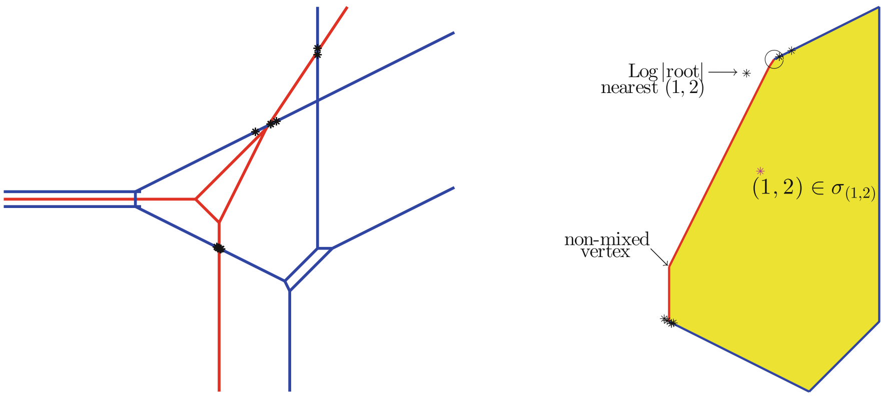
Remark 3.2 Note that, when applying Algorithm 3.1 for later Homotopy Continuation, we have the freedom to follow as few start-points, or as few paths, as we want. When our start-points (resp. paths) indeed converge to nearby roots, we obtain a tremendous savings over having to follow *all* start-points (resp. paths). ⋄

Definition 3.3 Given any n-dimensional polyhedra $P_1, \ldots, P_n \subset \mathbb{R}^n$, we call a vertex v of $\bigcap_{i=1}^n P_i$ *mixed* if and only if v lies on a facet of P_i for all i. ⋄

Note that, by construction, any vertex chosen in Step 3 of Algorithm 3.1 is mixed.

Example 3.4 Let us make a 2×2 polynomial system out of our first and third examples:

$$\begin{aligned} f_1 &:= 1 + x_1^3 + x_2^2 - 3x_1x_2 \\ f_2 &:= 0.1 + 0.2x_2^2 + 0.1x_2^4 + 10x_1x_2^2 + 0.001x_1x_2^4 \\ &\quad +0.01x_1^2x_2 + 0.1x_1^2x_2^2 + 0.000005x_1^3 \end{aligned}$$



The system $F := (f_1, f_2)$ has exactly 12 roots in $(\mathbb{C}^*)^2$, the coordinate-wise log-norms of which form the small clusters near certain intersections of ArchTrop(f_1) and ArchTrop(f_2) shown on the left illustration above. In particular, $\sigma_{(2,1)}$ is the heptagonal cell[3] magnified on the right of the figure above, and has exactly 2 vertices that are mixed. (The other 5 vertices of $\sigma_{(1,2)}$ are vertices of ArchTrop(f_i) lying in the interior of a two-dimensional cell of Σ(ArchTrop(f_{3-i})) for $i \in \{1, 2\}$.)

Applying Algorithm 3.1 we then have two possible outputs, depending on which mixed vertex of $\sigma_{(1,2)}$ we pick. The output corresponding to the circled vertex is the pair of index sets $(\{2, 3\}, \{3, 4\})$. More concretely, Algorithm 3.1 alleges that

[3]The cell looks hexagonal because it has a pair of vertices too close to distinguish visually.

the system $G := (g_1, g_2) := (x_1^3 + x_2^2, 0.1x_2^4 + 10x_1x_2^2)$ has roots with log-norm vector near a log-norm vector of a root of F that is in turn close to w. Indeed, the sole log-norm vector coming from the roots of G is $\left(\log 10, \frac{3}{2}\log 10\right)$ and the roots themselves are $\{(\pm 10, \sqrt{\mp 1000})\}$ (with both values of the square root allowed). All 4 roots in fact converge (under Newton iteration, with no need for Homotopy Continuation) to true roots of F. In particular, the root $(-10, \sqrt{1000})$ (resp. $(-10, -\sqrt{1000})$) converges to the root of F with closest (resp. third closest) log-norm vector to w. The other two roots of G converge to a conjugate pair of roots of F with log-norm vector $(2.4139, 3.5103)$ (to four decimal places) lying in the small circle in the illustration. $\diamond$

Remark 3.5 While we have relied upon Diophantine approximation and subtle aspects of the Simplex Method to prove our bit-complexity bounds in Theorems 1.4–1.6, one can certainly be more flexible when using Algorithm 3.1 in practical floating-point computations. For instance, heuristically, it appears that one can get away with less accuracy than stipulated by Theorem 2.13 when comparing linear combinations of logarithms. Similarly, one should feel free to use the fastest (but still reliably accurate) algorithms for linear optimization when applying our methods to large-scale polynomial systems. (See, e.g., [63].) $\diamond$

4 Proof of Theorem 1.5

Using $t-1$ comparisons, we can isolate all indices i such that $\max_i |c_i e^{a_i \cdot w}|$ is attained. Thanks to Theorem 2.13 this can be done in polynomial-time. We then obtain, say, J equations of the form $a_i \cdot w = -\log|c_i|$ and K inequalities of the form $a_i \cdot w > -\log|c_i|$ or $a_i \cdot w < -\log|c_i|$.

Thanks to Lemma 2.12, combined with Corollary 2.15, we can determine the exact cell of ArchTrop(f) containing w if $J \geq 2$. Otherwise, we obtain the unique cell of $\mathbb{R}^n \setminus$ ArchTrop(f) with relative interior containing w. Note also that an $(n-1)$-dimensional face of either kind of cell must be the dual of an edge of ArchNewt(f). Since every edge has exactly 2 vertices, there are at most $t(t-1)/2$ such $(n-1)$-dimensional faces, and thus σ_w is the intersection of at most $t(t-1)/2$ half-spaces. So we are done. $\blacksquare$

Remark 4.1 Theorem 1.5 also generalizes an earlier complexity bound from [4] for deciding membership in ArchTrop(f). $\diamond$

5 Proof of Theorem 1.4

Note *The hurried reader will more quickly grasp the following proof after briefly reviewing Theorems 1.3, 1.5, and 2.13.*

Since ArchTrop(f) and Amoeba(f) are closed and non-empty, $\inf_{v\in\text{ArchTrop}(f)}|v-w|=|w-v'|$ for some point $v'\in\text{ArchTrop}(f)$ and $\inf_{u\in\text{Amoeba}(f)}|u-w|=|w-u'|$ for some point $u'\in\text{Amoeba}(f)$.

Now, by the second upper bound of Theorem 1.3, there is a point $v''\in$ ArchTrop(f) within distance $\log(t-1)$ of u'. Clearly, $|w-v'|\leq|w-v''|$. Also, by the Triangle Inequality, $|w-v''|\leq|w-u'|+|u'-v''|$. So then,

$$\inf_{v\in\text{ArchTrop}(f)}|v-w|\leq\inf_{u\in\text{Amoeba}(f)}|u-w|+\log(t-1),$$

and thus $\inf_{u\in\text{Amoeba}(f)}|u-w|-\inf_{v\in\text{ArchTrop}(f)}|v-w|\geq-\log(t-1)$.

Similarly, by the first upper bound of Theorem 1.3, there is a point $u''\in$ Amoeba(f) within distance $(2t-3)\log(t-1)$ of v'. Clearly, $|w-u'|\leq|w-u''|$. Also, by the Triangle Inequality, $|w-u''|\leq|w-v'|+|v'-u''|$. So then, $\inf_{u\in\text{Amoeba}(f)}|u-w|\leq\inf_{v\in\text{ArchTrop}(f)}|v-w|+(2t-3)\log(t-1)$, and thus

$$\inf_{u\in\text{Amoeba}(f)}|u-w|-\inf_{v\in\text{ArchTrop}(f)}|v-w|\leq(2t-3)\log(t-1).$$

So our first assertion is proved.

Now if f has coefficients with real and imaginary parts that are rational, and n is fixed, Theorem 1.5 (which we've already proved) tells us that we can decide whether w lies in ArchTrop(f) using a number of bit operations polynomial in size(w) + size(f). So we may assume $w\not\in\text{ArchTrop}(f)$ and $\dim\sigma_w=n$.

Theorem 1.5 also gives us an explicit description of σ_w as the intersection of a number of half-spaces polynomial in t. Moreover, σ_w is ℓ-rational (recall Definition 2.14), with size polynomial in size(f). So we can compute the distance D from w to ArchTrop(f) by finding which facet of σ_w has minimal distance to w. The distance from w to any such facet can be approximated to the necessary number of bits in polynomial-time via Theorem 2.13 and the classical formula for distance between a point and an affine hyperplane: $\inf_{u\in\{x\mid r\cdot x=s\}}|u-w|=(|r\cdot w|-\text{sign}(r\cdot w)s)/|r|$. More precisely, comparing the facet distances reduces to checking the sign of an expression of the form $\gamma_1+\gamma_2\log\left(\frac{c_i}{c_{i'}}\right)+\gamma_3\log\left(\frac{c_j}{c_{j'}}\right)$ where γ_1 (resp. γ_2, γ_3) is a rational linear combination of $\sqrt{|a_i-a_{i'}|}$ and $\sqrt{|a_j-a_{j'}|}$ (resp. rational multiple of $\sqrt{|a_i-a_{i'}|}$ or $\sqrt{|a_j-a_{j'}|}$), with coefficients of size polynomial in size(f), for some indices $i,i',j,j'\in[t]$. We can then efficiently approximate D by approximating the underlying square-roots and logarithms to sufficient precision. The latter can be accomplished by Arithmetic-Geometric Iteration, as detailed in [9], and the amount of precision needed is explicitly bounded by an earlier variant of Theorem 2.13 covering inhomogeneous linear combinations of logarithms of algebraic numbers with algebraic coefficients [5]. The resulting bounds are somewhat worse than in Theorem 2.13, but still allow us to find polynomially many leading bits of $\inf_{v\in\text{Amoeba}(f)}|v-w|$ (for $w\in\mathbb{Q}^n$) in time polynomial in size(w) + size(f).

To prove the final assertion, we merely decide whether $\inf_{v\in \mathrm{ArchTrop}(f)} |v - w|$ strictly exceeds $\log(t-1)$ or not. To do so, we need only compute a polynomial number of leading bits of $\inf_{v\in \mathrm{ArchTrop}(f)} |v - w|$ (thanks to Theorem 2.13), and this takes time polynomial in $\mathrm{size}(w) + \mathrm{size}(f)$. Thanks to our initial observations using the Triangle Inequality, it is clear that Output (b) or Output (a) occurs according as $\inf_{v\in \mathrm{ArchTrop}(f)} |v - w| > \log(t-1)$ or not. So we are done. ■

6 Proving Theorem 1.6

6.1 Fast Cell Computation: Proof of the First Assertion

First, we apply Theorem 1.5 to (f_i, w) for each $i \in [k]$ to find which $\mathrm{ArchTrop}(f_i)$ contain w.

If w lies in no $\mathrm{ArchTrop}(f_i)$, then we simply use Corollary 2.15 (as in our proof of Theorem 1.5) to find an explicit description of the closure of the cell of $\mathbb{R}^n \setminus \bigcup_{i=1}^k \mathrm{ArchTrop}(f_i)$ containing w. Otherwise, we find the cells of $\mathrm{ArchTrop}(f_i)$ (for those i with $\mathrm{ArchTrop}(f_i)$ containing w) that contain w. Then, applying Corollary 2.15 once again, we explicitly find the unique cell of $\bigcap\limits_{\mathrm{ArchTrop}(f_i)\ni w} \mathrm{ArchTrop}(f_i)$ containing w.

Assume that f_i has exactly t_i monomial terms for all i. In either of the preceding cases, the total number of half-spaces involved is no more than $\sum_{i=1}^k t_i(t_i-1)/2$. So the overall complexity of our redundancy computations is polynomial in the input size and we are done. ■

6.2 Hardness of Detecting Mixed Vertices: Proving the Second Assertion

It will clarify matters if we consider a related **NP**-hard problem for rational polytopes first.

Ultimately, our proof boils down to a reduction from the following problem, equivalent to the famous **NP**-complete PARTITION problem (see below): Decide if a vertex of the hypercube $[-1,1]^n$ lies on a prescribed hyperplane defined by an equation of the form $a \cdot x = 0$ with $a \in \mathbb{Z}^n$. Because the coordinates of a are integral, we can replace the preceding equation by the inequality $0 \leq a \cdot x \leq 1/2$. With a bit more work, we can reduce PARTITION to the detection of a mixed vertex for a particular intersection of polyhedra. We now go over the details.

6.2.1 Preparation over $\mathbb{Q}$

In the notation of Definition 3.3, let us first consider the following decision problem. We assume all polyhedra are given explicitly as finite collections of rational linear inequalities, with size defined as in Sect. 2.2.

MIXED-VERTEX:

Given $n \in \mathbb{N}$ and polyhedra $P_1, \ldots, P_n$ in $\mathbb{R}^n$, does $P := \bigcap_{i=1}^n P_i$ have a mixed vertex? ■

While MIXED-VERTEX can be solved in polynomial time when n is fixed (by a brute-force check over all mixed n-tuples of facets), we will show that, for n varying, the problem is **NP**-complete, even when restricting to the case where all polytopes are full-dimensional and $P_1, \ldots, P_{n-1}$ are axes-parallel bricks.

Let e_i denote the $i^{\underline{\text{th}}}$ standard basis vector in $\mathbb{R}^n$ and let $M^\top$ denote the transpose of a matrix M. Also, given $\alpha \in \mathbb{R}^n$ and $b \in \mathbb{R}$, we will use the following notation for hyperplanes and certain half-spaces in $\mathbb{R}^n$ determined by α and b: $H_{(\alpha,b)} := \{x \in \mathbb{R}^n \mid \alpha \cdot x = b\}$, $H^{\leq}_{(\alpha,b)} := \{x \in \mathbb{R}^n \mid \alpha \cdot x \leq b\}$. For $i \in [n]$, let $s_i \in \mathbb{N}$, $M_i := [m_{i,1}, \ldots, m_{i,s_i}]^\top \in \mathbb{Z}^{s_i \times n}$, $b_i := (b_{i,1}, \ldots, b_{i,s_i})^\top \in \mathbb{Z}^{s_i}$, and $P_i := \{x \in \mathbb{R}^n \mid M_i x \leq b_i\}$. Since linear optimization can be done in polynomial-time (in the cases we consider) we may assume that the presentations $(n, s_i; M_i, b_i)$ are *irredundant*, i.e., P_i has exactly s_i facets if P_i is full-dimensional, and the sets $P_i \cap H_{(m_{i,j}, b_{i,j})}$, for $j \in [s_i]$, are precisely the facets of P_i for all $i \in [n]$.

Now set $P := \bigcap_{i=1}^n P_i$. Note that size(P) is thus linear in $\sum_{i=1}^n$ size(P_i).

Lemma 6.1 MIXED-VERTEX $\in$ **NP**.

Proof Since the binary sizes of the coordinates of the vertices of P are bounded by a polynomial in the input size, we can use vectors $v \in \mathbb{Q}^n$ of polynomial size as certificates. We can check in polynomial-time whether such a vector v is a vertex of P. If this is not the case, v cannot be a mixed vertex of P. Otherwise, v is a mixed vertex of P if and only if for each $i \in [n]$ there exists a facet F_i of P_i with $v \in F_i$. Since the facets of the polyhedra P_i admit polynomial-time decriptions as $\mathscr{H}$-polyhedra, this can be checked by a total of $s_1 + \cdots + s_n$ polyhedral membership tests. These membership tests are easily doable in polynomial-time since any of the underlying inequalities can be checked in polynomial-time and the number of faces of any P_i no worse than linear in the size of P_i.

So we can check in polynomial-time whether a given certificate v is a mixed vertex of P. Hence MIXED-VERTEX is in **NP**. ■

Since (in fixed dimension) we can actually list all vertices of P in polynomial-time, it is clear that MIXED-VERTEX can be solved in polynomial-time when n is fixed. When n is allowed to vary we obtain hardness:

Theorem 6.2 MIXED-VERTEX *is* **NP**-*hard, even in the special case where $P_1, \ldots, P_{n-1}$ are centrally symmetric axes-parallel bricks with vertex coordinates in $\{\pm 1, \pm 2\}$, and P_n has at most $2n + 2$ facets (with $2n$ of them parallel to coordinate hyperplanes).*

The proof of Theorem 6.2 will be based on a reduction from the following decision problem:

PARTITION

Given $d, \alpha_1, \ldots, \alpha_d \in \mathbb{N}$, is there an $I \subseteq [d]$ such that $\sum_{i \in I} \alpha_i = \sum_{i \in [d]\setminus I} \alpha_i$? ■

PARTITION was in the original list of **NP**-complete problems from [33].

Let an instance $(d; \alpha_1, \ldots, \alpha_d)$ of PARTITION be given, and set $\alpha := (\alpha_1, \ldots, \alpha_d)$. Then we are looking for a point $x \in \{-1, 1\}^d$ with $\alpha \cdot x = 0$.

We will now construct an equivalent instance of MIXED-VERTEX. With $n := d + 1$, $\bar{x} := (x_1, \ldots, x_{n-1})$ and $\mathbf{1}_n := (1, \ldots, 1) \in \mathbb{R}^n$ let

$$P_i := \left\{ (\bar{x}, x_n) | -1 \le x_i \le 1, \; -2 \le x_j \le 2 \text{ for all } j \in [n] \setminus \{i\} \right\}.$$

Also, for $i \in [n-1]$, let

$$P_n := \{ (\bar{x}, x_n) \mid -2 \cdot \mathbf{1}_{n-1} \le \bar{x} \le 2 \cdot \mathbf{1}_{n-1}, \; -1 \le x_n \le 1, \; 0 \le 2\alpha \cdot \bar{x} \le 1 \}$$

and set $P := \bigcap_{i=1}^n P_i$, $\hat{\alpha} := (\alpha, 0)$.

The next lemma shows that $P_n \cap \{-1, 1\}^n$ still captures the solutions of the given instance of partition.

Lemma 6.3 *$(d; \alpha_1, \ldots, \alpha_d)$ is a "no"-instance of* PARTITION *if and only if $P_n \cap \{-1, 1\}^n$ is empty.*

Proof Suppose, first, that $(d; \alpha_1, \ldots, \alpha_d)$ is a "no"-instance of PARTITION. If P_n is empty there is nothing left to prove. So, let $y \in P_n$ and $w \in \{-1, 1\}^{n-1} \times \mathbb{R}$. Since $\alpha \in \mathbb{N}^d$ we have $|\hat{\alpha} \cdot w| \ge 1$. Hence, via the Cauchy-Schwarz inequality, we have $1 \le |\hat{\alpha} \cdot w| = |\hat{\alpha} \cdot y + \hat{\alpha} \cdot (w - y)| \le |\hat{\alpha} \cdot y| + |\hat{\alpha} \cdot (w - y)| \le \frac{1}{2} + |\hat{\alpha}| \cdot |w - y| = \frac{1}{2} + |\alpha| \cdot |w - y|$ and thus $|w - y| \ge \frac{1}{2|\alpha|} > 0$. Therefore $P_n \cap \left(\{-1, 1\}^{n-1} \times \mathbb{R}\right)$ is empty.

Conversely, if $P_n \cap \{-1, 1\}^n$ is empty, then there is no $x \in \{\pm 1\}^{n-1}$ such that $0 \le \alpha \cdot \bar{x} \le \frac{1}{2}$. Since $\hat{\alpha} \in \mathbb{N}^{n-1}$, we have that $(d, \alpha_1, \ldots, \alpha_d)$ is a "No"-instance of PARTITION. ■

The next lemma reduces the possible mixed vertices to the vertical edges of the standard cube.

Lemma 6.4 *Following the preceding notation, let v be a mixed vertex of $P := \bigcap_{i=1}^n P_i$. Then $v \in \{-1, 1\}^{n-1} \times [-1, 1]$.*

Proof First note that $Q := \bigcap_{i=1}^{n-1} P_i = [-1, 1]^{n-1} \times [-2, 2]$. Therefore, for each $i \in [n-1]$, the only facets of P_i that meet Q are those in $H_{(e_i, \pm 1)}$ and $H_{(e_n, \pm 2)}$. Since $P \subset [-1, 1]^n$, and for each $i \in [n-1]$ the mixed vertex v must be contained in a facet of P_i, we have $v \in [-1, 1]^n \cap \bigcap_{i=1}^{n-1} \left(\bigcup_{\delta_i \in \{-1,1\}} H_{(e_i, \delta_i)} \right) = \{-1, 1\}^{n-1} \times [-1, 1]$, which proves the assertion. ■

The next lemma adds P_n to consideration.

Lemma 6.5 *Let v be a mixed vertex of $P := \bigcap_{i=1}^{n} P_i$. Then $v \in \{-1, 1\}^n$.*

Proof By Lemma 6.4, $v \in \{-1, 1\}^{n-1} \times [-1, 1]$. Since the hyperplanes $H_{(e_i,\pm 2)}$ do not meet $[-1, 1]^n$, we have $v \notin H_{(e_i,-2)} \cup H_{(e_i,2)}$ for all $i \in [n-1]$. Hence, v can only be contained in the constraint hyperplanes $H_{(\hat{\alpha},0)}, H_{(2\hat{\alpha},1)}, H_{(e_n,-1)}, H_{(e_n,1)}$. Since $\hat{\alpha} \in \mathbb{R}^{n-1} \times \{0\}$, the vector $\hat{\alpha}$ is linearly dependent on $e_1, \ldots, e_{n-1}$. Hence, $v \in H_{(e_n,-1)} \cup H_{(e_n,1)}$, i.e., $v \in \{-1, 1\}^n$. ■

We can now prove the **NP**-hardness of MIXED-VERTEX.

Proof of Theorem 6.2 First, let $(d; \alpha_1, \ldots, \alpha_d)$ be a "yes"-instance of PARTITION, let $x^* := (\xi_1^*, \ldots, \xi_{n-1}^*) \in \{-1, 1\}^{n-1}$ be a solution, and set $\xi_n^* := 1$, $v := (x^*, \xi_n^*)$, $F_i := H_{(e_i,\xi_i^*)} \cap P_i$ for all $i \in [n]$, and $\hat{F}_n := H_{(\hat{\alpha},0)} \cap P_n$. Then $v \in \hat{F}_n \subset P_n$, hence $v \in P$ and, in fact, v is a vertex of P. Furthermore, F_i is a facet of P_i for all $i \in [n]$, $v \in \bigcap_{i=1}^{n} F_i$, and thus v is a mixed vertex of P.

Conversely, let $(d; \alpha_1, \ldots, \alpha_d)$ be a "no"-instance of PARTITION, and suppose that $v \in \mathbb{R}^n$ is a mixed vertex of P. By Lemma 6.5, $v \in \{-1, 1\}^n$. Furthermore, v lies in a facet of P_n. Hence, in particular, $v \in P_n$, i.e., $P_n \cap \{-1, 1\}^n$ is non-empty. Therefore, by Lemma 6.3, $(d; \alpha_1, \ldots, \alpha_d)$ is a "yes"-instance of PARTITION. This contradiction shows that P does not have a mixed vertex.

Clearly, the transformation works in polynomial-time. ■

6.3 *Proof of the Second Assertion of Theorem 1.6*

It clearly suffices to show that the following variant of MIXED-VERTEX is **NP**-hard:
LOGARITHMIC-MIXED-VERTEX:
Given $n \in \mathbb{N}$ and ℓ-rational polyhedra $P_1, \ldots, P_n \subset \mathbb{R}^n$, does $P := \bigcap_{i=1}^{n} P_i$ have a mixed vertex? ■

Via an argument completely parallel to the last section, the **NP**-hardness of LOGARITHMIC-MIXED-VERTEX follows immediately from the **NP**-hardness of the following variant of PARTITION:
LOGARITHMIC-PARTITION
Given $d \in \mathbb{N}$, $\alpha_1, \ldots, \alpha_d \in \mathbb{N} \setminus \{0\}$, is there an $I \subseteq [d]$ such that $\sum_{i \in I} \log \alpha_i = \sum_{i \in [d] \setminus I} \log \alpha_i$? ■

We measure size in LOGARITHMIC-PARTITION just as in the original PARTITION Problem: $\sum_{i=1}^{d} \log \alpha_d$. Note that LOGARITHMIC-PARTITION is equivalent to the obvious variant of PARTITION where we ask for a partition making the two resulting *products* be identical. The latter problem is known to be **NP**-hard as well, thanks to [45], and is in fact also strongly **NP**-hard. ■

References

1. D'Andrea, C., Krick, T., Sombra, M.: Heights of varieties in multiprojective spaces and arithmetic Nullstellensatze. Ann. Sci. l'ENS fascicule **4**, 549–627 (2013)
2. D'Andrea, C., Galligo, A., Sombra, M.: Quantitative equidistribution for the solution of a system of sparse polynomial equations. Am. J. Math (to appear)
3. Arora, S., Barak, B.: Computational Complexity. A Modern Approach. Cambridge University Press, Cambridge (2009)
4. Avendaño, M., Kogan, R., Nisse, M., Rojas, J.M.: Metric Estimates and membership complexity for archimedean amoebae and tropical hypersurfaces. Submitted for publication, also available as Math ArXiV preprint `1307.3681`
5. Baker, A.: The theory of linear forms in logarithms. In: Transcendence Theory: Advances and Applications: Proceedings of a Conference Held at the University of Cambridge, Cambridge, January–February 1976. Academic, London (1977)
6. Baker, M., Rumely, R.: Potential Theory and Dynamics on the Berkovich Projective Line. Mathematical Surveys and Monographs, vol. 159. American Mathematical Society, Providence (2010)
7. Bates, D.J., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Numerically Solving Polynomial Systems with Bertini. Software, Environments and Tools Series. Society for Industrial and Applied Mathematics (2013)
8. Beltrán, C., Pardo, L.M.: Smale's 17th problem: Average polynomial time to compute affine and projective solutions. J. Am. Math. Soc. **22**, 363–385 (2009)
9. Bernstein, D.J.: Computing Logarithm Intervals with the Arithmetic-Geometric Mean Iterations. Available from http://cr.yp.to/papers.html
10. Bettale, L., Faugére, J.-C., Perret, L.: Cryptanalysis of HFE, multi-HFE and variants for odd and even characteristic. Designs Codes Cryptogr. **69**(1), 1–52 (2013)
11. Bihan, F., Rojas, J.M., Stella, C.: Faster real feasibility via circuit discriminants. In: Proceedings of ISSAC 2009 (July 28–31, Seoul, Korea), pp. 39–46. ACM (2009)
12. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer (1998)
13. Bürgisser, P., Scheiblechner, P.: On the complexity of counting components of algebraic varieties. J. Symb. Comput. **44**(9), 1114–1136 (2009)
14. Bürgisser, P., Cucker, F.: Solving polynomial equations in smoothed polynomial time and a near solution to Smale's 17th problem. In: Proceedings STOC (Symposium on the Theory of Computation) 2010, pp. 503–512. ACM (2010)
15. De Loera, J.A., Rambau, J., Santos, F.: Triangulations, Structures for Algorithms and Applications, Algorithms and Computation in Mathematics, vol. 25. Springer, Berlin (2010)
16. Dickenstein, A., Feichtner, E.M., Sturmfels, B.: Tropical discriminants. J. Am. Math. Soc. **20**(4), 1111–1133 (2007)
17. Einsiedler, M., Kapranov, M., Lind, D.: Non-Archimedean amoebas and tropical varieties. J. die reine Angew. Math. (Crelles J.) **2006**(601), 139–157 (2006)
18. Emiris, I.Z., Canny, J.: Efficient incremental algorithms for the sparse resultant and mixed volume. J. Symb. Comput. **20**(2), 117–149 (1995)
19. Faugère, J.-C., Gaudry, P., Huot, L., Renault, G.: Using symmetries in the index calculus for elliptic curves discrete logarithm. J. Cryptol. 1–40 (2013)
20. Faugère, J.-C., Hering, M., Phan, J.: The membrane inclusions curvature equations. Adv. Appl. Math. **31**(4), 643–658 (2003).
21. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco (1979)
22. Gel'fand, I.M., Kapranov, M.M., Zelevinsky, A.V.: Discriminants, Resultants and Multidimensional Determinants. Birkhäuser, Boston (1994)

23. Gritzmann, P.: Grundlagen der Mathematischen Optimierung: Diskrete Strukturen, Komplexitätstheorie, Konvexitätstheorie, Lineare Optimierung, Simplex-Algorithmus, Dualität. Springer (2013)
24. Gritzmann, P., Klee, V.: On the complexity of some basic problems in computational convexity: I. Containment problems. Discrete Math. **136**, 129–174 (1994)
25. Gritzmann, P., Klee, V.: On the complexity of some basic problems in computational convexity: II. Volume and mixed volumes. In: Bisztriczky, T., McMullen, P., Schneider, R., Ivic Weiss A. (eds) Polytopes: Abstract, Convex and Computational, pp. 373–466. Kluwer, Boston (1994)
26. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, New York (1993)
27. Grünbaum, B.: Convex Polytopes. Wiley-Interscience, London (1967); Ziegler, G. (ed) Convex Polytopes, 2nd edn. Graduate Texts in Mathematics, vol. 221. Springer (2003)
28. Hao, W., Hauenstein, J.D., Hu, B., Liu, Y., Sommese, A., Zhang, Y.-T.: Continuation along bifurcation branches for a tumor model with a necrotic core. J. Sci. Comput. **53**, 395–413 (2012)
29. Hauenstein, J., Rodriguez, J., Sturmfels, B.: Maximum likelihood for matrices with rank constraints. J. Algebraic Stat. **5**(1), 18–38 (2014). Also available as Math ArXiV preprint `1210.0198`
30. Herbst, M., Hori, K., Page, D.: Phases of $N = 2$ theories in $1 + 1$ dimensions with boundary. High Energy Phys. ArXiV preprint `0803.2045v1`
31. Huan, L.J., Li, T.-Y.: Parallel homotopy algorithm for symmetric large sparse eigenproblems. J. Comput. Appl. Math. **60**(1–2), 77–100 (1995)
32. Itenberg, I., Mikhalkin, G., Shustin, E.: Tropical Algebraic Geometry, 2nd edn. Oberwolfach Seminars, vol. 35. Birkhäuser Verlag, Basel (2009)
33. Karp, R.M.: Reducibility among combinatorial problems. In Miller, R.E., Thatcher, J.W. (eds) Complexity of Computer Computations, pp. 85–103. Plenum, New York (1972).
34. Khachiyan, L.: A polynomial algorithm in linear programming. Soviet Math. Doklady **20**, 191–194 (1979)
35. Khovanskii, A.G.: Fewnomials. AMS, Providence (1991)
36. Koiran, P., Portier, N., Rojas, J.M.: Counting Tropically Degenerate Valuations and p-adic Approaches to the Hardness of the Permanent. Submitted for publication, also available as Math ArXiV preprint `1309.0486`
37. Lee, T.-L., Li, T.-Y.: Mixed volume computation in solving polynomial systems. In: Randomization, Relaxation, and Complexity in Polynomial Equation Solving, Contemporary Mathematics, vol. 556, pp. 97–112. AMS (2011)
38. Litvinov, G.L., Sergeev, S.N. (eds) Tropical and Idempotent Mathematics, International Workshop TROPICAL-07 (Tropical and Idempotent Mathematics, August 25—30, 2007. Independent University, Contemporary Mathematics, vol. 495. AMS (2009)
39. Maclagan, D., Sturmfels, B.: Introduction to Tropical Geometry, AMS Graduate Studies in Mathematics, vol. 161, AMS (2015, to appear).
40. Matveev, E.M.: An explicit lower bound for a homogeneous rational linear form in logarithms of algebraic numbers. II. Izv. Ross. Akad. Nauk Ser. Mat. **64**(6), 125–180 (2000); translation in Izv. Math. **64**(6), 1217–1269 (2000)
41. Mikhalkin, G.: Decomposition into pairs-of-pants for complex algebraic hypersurfaces. Topology **43**(5), 1035–1065 (2004)
42. Müller, S., Feliu, E., Regensburger, G., Conradi, C., Shiu, A., Dickenstein, A.: Sign conditions for injectivity of generalized polynomial maps with applications to chemical reaction networks and real algebraic geometry. Math ArXiV preprint `1311.5493`
43. Nesterenko, Y.: Linear Forms in Logarithms of Rational Numbers. Diophantine Approximation (Cetraro, 2000). Lecture Notes in Math., vol. 1819, pp. 53–106. Springer, Berlin (2003)
44. Newton, I.: Letter to Oldenburg dated 1676 Oct 24, the correspondence of Isaac Newton, II, pp. 126–127. Cambridge University Press, Cambridge (1960)
45. Ng, CTD., Barketau, M.S., Cheng, T.C.E., Kovalyov, M.Y.: Product Partition and related problems of scheduling and systems reliability: Computational complexity and approximation. Eur. J. Oper. Res. **207**, 601–604 (2010)

46. Nilsson, L., Passare, M.: Discriminant coamoebas in dimension two. J. Commut. Algebra **2**(4), 447–471 (2010)
47. Ning, Y., Avendaño, M.E., Mortari, D.: Sequential design of satellite formations with invariant distances. AIAA J. Spacecraft Rockets **48**(6), 1025–1032 (2011)
48. Nisse, M., Sottile, F.: Phase limit set of a variety. J. Algebra Number Theory **7**(2), 339–352 (2013)
49. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1995)
50. Phillipson, K.R., Rojas, J.M.: $\mathscr{A}$-discriminants, and their cuttings, for complex exponents (2014, in preparation)
51. Pin, J.-E.: Tropical semirings. Idempotency (Bristol, 1994), Publ. Newton Inst., 11, pp. 50–69. Cambridge Univ. Press, Cambridge (1998)
52. Plaisted, D.A.: New NP-hard and NP-complete polynomial and integer divisibility problems. Theor. Comput. Sci. **31**(1–2), 125–138 (1984)
53. Schrijver, A.: Theory of Linear and Integer Programming. Wiley (1986)
54. Shub, M., Smale, S.: The complexity of Bezout's theorem I: Geometric aspects. J. Am. Math. Soc. **6**, 459–501 (1992)
55. Shub, M., Smale, S.: The Complexity of Bezout's theorem II: Volumes and Probabilities. In: Eyssette, F., Galligo, A. (eds) Computational Algebraic Geometry, pp. 267–285. Birkhauser (1992)
56. Shub, M., Smale, S.: The Complexity of Bezout's theorem III: Condition number and packing. J. Complexity **9**, 4–14 (1993)
57. Shub, M., Smale, S.: The complexity of Bezout's theorem IV: Probability of success; extensions. SIAM J. Numer. Anal. **33**(1), 128–148 (1996)
58. Shub, M., Smale, S.: The complexity of Bezout's theorem V: Polynomial time. Theor. Comput. Sci. **133**(1), 141–164 (1994).
59. Sipser, M.: Introduction to the Theory of Computation, 3rd edn. Cengage Learning (2012)
60. Smale, Steve, Mathematical problems for the next century. Math. Intelligencer **20**(2), 7–15 (1998)
61. Smale, S.: Mathematical Problems for the Next Century. Mathematics: Frontiers and Perspectives, pp. 271–294. Amer. Math. Soc., Providence (2000)
62. Sommese, A.J., Wampler, C.W.: The Numerical Solution to Systems of Polynomials Arising in Engineering and Science. World Scientific, Singapore (2005)
63. Spielman, D., Teng, S.H.: Smoothed analysis of termination of linear programming algorithms. Math. Program. Ser. B **97** (2003)
64. Verschelde, J.: Polynomial homotopy continuation with PHCpack. ACM Commun. Comput. Algebra **44**(4), 217–220 (2010)
65. Viro, O.Y.: Dequantization of real algebraic geometry on a logarithmic paper. In: Proceedings of the 3rd European Congress of Mathematicians. Progress in Math, vol. 201, pp. 135–146. Birkhäuser (2001)
66. Ziegler, G.M.: Lectures on Polytopes, Graduate Texts in Mathematics. Springer (1995)

Slycat Ensemble Analysis of Electrical Circuit Simulations

Patricia J. Crossno, Timothy M. Shead, Milosz A. Sielicki, Warren L. Hunt, Shawn Martin, and Ming-Yu Hsieh

1 Ensembles and Sensitivity Analysis

With recent advances in computational power, scientists can now run thousands of related simulations to explore a single problem. We refer to such a group of related simulations as an *ensemble*. More generally, an ensemble can be thought of as a set of samples or observations, each consisting of the same set of variables, in a shared high-dimensional space describing a particular problem domain. Practically, an ensemble is a collection of data sets with common attributes that we wish to analyze as a whole. Thus *ensemble analysis* is a form of meta-analysis that looks at the combined behaviors and features of the ensemble in an effort to understand and describe the underlying problem space. By looking at the collection as a whole, higher level patterns emerge beyond what can be seen by examining individual simulation runs.

As an example, *sensitivity analysis* is a type of ensemble analysis that evaluates how changes in simulation input parameters correlate with changes in simulation results. In addition to revealing the types and strengths of relationships between inputs and outputs, sensitivity analysis can be used to verify that simulation results are within expected ranges and to validate that the underlying model is behaving correctly. Unexpected results could point to unexplored physical regimes, poorly understood parameter spaces, or something as mundane as flaws in simulation codes. Input parameters form the set of independent variables, and outputs the set of dependent variables. Commonly, sensitivity analyses are performed using either simple regression (correlating a single input to a single output at a time), or multiple regression (correlating a group of inputs to a single output). However, neither of

P.J. Crossno (✉) • T.M. Shead • M.A. Sielicki • W.L. Hunt • S. Martin • M.-Y. Hsieh
Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185, USA
e-mail: pjcross@sandia.gov; tshead@sandia.gov; masieli@sandia.gov; wlhunt@sandia.gov; smartin@sandia.gov; myhsieh@sandia.gov

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4_16

these approaches provides a means for evaluating the collective relationships among multiple inputs and multiple outputs.

Our introduction to this work began while evaluating the likely impacts on workflows in analysis and visualization from proposed architectural changes for exascale computing. As part of the evaluation, we interviewed analysts working at Sandia National Laboratories in a variety of simulation domains, including thermal, solid mechanics, and electrical circuit analysis. Sensitivity analysis is a common component within each domain's work flow, although the simulation results vary widely, ranging from simple tables of metrics to time series and finite element models. The analysts typically use Dakota [2] to manage ensemble creation, using custom scripts to extract scalar metrics from finite element outputs or time series. The metrics are merged with tables of original input parameters and analyzed using Dakota, JMP, Matlab, Minitab, Excel, and other tools. Existing visualization tools such as ParaView [13] and EnSight [7] are used for remote visualization of large simulation results. However, these tools are fundamentally designed to visualize individual simulations, or handfuls of simulations that are loaded into memory simultaneously and visually superimposed. Ensembles containing hundreds or thousands of simulations require a different type of analysis, a different visual abstraction, and a different system architecture to effectively integrate so many results.

Our investigation led to the creation of Slycat, a system designed to meet the needs of ensemble analysis. For sensitivity analysis, and parameter studies in particular, Slycat provides a visual interface to answer the following questions about a given ensemble:

- Which input parameters are most strongly correlated with specific outputs?
- Which input parameters exhibit little or no impact on output responses?
- Which simulations are anomalous, and in what ways are they different?

We use Canonical Correlation Analysis (CCA) to model the relationships between input and output parameters because it maps well to the structure of our problem, especially in its ability to correlate multiple inputs against multiple outputs. Although powerful and ideally suited to the problem, CCA results can be difficult to interpret; making CCA accessible to domain experts through the tight integration of useful visualizations with iterative exploratory analysis is the central contribution of this work.

2 Related Work

The work that most closely aligns with our own is that of Degani et al., which applies CCA to the analysis of correlations between the operating environment of a Boeing aircraft and the actions and responses of the pilots [8]. Presenting the CCA correlations in a circular layout called a heliograph, the positive or negative correlation weights of different variables are represented as outward or inward facing bars, respectively. The concentric stacking of CCA components in the same plot leads to overlapping bars from adjacent components, potentially leading to misinterpretations of the results. Our system provides a simpler, easier to understand visualization of CCA results as applied to multiple inputs and outputs.

Much of the earlier ensemble research deals with data sets that are either geospatial or spatiotemporal in nature. Consequently, their analysis and visualization approaches rely on this, making them unsuitable for ensembles that lack a spatial component. For example, Wilson and Potter explore geospatial weather data and discuss how ensembles mitigate uncertainty in simulations [31]. A similar approach from Potter employs isocontours over spatial domains [20, 21]. Noodles, another tool for modeling and visualizing uncertainty in ensembles of weather predictions, displays the spatial distributions of inter-simulation uncertainty through a combination of ribbons and glyphs drawn as map overlays [22]. Waser et al. integrate computational steering controls into a spatiotemporal flood simulation framework, enabling users to steer parameter studies and generate ensembles on demand [29, 30].

Another branch of ensemble visualization research uses feature extraction. However, these techniques can still have spatial dependencies. In the technique of Smith et al., clustering based on feature identification is performed on time-varying, spatial data [24]. A suite of feature detection techniques, including CCA, is used by Sukharev et al. to reveal structure in multivariate, time-varying climate data sets. Once their data has been clustered, segmented, and correlations computed, the results are geo-spatially overlaid on the weather prediction region for visualization and interpretation [27]. Hummel et al., classifies fluid transport variance within neighborhoods over flow field ensembles. Linked views enable selection in the fluid feature space to produce a visualization over the physical domain [11]. Another system, EDEN, incorporates several multivariate feature detection techniques in a single interface [26]. Piringer et al. visualize multivariate data using downsampling, 3D surface plots, extracted scalar features, and glyph-based visualizations to explore an ensemble of 2D functions. In addition to comparing ensemble member functions against each other, this work attempts to illustrate the distribution of features across the ensemble [19].

CCA has also been used to analyze spatial data. To more clearly identify the relationships in functional magnetic resonance imaging data, Karhunen et al. have developed a method that exploits CCA prior to applying blind source separation techniques [12]. The research of Ge et al. demonstrates CCA correlations that reflect the spatial correlations in multiple sensor arrays, even in the presence of noise [9]. In a recent paper by Marzban et al., CCA is shown to capture complex weather relationships between model parameters and forecast quantities [15].

More broadly, parameter studies have a relationship to our work, though their intent is often either based on the design of experiments or on directing simulation results towards a particular outcome, neither of which aligns with our particular system goals. Examples include systems for exploring the relationship between input parameters and aesthetic visual effects in animations [5], and the steering of designs [6].

Matkovic et al. recognize the need for advanced tools to support engineers in visualizing and understanding ensembles, and incorporate in their system multivariate visualization techniques such as parallel plotting, multiple linked views, and scatterplots to display one-to-one correlations [16]. However, one-to-one correlation analysis is insufficient for evaluating the complex, multivariate relationships inherent in our user's data.

Within sensitivity analysis, sampling tools are typically relied on to provide coverage of the simulation parameter space [1, 2]. Even with a sampling method in place, much system behavior is unknown and there is more work to do to uncover input-output relationships. A study by Song and Zhao employs a variance-based method to identify the first-order model sensitivities when applied to forest growth simulations [25]. Other sensitivity analysis work applies classic statistical methods, such as ANOVA, to U.S. immigration model results [23], and statistical aggregation to models of large distributed computing systems [17]. In contrast, our research is focused on understanding the behavior of an ensemble with the intent of exposing hidden relationships between the simulation input parameters and the results, without the emphasis on numerical quantification of uncertainty.

3 System Architecture

To support answering the questions outlined in Sect. 1, and to support additional analysis types in the future, we designed Slycat around the following general requirements:

- Remote ensemble analysis, in which large data is analyzed in-place to minimize data movement.
- Ubiquitous access to analysis results regardless of the availability of the original data sets or source platforms.
- Desktop delivery providing interactive exploration of ensemble analysis results, and collaborative sharing with appropriate access controls.

The need for remote ensemble analysis is driven by the ever widening gap between high performance computing (HPC) computational performance and I/O performance. Practically speaking, we have reached a point where computation is effectively "free" while data movement has become very expensive, and moving raw ensemble data from the HPC platform where it was generated to the host running Slycat would take significantly more time than the analysis computations to follow! Better instead to perform those computations on the machine where the ensemble data is located, so that only the model—typically orders of magnitude smaller than the original data—is moved across the network to the Slycat host. This leads to the Slycat architectural design of Fig. 1.

An important practical consideration for users of HPC platforms is that ensemble results may often become temporarily or permanently unavailable—login nodes come and go due to resource contention, users often must archive or delete their

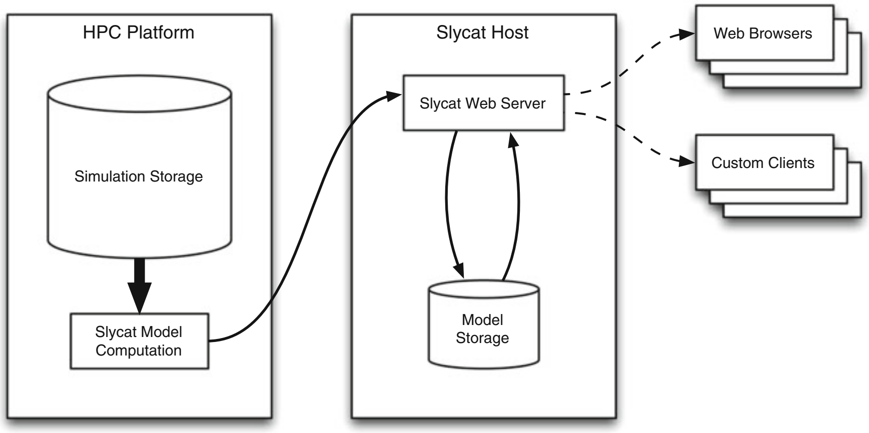


Fig. 1 Slycat system diagram depicting how large data on an HPC platform is analyzed in-place to produce greatly reduced model artifacts that are stored by the Slycat web server. Later, these artifacts are delivered—incrementally and on-demand—to interactive clients

data as scratch filesystems near capacity, and so on. Because Slycat stores its own greatly-reduced models of the underlying raw data, and only those models are necessary to produce a visualization, users can continue to access their Slycat analysis results even when the original HPC resources are unavailable.

Finally, we want a system architecture that supports easy desktop delivery and collaboration, enabling users to share results seamlessly with colleagues across the network without requiring any software downloads or installation. That means using existing web standards and clients, dictating much of the subsequent design and derived requirements. It means adopting a web server as the front-end for the system and standard web browsers as clients (or custom clients using standard web protocols to communicate). In turn, interactions and visualizations must be developed using only the set of technologies that are widely available within web browsers, such as HTML5, JavaScript, AJAX, SVG, and Canvas.

Unlike dedicated visualization tools such as ParaView or Ensight, we cannot rely on the client to perform serious calculations. This necessitates Slycat's pre-computation of visualization artifacts, organized for rapid, incremental retrieval from the server on-demand. As an example, we allow users to interact with data tables that can contain thousands of columns and millions of rows - which would cause unacceptable slowdowns if they had to be transferred from server to client in their entirety. Instead, only the data needed to display the currently-visible table rows are transferred as the user scrolls through the table. This "just-in-time" approach to data is used throughout the client, minimizing total bandwidth consumption and keeping the interface responsive.

Although working around the constraints of web browsers has been a challenge, the rewards have been significant, enabling Slycat users to "bookmark" the state of a visualization and share it with colleagues simply by sharing a hyperlink.

4 Canonical Correlation Analysis

H. Hotelling, who was also instrumental in developing Principal Component Analysis, proposed Canonical Correlation Analysis (CCA) [10]. CCA is a method that can be used to understand relationships between two sets of multivariate data. Writing our sets as matrices, $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]_{p\times n}$ is presumed to be independent, and $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]_{q\times n}$ to be dependent, where the $\mathbf{x}_i$ and $\mathbf{y}_j$ lie in $\mathbb{R}^p$ and $\mathbb{R}^q$, respectively (i.e. each vector $\mathbf{x}_i$ has p components and each vector $\mathbf{y}_j$ has q components). CCA attempts to find projections $\mathbf{a}$ and $\mathbf{b}$ such that $R^2 = \text{corr}(\mathbf{a}^T X, \mathbf{b}^T Y)$ is maximized, where $\text{corr}(\bullet, \bullet)$ denotes the standard Pearson correlation, which in our case is given by

$$\text{corr}(\mathbf{a}^T X, \mathbf{b}^T Y) = \frac{\mathbf{a}^T C_{xy}\mathbf{b}}{\sqrt{\mathbf{a}^T C_{xx}\mathbf{a}\mathbf{b}^T C_{yy}\mathbf{b}}},$$

where C_{xx} is the $p \times p$ covariance matrix $\frac{1}{n}XX^T$, C_{yy} is the $q \times q$ covariance matrix $\frac{1}{n}YY^T$, and C_{xy} is the $p \times q$ covariance matrix $\frac{1}{n}XY^T$. (Note that we are assuming for convenience that X and Y are mean-subtracted and unit variance.)

Thus to find the projections $\mathbf{a}$ and $\mathbf{b}$ we want to solve

$$\begin{array}{ll} \text{maximize} & \mathbf{a}^T C_{xy}\mathbf{b} \\ \text{subject to} & \mathbf{a}^T C_{xx}\mathbf{a} = \mathbf{b}^T C_{yy}\mathbf{b} = 1. \end{array}$$

This problem reduces to a generalized eigenvalue problem, and thus has a unique, global minimum (up to subspace isomorphism and assuming adequate matrix ranks). CCA is a linear method and is a direct generalization of several standard statistical techniques, including PCA, multiple linear regression (MLR), and Partial Least Squares (PLS) [3, 4].

The vectors $\mathbf{a}^T X$ and $\mathbf{b}^T Y$ are known as the first pair of canonical variables. Further pairs of canonical variables are orthogonal and ordered by decreasing importance. In addition to the canonical variables, the R^2 value for each variable pair is obtained, and various statistics can be computed to determine the significance of the correlation. A common statistic used in this context is the p-value associated with Wilks' λ [14].

Once the canonical variables are determined, they can be used to understand how the variables in X are related to the variables in Y, although this should be done with some caution. The components of the vectors $\mathbf{a}$ and $\mathbf{b}$ can be used to determine the relative importance of the corresponding variables in X and Y. These components are known as canonical coefficients. However, the canonical coefficients are considered difficult to interpret and may hide certain redundancies in the data. For this reason, it is more typical to analyze the canonical loadings, also known as the structure coefficients. The structure coefficients are given by the correlations between the canonical variables and the original variables (e.g. $\text{corr}(\mathbf{a}^T X, X)$). The structure coefficients are generally preferred over the canonical coefficients due to the fact that they are more closely related to the original variables.

CCA is a powerful method that is particularly useful for certain problems because it acts on pairs of data sets (unlike PCA, which acts on a single data set). However, it is also complex and difficult to interpret. One of our central goals in this work has been to provide CCA-based analysis for data ensembles, but in a framework more understandable to domain experts.

5 Visualization

In Slycat, sensitivity analysis is performed through an iterative cycle of designating which input and output variables to include in the analysis, performing CCA, and visually exploring the resulting model. Analysis typically starts with an all-to-all evaluation to get an initial sense of the data, revealing the most strongly correlated combinations of variables. Some cases require iterative refinement to tease apart disparate groups of inputs and outputs.

To provide integrated perspectives, Slycat combines three levels of detail in a single web page using multiple linked views, as shown in Fig. 2. In the upper left, the *Correlation View* shows the relationships found for the ensemble as a whole, displaying structure coefficients in tabular form, grouped by correlation components

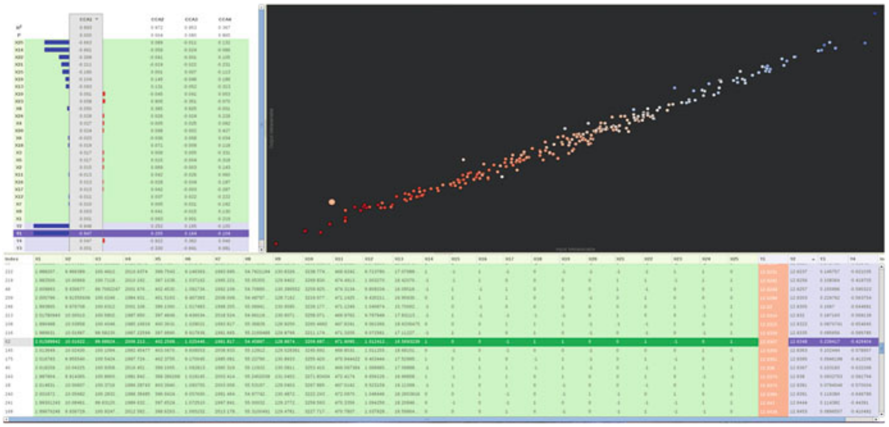


Fig. 2 Slycat visualization showing an all-to-all CCA analysis of a 250 simulation electrical circuit ensemble. As seen through the lengths and shared colors/directions of the bars in the *Correlation View* in the *upper left*, the first CCA component exhibits a strong positive correlation predominantly between the combined inputs X25 and X14 and both of the outputs Y1 and Y2, an example of a many-to-many relationship. In the *Simulation View* scatterplot on the *upper right*, the highlighted simulation (enlarged point) on the *lower left* can be seen to be anomalous given its offset from the diagonal and its lower Y1 value (*pink*) relative to other simulations with similar inputs (*reds*)

into columns and by ensemble variables into rows. The top two rows display each component's R^2 and p-value. Variable names are shown along the left edge of the view. Rows for input variables are colored green, while output variable rows are lavender. This green/purple color coding is used consistently throughout the interface to designate inputs and outputs. The rows in each column can be sorted by the correlation strengths of the variables in either ascending or descending order.

Users select a component by clicking its column header, expanding the coefficients into an inline vertical bar chart. The bars visually encode the signed values of the structure coefficients for each variable, with left-facing blue bars representing negative values and right-facing red bars representing positive ones. Color coding the bars visually reinforces the relationship types. Variables with matching colors are positively correlated, and variables with mismatched colors are negatively correlated. Bar length indicates a variable's relative importance. Sorting a component in descending order separately displays the input and output variables in order of significance with the longest bars at the top and the shortest at the bottom. This ordering makes it simple to evaluate which variables exhibit strong correlations, whether the correlations represent positive or negative relationships between the variables, and which inputs are driving a particular group of results.

The scatterplot in the upper right of Fig. 2 is the *Simulation View*. It displays how well individual simulations are described by the correlations for the ensemble as a whole. The axes are the canonical variables, $\mathbf{a}^T X$ and $\mathbf{b}^T Y$, and each simulation is rendered as a point with x and y coordinates computed as sums of input and output values, respectively. This coordinate space is highly abstract, with the x-axis representing a metavariable that is the sum of all the inputs and the y-axis a metavariable of all the outputs. Consequently, the scatterplot changes whenever a new canonical component is selected, since each component's structure coefficients are different. If the model finds a perfect linear correlation between inputs and outputs, the scatterplot points will form a diagonal line. Anomalous simulations will appear offset from the diagonal as positional outliers.

Points can be color coded by the values of input or output variables, providing another way to identify outliers. Selecting a row in the *Correlation View* or a column header in the *Variable Table* (see below) selects that variable's values for color coding in the scatterplot. We use a diverging blue/white/red color map [18] to encode the values, where blue is at the low end of the scale, and red is at the high end. While we do not assign any particular meaning to the central values shown in white, we find in practice that the diverging color map makes it easier to interpret values of nearby points.

Across the bottom of the page, the *Variable Table* displays raw data with a row for each individual simulation and a column for every ensemble variable. Column (variable) selections in the *Variable Table* are linked with row selections in the *Correlation View* and the scatterplot color map. The selected column is color coded using the same color map as the scatterplot to visually correlate the two views. The table columns can be sorted in ascending or descending order. Additionally, row (simulation) selections are linked with point selections in the scatterplot. Darker green/purple backgrounds highlight selected table rows, while selected points in the scatterplot are drawn with a larger radius.

6 Electrical Simulation Sensitivity Analysis

Our users model electrical circuits using Xyce, an open source, high-performance circuit simulator developed by Sandia National Laboratories as part of the Advanced Simulation and Computing (ASC) Program [28]. We will examine two circuit ensembles of differing scales: a small ensemble of 250 runs, and a large ensemble of 2,641 runs. In both cases, some of the input variables take a restricted set of values (−1, 0, or 1). These values are used to select different simulation models whose responses are *low*, *nominal*, and *high*, respectively. The models act to encapsulate groups of input variables, thereby reducing the number of variables and the size of the ensemble.

6.1 Small Ensemble

The small ensemble has 250 simulations, each with 25 input variables and 4 output variables. Outputs Y1 and Y2 measure voltage responses, while Y3 and Y4 measure current. The goal of this analysis is to answer the first two questions from the list in Sect. 1: Which inputs are most strongly correlated with specific outputs, and which inputs exhibit little or no impact on the outputs?

As seen in the *Correlation View* bar chart in Fig. 2, the first CCA component shows a positively correlated relationship that is mostly between the input parameters X25 and X14 and both of the voltage outputs. The inputs are listed in decreasing order of importance, so the parameters at the bottom of the green region in the first column exhibit little or no impact on voltage responses.

In Fig. 3, the sorted variables for the second CCA component reveal a strong negative correlation predominantly between the input X23 and the current response, Y4. Color coding the simulations first by the input values (upper image), then by the outputs (lower image), we can see a strong correspondence between the *low*, *nominal*, and *high* values of X23 and groups in the Y4 response values. Although the other current response, Y3, is present in the second CCA component, it is more strongly described by the third component, as shown in Fig. 4. The central relationship there is a negative correlation between the input X8 and the output Y3.

6.2 Large Ensemble

The large ensemble is roughly two orders of magnitude larger than the small ensemble, containing 2,641 simulations of a different electrical circuit with 266 input variables and 9 outputs. The outputs for this circuit are more varied than the previous circuit, capturing events and features rather than simple voltages or currents. Slycat easily scales to handle this data, and our largest ensemble to date has contained more than 500,000 simulation runs.

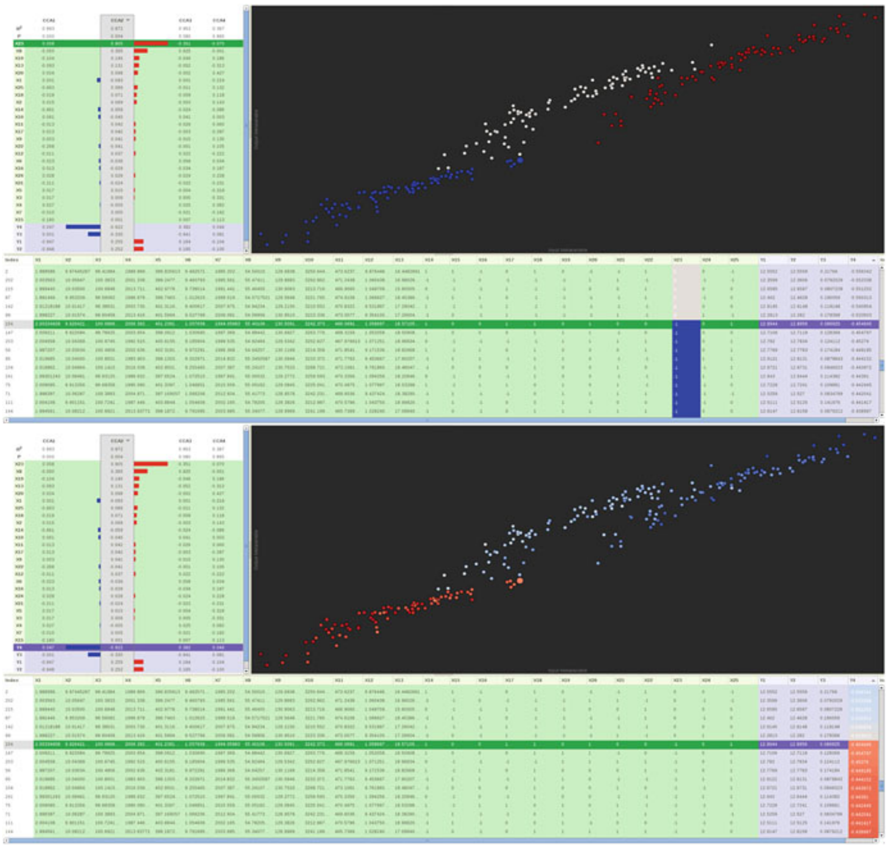


Fig. 3 In the second CCA component, the correlation is predominantly a negative relationship between input X23 and output Y4. Color coding the scatterplot by the values of X23 (*top image*) and Y4 (*bottom image*), we can see a one-to-one correspondence between the *low* values of X23 and the high Y4 responses. The *nominal* and *high* values of X23 combine into a single low-valued group in Y4. In both images, table rows are sorted by Y4 value. Note the selected (enlarged) simulation point near the scatterplot center, which lies on the boundary between the two groups

Given the large number of input variables, an initial analysis goal is to reduce the number of variables required to drive the simulations. Using a process similar to that of the previous section, we are able to cull the number of input variables needed to drive future simulations of this circuit from 266 down to 21.

Finally, we demonstrate how Slycat can be used to answer the third analysis question from Sect. 1: which simulations are anomalous, and in what ways do they differ from the norm? In our initial all-to-all analysis, we noticed four anomalous simulations in the upper part of the scatterplot, the highlighted red points shown in Fig. 5. Distance from the diagonal is a metric for how well the linear correlation found by CCA describes a particular simulation, so these four runs immediately stand out as anomalous. What sets them apart? Since the Y-axis in the scatterplot is

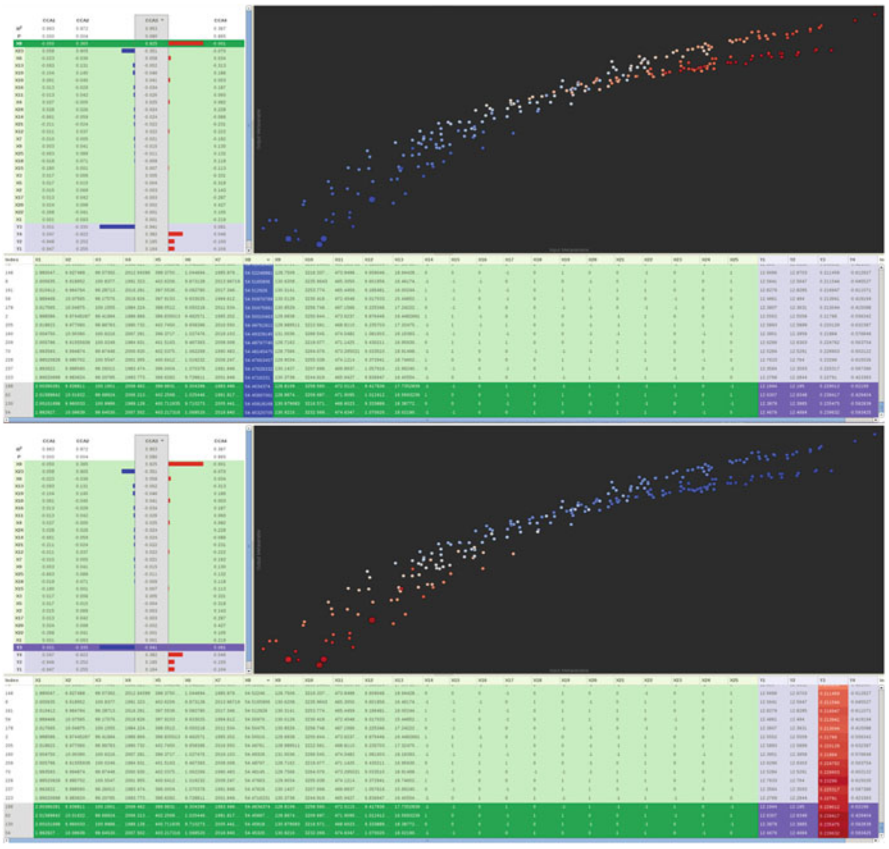
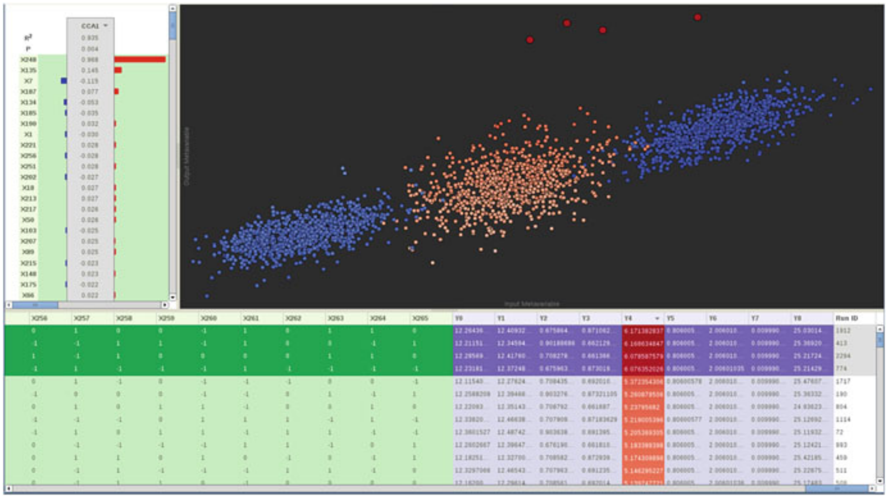


Fig. 4 In the third CCA component, the input parameter X8 is negatively correlated with the current response Y3. Color coding the scatterplot by the values of X8 (*top image*) and Y3 (*bottom image*), we can see the negative relationship between low values (*dark blue*) in the input parameter and high values (*dark red*) in the output. In both images, the table rows are sorted by decreasing X8 values and we have selected the four simulations with the lowest values in X8

a metavariable based on the simulation outputs, vertical placement is a function of output variable values. Interactively switching the color coding between the various outputs, we discover that the Y4 values for these four simulations are at the high end of the scale. Sorting the Y4 values in the table, we see that these four simulations have Y4 values that are distinctly higher than any of the others (notice that they are in red, while the next largest values are in orange).

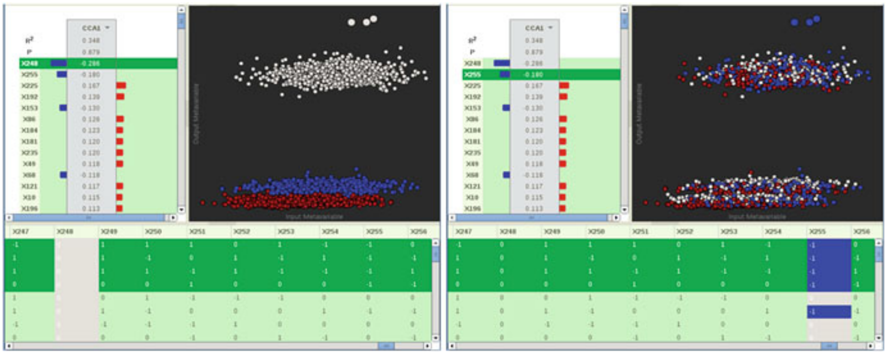
Next we investigate similarities amongst the four simulations' inputs, hypothesizing a common factor leading to the higher responses. We perform a many-to-one CCA analysis between all of the inputs and Y4. The two most highly correlated input variables, X248 and X255, both have identical values for all four simulations. However, each of these variables provides a range of other Y4 responses for the same input values, as seen in Fig. 6, so neither variable in isolation is the



	CCA1
R²	0.935
P	0.004
X248	0.968
X135	0.145
X7	-0.115
X187	0.077
X134	-0.053
X185	-0.035
X190	0.032
X1	-0.030
X221	0.028
X256	-0.028
X251	0.028
X202	-0.027
X18	0.027
X213	0.027
X217	0.026
X50	0.026
X103	-0.025
X207	0.025
X89	0.025
X215	-0.023
X148	0.023
X175	-0.022
X86	0.022

X256	X257	X258	X259	X260	X261	X262	X263	X264	X265	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Run ID
0	1	0	0	-1	1	0	1	1	0	12.26436…	12.40932…	0.675864…	0.871062…	6.171382837	0.806005…	2.006010…	0.009990…	25.03014	1912
-1	-1	1	1	-1	1	0	0	-1	1	12.21151…	12.34584…	0.90188686	0.662129…	6.168634847	0.806005…	2.006010…	0.009990…	25.36920	413
1	-1	1	0	0	0	0	1	1	0	12.28569…	12.41780…	0.708278…	0.661386…	6.079587579	0.806005…	2.006010…	0.009990…	25.21724	2294
-1	1	-1	-1	-1	1	-1	-1	-1	-1	12.23181…	12.37248…	0.675963…	0.873018…	6.076352026	0.806005…	2.00601035	0.009990…	25.21429	774
0	1	-1	0	-1	-1	-1	0	0	-1	12.11540…	12.27624…	0.708435…	0.692016…	5.372354306	0.80600578	2.006010…	0.009990…	25.47607…	1717
-1	0	0	0	-1	-1	0	1	-1	1	12.2588208	12.39468…	0.903276…	0.87321105	5.260879508	0.806005…	2.006010…	0.009990…	25.36332…	190
0	1	0	1	1	-1	0	0	1	0	12.22083…	12.35143…	0.708792…	0.661887…	5.23795482	0.806005…	2.006010…	0.009990…	24.83623…	804
-1	-1	-1	0	1	1	1	-1	1	-1	12.33820…	12.46638…	0.707908…	0.87183629	5.219005396	0.80600577	2.006010…	0.009990…	25.12692…	1114
-1	1	0	1	1	-1	1	1	-1	-1	12.3601527	12.48742…	0.903638…	0.681395…	5.205369305	0.806005…	2.006010…	0.009990…	25.11932…	72
0	-1	-1	-1	-1	1	1	-1	0	1	12.2602667	12.39647…	0.676196…	0.661816…	5.183989398	0.806005…	2.006010…	0.009990…	25.12421…	993
0	1	1	0	1	0	-1	0	-1	1	12.18251…	12.32700…	0.708582…	0.872939…	5.174309898	0.806005…	2.006010…	0.009990…	25.42185…	459
0	-1	1	-1	-1	-1	1	1	-1	0	12.3297066	12.46543…	0.707963…	0.681235…	5.146295227	0.806005…	2.006010…	0.009990…	25.22675…	511
0	-1	1	1	0	-1	0	0	0	0	12.18200…	12.29614…	0.708561…	0.682014…	[illegible]	0.806005…	2.00601036	0.009990…	25.17483…	508

Fig. 5 In the first CCA component of the all-to-all analysis of the large ensemble, anomalous runs (in *red*) are highlighted near the top of the scatterplot. We initially notice them based on their position. The vertical position indicates that the difference between these simulations and the others is based on one of the outputs. *Color coding*, combined with sorting the table, shows that these four simulations have much higher values in Y4 than any of the other simulations. Note that since color encodes value, selection in the scatterplot is shown by increased point radius



	CCA1
R²	0.348
P	0.879
X248	-0.286
X255	-0.180
X225	0.167
X192	0.139
X153	-0.130
X86	0.126
X184	0.123
X181	0.120
X235	0.120
X89	0.118
X68	-0.118
X121	0.117
X18	0.115
X196	0.113

X247	X248	X249	X250	X251	X252	X253	X254	X255	X256
-1	0	1	1	1	0	1	-1	-1	0
1	0	1	-1	0	1	-1	1	-1	-1
1	0	1	1	-1	1	-1	-1	-1	1
0	0	0	0	1	0	0	0	-1	-1
1	0	0	1	-1	-1	-1	0	0	0
1	0	1	-1	0	0	0	1	-1	-1
-1	0	-1	-1	-1	1	0	0	0	0
0	0	0	0	-1	0	1	-1	0	-1

	CCA1
R²	0.348
P	0.879
X248	-0.286
X255	-0.180
X225	0.167
X192	0.139
X153	-0.130
X86	0.126
X184	0.123
X181	0.120
X235	0.120
X89	0.118
X68	-0.118
X121	0.117
X18	0.115
X196	0.113

X247	X248	X249	X250	X251	X252	X253	X254	X255	X256
-1	0	1	1	1	0	1	-1	-1	0
1	0	1	-1	0	1	-1	1	-1	-1
1	0	1	1	-1	1	-1	-1	-1	1
0	0	0	0	1	0	0	0	-1	-1
1	0	0	1	-1	-1	-1	0	0	0
1	0	1	-1	0	0	0	1	-1	-1
-1	0	-1	-1	-1	1	0	0	0	0
0	0	0	0	-1	0	1	-1	0	-1

Fig. 6 We perform an all-to-one analysis to discover which input variables are most highly correlated with Y4, and which could be driving the four anomalous output values. The top two variables, X248 (*left image*) and X255 (*right image*), have identical input values for the four runs, but neither is the sole driver, since both demonstrate the same inputs driving a variety of outputs. The anomalous outputs must stem from a combination of inputs acting in concert

cause. Using the table, we find nine variables that share identical values for all four simulations (X248, X255, X224, X175, X176, X187, X196, X213, and X229). Given X248's strong correlation with Y4, it is definitely involved. Further simulation by our collaborators is needed to isolate which additional input variables are involved, but Slycat allowed us to narrow down the possibilities to a handful of variables.

7 Performance

The use cases that we have presented each took less than a minute to compute. To better characterize the performance of Slycat's CCA implementation, we synthesized a series of test ensembles, varying the number of observations and variables in each, then timed how long it took to upload the data into Slycat and compute a CCA model.

The test hardware included a Mac OSX workstation client that generated the ensembles, uploading them via a 10 Gb ethernet network to a Linux server running the Slycat software with eight 2.4 GHz Quad-Core AMD Opteron processors and 78 GB of RAM. Each test was run three times and the timings averaged. Note that some combinations of observations and variables could not be tested, either because there were too few observations for a given number of variables, or because the CCA computations for the combination would have exceeded the available memory on the server.

During ingestion, the data was uploaded one-variable-at-a-time to avoid exceeding request size limits imposed by HTTP. Reading from left-to-right in Table 1, the ingestion times are almost perfectly linear in the number of variables uploaded. Reading top-to-bottom, we would expect similarly linear behavior as we increase the number of observations; however, the timings are complicated by other factors, including overhead for HTTP request handling, database access, and disk I/O for each upload.

Each test dataset contained an even number of variables, and we configured the CCA computations to split them into equal numbers of inputs and outputs. This configuration ensured that the CCA computation would produce the maximum number of CCA components possible for a given dataset. From Table 2, we see that increasing the number of variables (reading left-to-right) has a larger impact on runtimes than increasing the number of observations (reading top-to-bottom).

Table 1 Data ingestion times (s)

	Variables								
Observations	4	8	16	32	64	128	256	512	1024
10	1.901	2.6							
100	1.897	2.583	4.114	7.019	13.17				
1,000	1.967	2.607	4.13	7.112	13.08	25.96	51.61	104.7	
10,000	2.191	2.998	4.735	8.577	16.13	31.94	63.48	133.8	276.9
100,000	3.238	5.567	7.996	15.43	29.29	57.35	116.9	233.7	
1,000,000	12.53	24.66	48.54	96.02	193	381.7	761.4		

Table 2 CCA compute times (s)

	Variables								
Observations	4	8	16	32	64	128	256	512	1024
10	2.515	2.724							
100	2.516	2.71	3.289	4.489	7.025				
1,000	2.642	2.73	3.326	4.562	7.146	13.82	30.35	85.84	
10,000	2.667	2.827	3.576	4.907	8.023	18.13	74.87	279.4	966
100,000	2.812	3.417	5.952	14.06	35.13	120.4	419.6	1791	
1,000,000	4.727	9.361	26.58	86.04	317.2	1176	5419		

This is consistent with our expectations for the CCA implementation. Since CCA solves an eigenvalue problem based on a covariance matrix, its expected complexity is $O(n) + O(p^3)$, where n is the number of observations and p is the number of variables.

Conclusions and Future Work

We have demonstrated how Slycat meets our design goals, illustrated its utility on two real-life electrical circuit analysis examples of varying scale, and presented performance results for a series of synthetic test ensembles. Slycat's linked views display multiple levels of abstraction from high level ensemble-wide context to deep exploration of relationships at the level of individual simulation inputs and outputs. This combination of iterative analysis and visualization makes CCA more approachable, allowing users to interactively develop and test hypotheses about relationships among variables.

As of this writing, Slycat includes a second analysis type based on time series clustering, which allows us to directly analyze time series data such as the voltage and current waveforms that were reduced to individual output features in our examples. We are also considering a new type of analysis model based on factoring of arbitrary-dimension tensors. The Slycat source code and documentation are freely available under an open source license at https://github.com/sandialabs/slycat, and we welcome collaborators interested in incorporating their own models into Slycat.

Acknowledgements The authors wish to thank Ben Long at Sandia National Laboratories for sharing his work processes, expertise, and data. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. Abdellatif, A.S., El-Rouby, A., Abdelhalim, M.B., Khalil, A.: Hybrid Latin hypercube designs. In: The 7th International Conference on Informatics and Systems (INFOS), pp. 1–5 (2010)
2. Adams, B.M., Ebeida, M.S., Eldred, M.S., Jakeman, J.D., Swiler, L.P., Bohnhoff, W.J., Dalbey, K.R., Eddy, J.P., Hu, K.T., Vigil, D.M., Bauman, L.E., Hough, P.D.: Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.3.1 user's manual. Tech. Rep. SAND2010-2183, Sandia National Laboratories (2013)
3. Anderson, T.W.: An Introduction to Multivariate Statistical Analysis, 3rd edn. Wiley, New York (2003)
4. Borga, M.: Learning multidimensional signal processing. PhD thesis, Linkoping University, Linkoping (1998)
5. Bruckner, S., Moller, T.: Result-driven exploration of simulation parameter spaces for visual effects design. IEEE Trans. Vis. Comput. Graph. **16**(6), 1468–1476 (2010)
6. Coffey, D., Lin, C.L., Erdman, A., Keefe, D.: Design by dragging: An interface for creative forward and inverse design with simulation ensembles. IEEE Trans. Vis. Comput. Graph. **19**(12), 2783–2791 (2013)
7. Computational Engineering International I: URL http://www.ceisoftware.com/ (2013)
8. Degani, A., Shafto, M., Olson, L.: Canonical correlation analysis: Use of composite heliographs for representing multiple patterns. In: Diagram 2006, Lecture Notes in Artificial Intelligence, vol. 4045. Springer, pp. 93–97 (2006)
9. Ge, H., Kirsteins, I., Wang, X.: Does canonical correlation analysis provide reliable information on data correlation in array processing? In: IEEE International Conference on Acoustics, Speech and Signal Processing, 2009 (ICASSP 2009)
10. Hotelling, H.: Relations between two sets of variates. Biometrika **28**(3/4), 321–377 (1936)
11. Hummel, M., Obermaier, H., Garth, C., Joy, K.: Comparative visual analysis of Lagrangian transport in cfd ensembles. IEEE Trans. Vis. Comput. Graph. **19**(12), 2743–2752 (2013)
12. Karhunen, J., Hao, T., Ylipaavalniemi, J.: Finding dependent and independent components from related data sets: A generalized canonical correlation analysis based method. Neurocomputing **113**, 153–167 (2013)
13. Kitware, I.: URL http://www.paraview.org/ (2013)
14. Krzanowski, W.J.: Principles of Multivariate Analysis. A User's Perspective. Oxford University Press, London (1988)
15. Marzban, C.: Model tuning with canonical correlation analysis. Mon Wea Rev (Conditionally Accepted). URL http://faculty.washington.edu/marzban/cca.pdf (2013)
16. Matkovic, K., Gracanin, D., Jelovic, M., Ammer, A., Lez, A., Hauser, H.: Interactive visual analysis of multiple simulation runs using the simulation model view: Understanding and tuning of an electronic unit injector. IEEE Trans. Vis. Comput. Graph. **16**(6), 1449–1457 (2010)
17. Mills, K., Filliben, J., Dabrowski, C.: An efficient sensitivity analysis method for large cloud simulations. In: 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 724–731 (2011)
18. Moreland, K.: Diverging color maps for scientific visualization. In: Advances in Visual Computing, vol. 5876, pp. 92–103. Springer, Berlin (2009)
19. Piringer, H., Pajer, S., Berger, W., Teichmann, H.: Comparative visual analysis of 2d function ensembles. Comput Graph. Forum **31**(3 Pt 3), 1195–1204 (2012)
20. Potter, K., Wilson, A., Bremer, P.T., Williams, D., Doutriaux, C., Pascucci, V., Johhson, C.: Visualization of uncertainty and ensemble data: Exploration of climate modeling and weather forecast data with integrated visus-cdat systems. J. Phys. Conf. Ser. **180**(1), 012, 089 (2009)
21. Potter, K., Wilson, A., Bremer, P.T., Williams, D., Doutriaux, C., Pascucci, V., Johnson, C.: Ensemble-vis: A framework for the statistical visualization of ensemble data. In: IEEE International Conference on Data Mining Workshops, 2009 (ICDMW '09), pp. 233–240 (2009)

22. Sanyal, J., Zhang, S., Dyer, J., Mercer, A., Amburn, P., Moorhead, R.: Noodles: A tool for visualization of numerical weather model ensemble uncertainty. IEEE Trans. Vis. Comput. Graph. **16**(6), 1421–1430 (2010)
23. Shearer, N., Khasawneh, M., Zhang, J., Bowling, S., Rabadi, G.: Sensitivity analysis of a large-scale system dynamics immigration model. In: Systems and Information Engineering Design Symposium (SIEDS), 2010, pp. 78–81. IEEE (2010)
24. Smith, K.M., Banks, D.C., Druckman, N., Beason, K., Hussaini, M.Y.: Clustered ensemble averaging: A technique for visualizing qualitative features of stochastic simulations. J. Comput. Theor. Nanosci. **3**(5), 752–760 (2006)
25. Song, X., Zhao, G.: Sensitivity analysis for a forest growth model: A statistical and time-dependent point of view. In: Plant Growth Modeling, Simulation, Visualization and Applications (PMA), 2012 IEEE 4th International Symposium (2012)
26. Steed, C.A., Ricciuto, D.M., Shipman, G., Smith, B., Thornton, P.E., Wang, D., Shi, X., Williams, D.N.: Big data visual analytics for exploratory earth system simulation analysis. Comput. Geosci. **61**, 71–82 (2013)
27. Sukharev, J., Wang, C., Ma, K.L., Wittenberg, A.: Correlation study of time-varying multivariate climate data sets. In: Visualization Symposium, 2009 (PacificVis '09), IEEE Pacific, pp. 161–168 (2009)
28. Thornquist, H.K., Keiter, E.R., Rajamanickam, S.: Electrical modeling and simulation for stockpile stewardship. XRDS **19**(3), 18–22 (2013)
29. Waser, J., Fuchs, R., Ribicic, H., Schindler, B., Bloschl, G., Groller, M.: World lines. IEEE Trans. Vis. Comput. Graph. **16**(6), 1458–1467 (2010). DOI 10.1109/TVCG.2010.223
30. Waser, J., Ribicic, H., Fuchs, R., Hirsch, C., Schindler, B., Bloschl, G., Groller, M.: Nodes on ropes: A comprehensive data and control flow for steering ensemble simulations. IEEE Trans. Vis. Comput. Graph. **17**(12), 1872–1881 (2011). DOI 10.1109/TVCG.2011.225
31. Wilson, A.T., Potter, K.C.: Toward visual analysis of ensemble data sets. In: Proceedings of the 2009 Workshop on Ultrascale Visualization (UltraVis '09), pp. 48–53. ACM, New York (2009)

Index

J. Bennett et al. (eds.), *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, DOI 10.1007/978-3-662-44900-4