

Marc Christie  
Tsai-Yen Li (Eds.)

LNCS 8698

# Smart Graphics

12th International Symposium, SG 2014  
Taipei, Taiwan, August 27–29, 2014  
Proceedings



 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Alfred Kobsa

*University of California, Irvine, CA, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*TU Dortmund University, Germany*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max Planck Institute for Informatics, Saarbruecken, Germany*

Marc Christie Tsai-Yen Li (Eds.)

# Smart Graphics

12th International Symposium, SG 2014  
Taipei, Taiwan, August 27-29, 2014  
Proceedings



Springer

## Volume Editors

Marc Christie  
Inria/IRISA Rennes  
35042 Rennes Cedex, France  
E-mail: marc.christie@irisa.fr

Tsai-Yen Li  
National Chengchi University  
11605 Taipei City, Taiwan, R.O.C.  
E-mail: li@nccu.edu.tw

ISSN 0302-9743  
ISBN 978-3-319-11649-5  
DOI 10.1007/978-3-319-11650-1  
Springer Cham Heidelberg New York Dordrecht London

e-ISSN 1611-3349  
e-ISBN 978-3-319-11650-1

Library of Congress Control Number: 2014948639

LNCS Sublibrary: SL 6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

© Springer International Publishing Switzerland 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

*Typesetting:* Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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

# Preface

After two years without our yearly Smartgraphics Symposium, it was with great pleasure that we held this 2014 event in the lively city of Taipei.

In the past few years, the emphasis on multidisciplinary approaches in graphics has never been so important, and we are thrilled to see that Smartgraphics has been contributing along this path since the year 2000.

By tackling challenges at the intersection of graphics, AI, HCI, and cognitive sciences, Smartgraphics pushed once again its characteristic interdisciplinary endeavor, and created a place for rich exchanges and great discussions.

This year's event was held during August 27–29, 2014, at the National Chengchi University in Taipei, Taiwan. The Organizing Committee gathered a lively and broad program, with four sessions, presenting altogether 14 full papers, 4 short papers, and a number of posters, demonstrations, and art pieces. Sessions were within the typical Smartgraphics scope with a first session on data visualization, a second session on sketching and multi-touch interaction, a third on aesthetics and smart tools for artists, and a last one on smart tools for 3D contents.

As usual in our event, a demo session was organized in which researchers and practitioners can present their tools, triggering rich discussions and valuable feedback through an open and informal setup.

Smartgraphics was honored to host two very inspiring researchers to share their contributions and views on graphics and interdisciplinary endeavor during two very exciting keynotes: Takeo Igarashi from the University of Tokyo, and Karan Singh from the University of Toronto. The Organizing Committee is extremely grateful for their time.

At last, we would like to thank all authors and speakers for their contributions, the reviewers for their careful work, and the Program Committee for selecting and ordering contributions for the final program. Special thanks go to Helen Wu, who did a tremendous job in assisting the organization of the event, setting up the details, taking close care of the deadlines, and gathering and compiling the proceedings.

August 2014

Marc Christie  
Tsai-Yen Li

# Organization

Smart Graphics 2014 was organized by National Chengchi University.

## Organizing Committee

### Conference Chairs

Marc Christie	IRISA/Inria Rennes, France
Tsai-Yen Li	National Chengchi University, Taiwan

### Steering Committee

Andreas Butz	University of Munich, Germany
Antonio Krueger	University of Muenster, Germany
Brian Fisher	Simon Fraser University, Canada
Marc Christie	IRISA/Inria Rennes, France
Patrick Olivier	University of Newcastle Upon Tyne, UK

### Art Exhibition Chair

Neng-Hao Yu	National Chengchi University, Taiwan
-------------	--------------------------------------

### Poster Chair

Hui-yin Wu	Inria Rennes/IRISA, France
------------	----------------------------

### Local Organizing Committee

Chun-Feng Liao	National Chengchi University, Taiwan
Hao-Chuan Wang	National Tsing-Hua University, Taiwan
Ming-Te Chi	National Chengchi University, Taiwan
Neng-Hao Yu	National Chengchi University, Taiwan
Tsai-Yen Li	National Chengchi University, Taiwan
Wen-Hung Liao	National Chengchi University, Taiwan

## Program Committee

Benjamin Walther-Franks	University of Bremen, Germany
Bernhard Preim	University of Magdeburg, Germany
Bing-Yu Chen	National Taiwan University, Taiwan
Christian Jacquemin	LIMSI-CNRS, France

## VIII Organization

Elisabeth André	University of Augsburg, Germany
Hiroshi Hosobe	Hosei University, Japan
Hung-Hsuan Huang	Ritsumeikan University, Japan
Lutz Dickmann	Bremen University, Germany
Mateu Sbert	University of Girona, Spain
Roberto Ranon	University of Udine, Italy
Roberto Therón	University of Salamanca, Spain
Shigeo Takahashi	University of Tokyo, Japan
Shigeru Owada	Sony CSL, Japan
Tevfik Metin Sezgin	Koc University, Turkey
Thomas Rist	University of Applied Sciences, Germany
Tong-Yee Lee	National Cheng-Kung University, Taiwan
Tracy Hammond	Texas A&M University, USA
Tsvi Kuflik	University of Haifa, Israel
Wen-Hung Liao	National Chengchi University, Taiwan
William Bares	College of Charleston, USA
Yaxi Chen	Southwest University for Nationalities, China

## Sponsoring Institutions

The Smart Graphics Symposium was held in cooperation with Inria/IRISA Rennes Bretagne Atlantique, France, and sponsored by the Ministry of Science and Technology of the Republic of China, Springer, ACM SIGGRAPH Taipei Chapter, and Intel.

# Table of Contents

## Data Visualization

- Storytelling via Navigation: A Novel Approach to Animation for Scientific Visualization . . . . . 1  
*Isaac Liao, Wei-Hsien Hsu, and Kwan-Liu Ma*
- A Deep Dive into Decades of Baseball's Recorded Statistics . . . . . 15  
*Antonio G. Losada, Roberto Theron, and María Vaquero*

## Sketching and Multi-Touch Interaction

- A Sketch-Based Generation System for Oriental Cloud Pattern Design . . . . . 27  
*Ming-Te Chi, Chen-Chi Hu, and Yu-Jyun Jhan*
- D-Sweep: Using Profile Snapping for 3D Object Extraction from Single Image . . . . . 39  
*Pan Hu, Hongming Cai, and Fenglin Bu*
- Fixit: A 3D Jigsaw Puzzle Game Using Multi-touch Gestures . . . . . 51  
*Yi-Hsiang Lo, Che-Chun Hsu, Hsin-Yin Chang, Wen-Yao Kung, Yen-Chun Lee, and Ruen-Rone Lee*
- Towards an Intelligent Framework for Pressure-Based 3D Curve Drawing . . . . . 63  
*Chan-Yet Lai and Nordin Zakaria*

## Aesthetics and Smart Tools for Artists

- Analysis of Visual Elements in Logo Design . . . . . 73  
*Wen-Hung Liao and Po-Ming Chen*
- Controllable and Real-Time Reproducible Perlin Noise . . . . . 86  
*Wei-Chien Cheng, Wen-Chieh Lin, and Yi-Jheng Huang*
- Emotional Line: Showing Emotions through the Sharpness, Height, Width and Speed of a Series of Concatenated Digital Shifting Curves . . . . . 98  
*Jesús Ibáñez and Carlos Delgado-Mata*
- Application Friendly Voxelization on GPU by Geometry Splitting . . . . . 112  
*Zhuopeng Zhang and Shigeo Morishima*



## Smart Tools for 3D Contents

A Pattern-Based Tool for Creating Virtual Cinematography in Interactive Storytelling .....	121
<i>Pei-Chun Lai, Hui-Yin Wu, Cunka Sanokho, Marc Christie, and Tsai-Yen Li</i>	
Semantically Consistent Hierarchical Decomposition of Virtual Urban Environments .....	133
<i>Carl-Johan Jorgensen and Fabrice Lamarche</i>	
A New Way to Model Physics-Based Topology Transformations: Splitting MAT .....	146
<i>Saman Kalantari, Annie Luciani, and Nicolas Castagné</i>	
3D Artistic Face Transformation with Identity Preservation .....	154
<i>Tanasai Sucontphunt</i>	

## Posters

Real Time Visualization of Large Data Using Clustered Projections ....	166
<i>Pranav D. Bagur</i>	
Crowdsourcing 3D Motion Reconstruction .....	170
<i>Yueh-Tung Chen, Cheng-Hsien Han, Hao-Wei Jeng, and Hao-Chuan Wang</i>	
Understanding Which Graph Depictions Are Best for Viewers .....	174
<i>Johanne Christensen, Ju Hee Bae, Ben Watson, and Micheal Rappa</i>	
A New Modeling Pipeline for Physics-Based Topological Discontinuities: The DYNAMe Process .....	178
<i>Annie Luciani, Nicolas Castagné, Philippe Meseure, Xavier Skapin, Saman Kalantari, and Emmanuelle Darles</i>	
<b>Author Index</b> .....	183

# Storytelling via Navigation: A Novel Approach to Animation for Scientific Visualization

Isaac Liao, Wei-Hsien Hsu, and Kwan-Liu Ma

University of California-Davis, One Shields Avenue, Davis, California, USA

**Abstract.** In scientific visualization, volume rendering is commonly used to show three-dimensional, time-varying data sets. To understand the spatial structure of volume data sets and how they change over time, animation is often necessary. Most current visualization tools use conventional keyframe-based interfaces to create animations, which can be tedious and time-consuming. We present a new, semi-automatic approach to creating animations of volume data based on a user's interaction history as they navigate through a data set and adjust rendering parameters. Through a user study, we show that this new method requires significantly less time and perceived effort on the part of users compared to keyframe-based approaches, while still generating animations of comparable quality.

**Keywords:** Scientific visualization, volume visualization, animation.

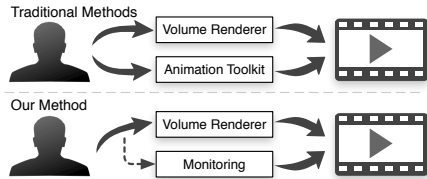
## 1 Introduction

Scientific visualizations usually deal with spatial, three-dimensional data sets. Animation is a natural way to provide an overview of this type of data. The growing popularity of video abstracts for summarizing journal publications and conference proceedings, along with the ubiquity and accessibility of streaming, high-definition video websites, means that the ability to produce effective, comprehensible video summaries of scientific work is becoming increasingly important.

However, creating animations using modern scientific visualization software is often tedious and time-consuming. Animation packages in most current visualization tools use conventional keyframe-based methods, which require training and practice to use effectively. In addition, certain types of animation sequences, such as smoothly curved camera paths, are difficult or impossible to create using keyframes. Thus, scientists who wish to present interesting findings or illustrate complex phenomena are forced to become proficient with an unfamiliar system that may not help them achieve their goals.

To help address this problem, we present a method for automatically generating animations based on user interaction history. As a scientist explores a data set using a visualization tool they are already familiar with, our system records all of the user's interactions, including viewpoint movements, transfer function modifications, and time spent at each view. This exploration history is then analyzed and used to create an animation that highlights the relevant features of the data set, without any additional work on the part of the user.

**Fig. 1.** Our method vs. the conventional keyframe method of animation. When using keyframes, users must set volume rendering parameters as well as viewpoints and transition timings. Our method uses navigation as input for generating animations, eliminating an extra step.



To evaluate the usefulness and effectiveness of our approach, we conducted a user study. Our method was compared to a traditional keyframe-based approach in terms of time spent and perceived benefit. We found that our method requires much less time and perceived effort on the part of users, and generates animations of comparable quality to those created using keyframes.

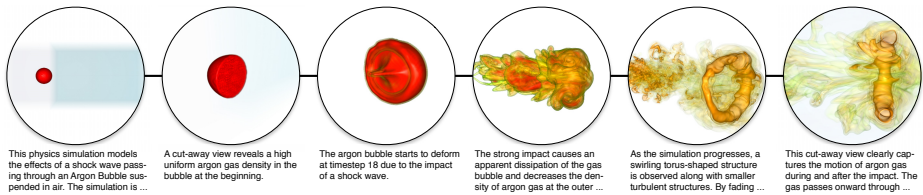
## 2 Methods

To be able to automate the generation of volume visualization animations, we should first examine how such animations are manually created. Intrinsicly, an animation for volume visualization is a sequence of images with rendering parameters that change over time. A typical way to control these changing parameters is by using keyframes. A user specifies a number of keyframes with different configurations at different time points, and the parameters for the frames in-between are interpolated from the nearest preceding and subsequent keyframes. This method requires a great deal of user effort to decide which parameter(s) to keyframe, where to position each keyframe in the animation timeline, whether to insert pauses between consecutive animated transitions, etc. We present a new method that allows users to create animations without needing to learn additional tools (Figure 1).

### 2.1 Overview

A user exploring a volume data set to find interesting features goes through a process where they gradually make sense of the data. Similarly, to tell a story effectively, the order and rate at which pieces of information are revealed must be carefully paced. Animations of volume data are no exception, and would likely benefit from using a storytelling approach to present a scientist’s reasoning process. Thus, the manner in which a user navigates a volume data set as they are exploring it may serve as a reasonable starting point for considering how to present the data. As an additional benefit, users would be freed from having to learn how to use animation tools. Since our target users are domain experts who are already familiar with and use volume rendering daily, this approach effectively eliminates the additional training and effort that is currently required to create animations.

In order to capture a user’s reasoning process in exploring and navigating a volume data set, we employ the following procedure: 1) A user explores a volume data set as he/she usually does. 2) During this exploration, the user identifies



**Fig. 2.** Story views for the Argon bubble data set. While users identify and annotate important views using a volume renderer, their interaction history is recorded and used to generate an animation automatically.

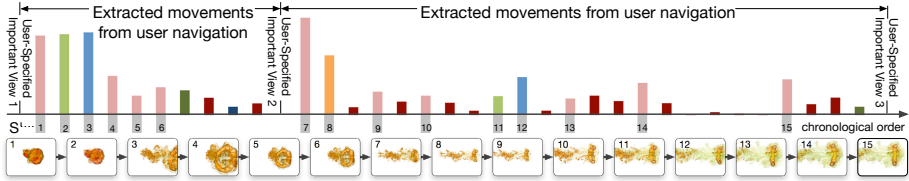
important views and gives short descriptions for each of the views. An example is shown in Figure 2. Using this storytelling-like procedure, we hope to enable users to generate animations with minimum additional effort.

We must carefully consider user behavior while navigating volume data to better understand and utilize this information. We have the following observations: First, a user can only change one parameter at a time in most cases. Furthermore, users repeatedly adjust the same parameter as interesting views are approached. Although this behavior is mostly due to the limitations of the renderer UI, this observation implies that users are going through a typical sense making process. Second, users tend to go through as much trial-and-error as possible before nailing down important views. This reflects the main workflow of an interactive rendering system, in which a user iteratively explores data, gleans insights, and identifies important features. This essential capability of volume rendering systems should not be abandoned while developing new functionality. These two behaviors alternate as users adjust different parameter settings, causing difficulty when creating smooth animations. Thus, we cannot simply replay user navigation history to create an explanatory animation.

Based on these observations, there are three critical tasks that must be carefully addressed to produce fluid animations. Given a subset of a user’s navigation records, we need to 1) remove redundant movements, 2) extract representative rendering settings, and 3) create smooth and pleasant transitions. In other words, our method should be able to determine “keyframes” based on the user’s navigation history. Furthermore, these automatically extracted “keyframes” must be concise enough to eliminate unnecessary trial-and-error patterns, yet coherent enough to animate the data in a natural way.

## 2.2 Technical Details

In this study, we consider the four most common types of parameters in volume visualization: camera position, timestep, opacities, and clipping planes. We further divide camera movements into three subtypes – trackball rotation, zooming, and look-at repositioning – because these three camera movements often have different meanings in a sense-making process: rotation signifies an examination around an area, zooming suggests a detailed inquiry, and look-at repositioning indicates an attention transfer from one point of interest to another. Based on



**Fig. 3.** An example timeline of changes in viewing and rendering parameters (MOVES). Red, blue, green, and orange: changes in camera position, timestep, opacity, and clipping planes, respectively. The height of each bar corresponds to the magnitude of each MOVE. MOVES are thresholded and considered either a substantial change **S** (light colors) or a fine-tuning change **F** (dark colors). Gray bars and corresponding snapshots (bottom) show the resulting set,  $\mathbf{S}'$ .

these observations, we make four assumptions that contributed to the development of our automatic animation method:

1. Users can only change one type of parameter at a time.
2. Users usually find the most direct way to travel between important views.
3. A large or sudden change implies a confident move.
4. Users tend to slowly fine-tune parameters as they approach an important view.

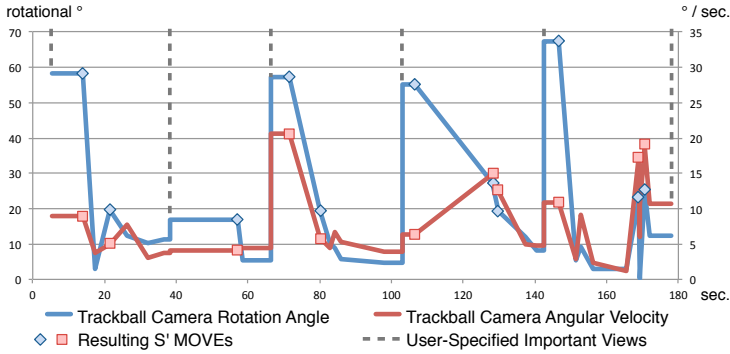
A user exploration record  $\mathbb{R}$  contains  $n$  important views, including the first view. We can divide the record into  $n - 1$  pieces, each of which consists of the user’s exploration from one important view to the next. Based on assumption 1, we can further divide a record piece into  $m_i$  moves, where each move consists of a change in one parameter. The entire record then becomes  $\text{VIEW}_1 \rightarrow \text{MOVE}_{1,1} \rightarrow \text{MOVE}_{1,2} \rightarrow \dots \rightarrow \text{MOVE}_{1,m_1} \rightarrow \text{VIEW}_2 \rightarrow \dots \rightarrow \text{VIEW}_n$ .

We categorize MOVES into two groups: substantial (**S**) and fine-tuning (**F**). The displacement  $x$  between the first and last frames of a MOVE in the parameter space is an important clue, and the velocity  $v = x/t$ , where  $t$  is the duration of a MOVE, can also be used to categorize a MOVE. We formulate this classification as:

$$\text{Group}(\text{MOVE}) = \begin{cases} \mathbf{S} & x \geq \mathbf{X} \text{ or } v \geq \mathbf{V} \\ \mathbf{F} & x < \mathbf{X} \text{ and } v < \mathbf{V} \end{cases} \quad (1)$$

where  $\mathbf{X}$  and  $\mathbf{V}$  are thresholds for the displacement and velocity, respectively. Note that the displacement of a MOVE and the two thresholds are dependent on the type of the changing parameter of the MOVE. For camera viewpoints, we use the displacement of the camera look-at position and the angular differences of the viewing and up vectors; for the other parameters, we adopt the Euclidean distances of the timesteps, clipping distances, and opacity values.

The first occurrence of a type of MOVE between two consecutive VIEWs is put into a new group,  $\mathbf{S}'$ . Then, we iteratively merge every **F** MOVE with the first preceding **S** MOVE of the same type, and add these merged **S** MOVES to  $\mathbf{S}'$ . We combine successive camera movement subtypes in  $\mathbf{S}'$  to create flowing

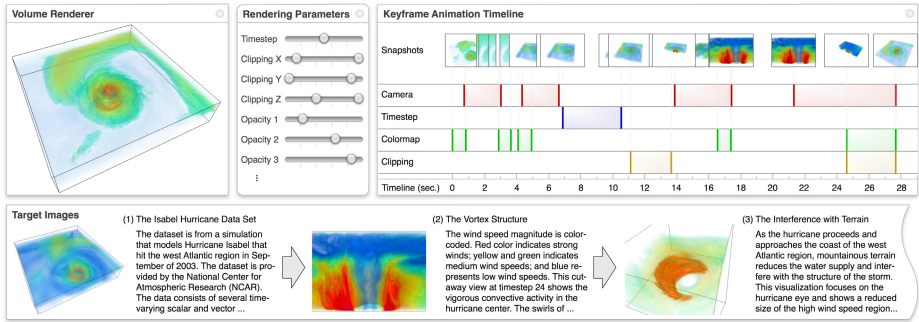


**Fig. 4.** Displacements over time for a single type of MOVE (trackball camera rotation). The vertical axes represent rotational angle (degrees) and angular velocity (degrees per second). MOVEs whose displacements exceeded thresholds are assigned to  $S'$  (shown with markers). In this example, the user tends to make large changes after each important view, followed by smaller movements as they approach the next important view.

camera transitions. The parameter settings of the  $S'$  MOVEs are then used as keyframes to generate an animation. The duration of each transition is determined according to the displacement of the corresponding  $S'$  MOVE. Finally, we insert an appropriate pause at each important view for enhanced emphasis.

Figure 3 shows the portion of user navigation history corresponding to the last three important views of the Argon bubble data set in Figure 2. The top bar chart shows the underlying MOVEs of the user navigation in chronological order. We use colors to distinguish different types of MOVEs: red, blue, green, and orange represent movements in camera position, timestep, opacities, and clipping planes, respectively. The height of each MOVE depicts its displacement. Since there is no uniform scale for displacements between different types of MOVEs, and their thresholds also vary, we use shade to indicate how a MOVE has been categorized: light colors are  $S$  MOVEs, whereas dark colors are  $F$  MOVEs. The middle grey bars in the figure show where each  $S'$  MOVE is located, and the bottom snapshots are generated based on the parameter settings at the end point of each  $S'$  MOVE.

By analyzing the displacements and velocities of extracted MOVEs, we can see that users tend to make large movements at the beginning of transitions away from important views, and finer movements as they approach the next important view. This is consistent with assumptions 3 and 4; that is, users are confident in moving away from important views because they have already marked them, while they approach potential important views more carefully. Figure 4 shows a representative series of trackball rotation MOVEs extracted from one user navigation record collected in our user study. The markers along the two lines in the figure denote the locations of  $S'$  MOVEs. Notice that the overall displacement of a MOVE is often the determining factor in whether it is assigned to  $S'$ ; however, our method can also capture sudden, small changes and include them in the resulting animation.



**Fig. 5.** Volume rendering system overview. Top row, left to right: a volume rendering view, an interface for adjusting rendering parameters, and a keyframe animation tool used for comparison in our user study. Bottom row: annotated target images for the training data set.

### 3 User Study

To test the usability of our approach, we compare it with a keyframe-based animation tool. We choose to use a tool that provides comprehensive keyframe functionality, allowing users to create keyframes for each individual parameter. The GUI of the tool, as shown in the top right in Figure 5, allows users to move keyframes to any time point by simply dragging them along the animation timeline. Since users are able to control the magnitude of each individual parameter and the timecourse of changes in parameters, this interface provides powerful support for creating complex animations.

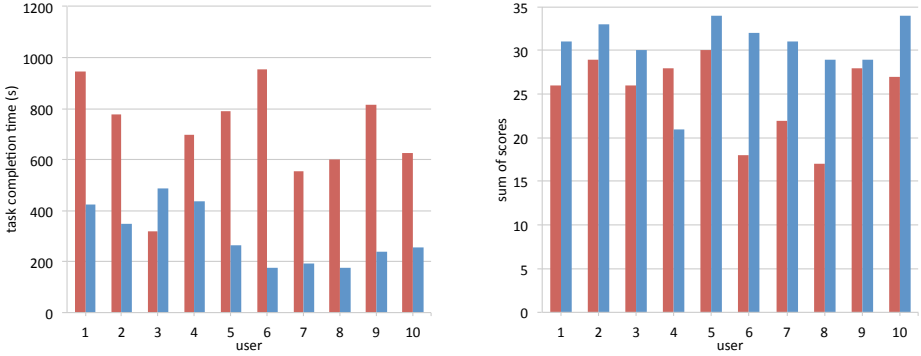
#### 3.1 Procedure

To familiarize users with the system’s user interface and navigation controls, users were trained before each task using a training data set. For this study, we used a hurricane time-varying volume data set as the training data set (Figure 5).

Specifically, for each user, the training proceeded as follows:

1. The user was familiarized with the user interface using the training data set.
2. The user was shown annotated target images, and asked to replicate them.
3. The user completed an interactive tutorial on how to use the keyframe system.
4. The user watched a video demonstrating the use of the keyframe system.
5. The user was shown annotated target images for the training data set, and asked to create an animation including them using the keyframe system.

After training, each user completed two main tasks. During each task, users were shown a set of target images of the task data set. For this study, we used the Argon bubble time-varying volume data set (Figure 2). To help users comprehend the data set and formulate a narrative, target images were annotated with brief descriptions.



**Fig. 6.** Left: Task completion times (seconds) for each user, separated by task. Right: Sum of questionnaire scores for each user, separated by task. Red: Keyframe task. Blue: Navigation task.

In the *keyframe task*, users created an animation of the data set using a conventional keyframe-based method. Users were instructed to include the target images in the movie.

In the *navigation task*, users navigated through the data set without specifying keyframes, and an animation was automatically generated based on a record of their exploration. Again, users were instructed to re-create views of the data set matching the target images.

All users completed both tasks. To account for learning effects, the order of tasks was counterbalanced. Users were randomly assigned to either a *keyframe-first group*, or a *navigation-first group*.

After completing both main tasks, users were shown both the animation they created during the keyframe task, as well as an automatically-generated animation based on their inputs during the navigation task. Finally, users answered a questionnaire comparing their preferences for the keyframe-based approach versus the navigation-based approach.

First, participants were familiarized with the data set and the user interface. Second, they were allowed to freely explore the data set for 5-10 minutes. Third, they were familiarized with the keyframe-based animation controls. Fourth, they were asked to create a 30-second video summarizing the salient features of the data set. Finally, they were shown an automatically-generated video based on their user interaction history during the free exploration period, and answered a questionnaire comparing the keyframe-based approach and the interaction history-based approach.

## 4 Results

A total of 10 users participated in the study. 5 users were assigned to the keyframe-first group, and 5 users were assigned to the navigation-first group.



**Table 1.** User questionnaire. Users responded “strongly disagree”, “disagree”, “neither agree nor disagree”, “agree”, or “strongly agree”. (During the user study, the navigation-based method was referred to as the “tracking” method.)

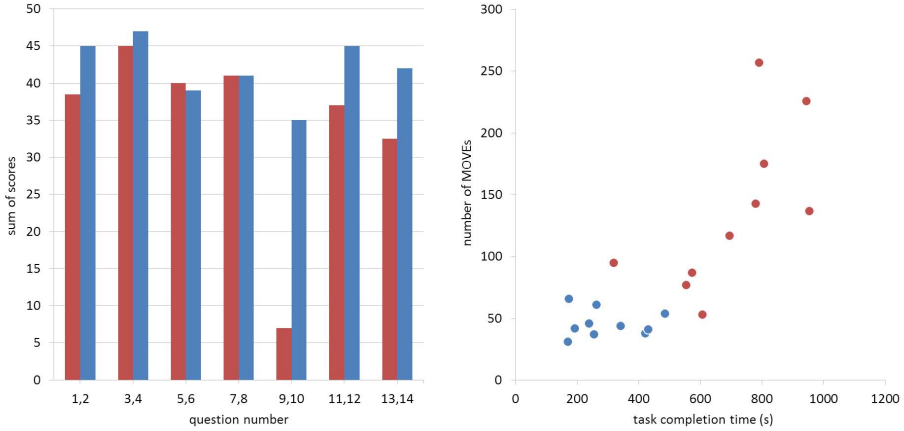
1. The keyframe tool is easy to understand and easy to use.
2. The tracking method is easy to understand and easy to use.
3. The video that was created using keyframes shows all relevant features of the dataset.
4. The automatically-generated video shows all relevant features of the dataset.
5. In the video that was created using keyframes, the transitions between features are smooth and natural.
6. In the automatically-generated video, the transitions between features are smooth and natural.
7. The video that was created using keyframes is pleasant to watch.
8. The automatically-generated video is pleasant to watch.
9. The keyframe method took a lot of work.
10. The tracking method took a lot of work.
11. The amount of work that it took to make a video using the keyframe method is worth it in the end.
12. The amount of work that it took to make a video using the tracking method is worth it in the end.
13. I would like to use the keyframe method to make a video in the future.
14. I would like to use the tracking method to make a video in the future.
15. I consider myself an advanced computer user.
16. I am familiar with volume rendering.
17. I am familiar with keyframe-based animation.

Users consisted of computer science graduate students familiar with animation and visualization.

First, we compared task completion times. Our null hypothesis was that both tasks would take the same amount of time. Since each subject performed both tasks, and since we make no assumptions about directionality of putative differences, we used a paired, two-tailed  $t$ -test to measure significance. The mean completion times were 300.8 seconds for the navigation task, and 707.9 seconds for the keyframe task (Figure 6, left). A paired two-tailed  $t$ -test gives  $p \leq 0.00057$ . This, we are able to reject the null hypothesis, and conclude that users completed the navigation task significantly faster than the keyframe task.

Next, we examined the effect of task order on completion time. Null hypothesis 1 was that users took the same amount of time to complete the keyframe task regardless of whether they were in the keyframe-first group or the navigation-first group. Null hypothesis 2 was that users took the same amount of time to complete the navigation task regardless of whether they were in the keyframe-first group or the navigation-first group. Since we are comparing completion times of different users, and do not assume equal variance between populations, we use an unpaired unequal-variance  $t$ -test to measure significance. Users in the keyframe-first group performed the navigation task in a mean completion time of 208.2 seconds, while users in the navigation-first group completed the same task in a mean of 393.4 seconds (unpaired unequal-variance two-tailed  $t$ -test:  $p \leq 0.006$ ). The mean average keyframe task completion times for the keyframe-first group and the navigation-first group were 710.3 seconds and 705.5 seconds, respectively (unpaired unequal-variance two-tailed  $t$ -test:  $p \leq 0.97$ ). Thus, we are able to reject null hypothesis 1, but not null hypothesis 2, and conclude that there was a significant learning effect for the navigation task, but not for the keyframe task.

The first 14 questions of the questionnaire measured user opinions about the keyframe-based method and the navigation-based method. Questions were paired, such that the same question was asked about both the keyframe method



**Fig. 7.** Left: Sum of scores for each question. Questions 9 and 10 have their scores inverted because in those cases, a lower score is better. Error bars not shown because survey results were treated as ordinal data for analysis purposes. Right: Completion time (x-axis, seconds) vs. number of MOVES (y-axis) for each user and task. Red: Keyframe task. Blue: Navigation task.

and the navigation method. To provide an overview of each user’s responses, we totaled each user’s score for each task (Figure 6, right). Since there were 7 questions per task, each scored 1-5, the possible range of total scores for each task is 7-35. For questions 9 and 10, “The keyframe/navigation method took a lot of work”, a lower score is better, so we inverted the score before adding to the total. Additionally, to provide an overview of responses to each question, we totaled each question’s score across users (10 users, scored 1-5 for each question, possible total range 10-50; Figure 7, left). Again, the scores for questions 9 and 10 were inverted.

Since users responded using a non-quantitative “strongly disagree / disagree / neutral / agree / strongly agree” axis, we consider their responses ordinal/ranked data rather than quantitative data for analysis purposes. Thus, we use the Wilcoxon signed-rank test, which is a non-parametric, paired-difference test. For the first 8 questions, which compared the two methods in terms of ease of use, ability to capture relevant features, smoothness of transitions, and aesthetics of the generated animation, the test statistic  $W$  calculated from differences in user opinions did not meet critical values for significance (Wilcoxon signed-rank test). For questions 9 and 10, users rated the keyframe-based method as taking significantly more work than the navigation-based method (Wilcoxon signed-rank test;  $W = 55$ ;  $N_r = 10$ ;  $z = 2.78$ ; two-tailed  $p \leq 0.0054$ ). For questions 11 and 12, users rated the animation created using the navigation-based method as more worth the effort than the animation created using the keyframe-based method (Wilcoxon signed-rank test;  $W = -36$ ;  $N_r = 9$ ; two-tailed  $p \leq 0.05$ ). Finally, in questions 13 and 14, users showed greater preference for the navigation-based

method than the keyframe-based method for creating animations in the future (Wilcoxon signed-rank test;  $W = -30$ ;  $N_r = 8$ ; two-tailed  $p \leq 0.05$ ).

To determine whether task order affected user opinions about the animation methods and the resulting animations, we examined responses of the keyframe-first group in comparison to those of the navigation-first group. For each of the first 14 questions of the questionnaire, the 5 keyframe-first user responses were compared to the 5 navigation-first user responses using a Mann-Whitney  $U$  test. Thus, sample sizes were always  $n_1 = n_2 = 5$ . For these sample sizes,  $U \leq 2$  or  $U \geq 23$  represents a significant difference at  $p \leq 0.05$ . However, across the first 14 questions,  $U$  ranged from 5 to 20, indicating that task order was not associated with a significant difference in user responses.

To check for disparities or outliers in the recruitment pool, the last 3 questions asked users to rate their familiarity with computers, volume rendering, and keyframe-based animation. All users responded “agree” or “strongly agree” when prompted whether they consider themselves advanced computer users, and whether they are familiar with volume rendering. All users but one responded “agree” or “strongly agree” when asked whether they are familiar with keyframe-based animation. One user responded “neither agree nor disagree”; however, this user’s completion times for the keyframe and navigation tasks were not significantly different from those of other users.

## 5 Discussion

The most striking finding from the user study is that, although creating an animation using keyframes took over twice as long as automatic generation, users rated the two resulting videos similarly in terms of completeness, smoothness of transitions, and aesthetic pleasantness. This result shows that it is possible to provide support for creating animations in a way that reduces demands on user time without making sacrifices in animation quality.

Since the amount of time that a task takes to complete is not necessarily representative of its perceived workload, we then examine user ratings of the animation methods. Not only did users take longer to complete the keyframe task, but they also rated the keyframe task as taking significantly more work than the navigation-based method. Perhaps most significantly, when asked whether the amount of work invested was “worth it in the end”, users rated the navigation-based method significantly higher than the keyframe-based method. This result shows that the navigation method effectively reduces the perceived workload of users, while at the same time increasing the perceived benefit of invested work, in comparison to the keyframe method.

To determine whether there was a significant learning effect, we compare task completion times across tasks and groups. The significantly reduced task completion time of the navigation task for users in the keyframe-first group indicates that practice with the system during the keyframe task is transferable to the navigation task. In addition, there was no significant difference in keyframe task completion time across groups, indicating that practice with the system during

the navigation task does not transfer to the keyframe task. This is perhaps unsurprising, since the navigation task can be considered a subset of the keyframe task. However, this result indicates that the keyframe task takes longer to complete due to the added complexity of the keyframe system, and in a way that does not improve as users gain familiarity with the underlying navigation controls.

Although keyframes proved to be more time-consuming than animation based on user navigation, one question that merits further investigation is whether the degree of control afforded by keyframes is worth the time cost to users. Figure 7 (right) shows that users who took longer to complete tasks also tended to make more MOVEs, rather than spending additional time passively planning or considering. On one hand, keyframes require more user time, but provide precise, predictable control over the final animation, while on the other hand, navigation-based animation takes less user time, but gives users little control over the final output animation. While our questionnaire results show that users would prefer to use the navigation-based system over the keyframe-based system overall, it is unclear whether this preference is affected positively or negatively by the lesser degree of control afforded by the navigation-based system.

Since users were not already familiar with the types of data presented in the user study, it was necessary to provide them with annotated target images to help them understand features in the data set. In a more realistic scenario, the user would be a domain expert who is already familiar with the subject of the data set (e.g., time-varying volume visualizations of hurricanes), and would not be provided with annotated target images. Instead, they would be exploring a data set, looking for interesting features. It would be interesting to see if high-quality animations can be generated using only this meaningful exploration interaction history. Future work will focus on gathering domain experts, presenting new data sets to them, and generating animations based on the interaction history of their initial exploration.

## 6 Related Work

Animation is a versatile and powerful tool for transitioning between views, highlighting data, showing complex spatial structures, and understanding time-varying phenomena. Heer and Robertson survey a number of animated transitions in statistical data graphics and propose design principles for creating effective infographics transitions [1]. Woodring and Shen use animation to highlight data in static visualizations, leveraging knowledge of motion perception [2]. In volume visualization, animation is also used to create comprehensible transitions between focus and context [3,4], or to reveal uncertainty in the data [5].

Current fundamental methods of generating animations include keyframes, motion capture, and physics. After decades of advances in animation, keyframes remains one of the basic tools for creating motion effects, having become widely used in computer animation and video editing [6]. Modern character animation uses technologies such as motion capture to rapidly create realistic animations[7]. Movies and computer games take advantage of physics-based animation to reduce the amount of user input required to animate complex scenes [6].

Although keyframes have become widely adopted for creating animations, they can be time-consuming and difficult to use. As a result, high-quality animations in scientific visualization are often handcrafted by professional animators using sophisticated, specialized software. Scientific research organizations, such as NASA’s Goddard Scientific Visualization Studio ([svs.gsfc.nasa.gov](http://svs.gsfc.nasa.gov)), NOAA’s Environmental Visualization Lab ([www.nvvl.noaa.gov](http://www.nvvl.noaa.gov)), and UCAR’s Visualization & Enabling Technologies ([www.vets.ucar.edu/vg](http://www.vets.ucar.edu/vg)), create animations for educational purposes and to tell complex scientific stories. In volume visualization, domain experts use animations to show interesting features and important findings in volume data sets. However, existing visualization packages such as EnSight ([ensight.com](http://ensight.com)) provide limited keyframe animation functionality. VisIt ([visit.llnl.gov](http://visit.llnl.gov)) and ParaView ([www.paraview.org](http://www.paraview.org)) offer timeline-based animation editors, but require users to manipulate each and every keyframe. Therefore, domain experts also often seek professional help to create high-quality animations of volume visualizations.

Keyframes have been used in volume visualization to provide tools for making animations. Moltedo and Morigi developed a system that combines keyframe animation and a graphical user interface to produce animations of scientific data [8]. More recently, improvements to keyframe-based animation in volume visualization has been the focus of several projects. Akiba et al. enhanced the keyframe-based approach by classifying common motion types into a list and providing them as a set of predefined motion templates that could be used to facilitate the authoring of keyframe animations [9]. Mühler and Preim presented a technique that allows users to define “keystates” while exploring a medical data set [10]. These keystates can then be applied to similar data sets, or used to generate animations. Wu et al. developed a palette-style volume visualization system which allows users to quickly explore different configurations of rendering parameters and to rapidly create animations by connecting points on a palette wheel [11]. A more automatic approach was proposed by Yu et al. for generating animations of time-varying data visualizations [12]. Finally, Hsu et al. introduced a semi-automatic method for camera motion planning in volume data animation [13].

Previous work has been done on automatic keyframe extraction. In character animation, several methods extract specific movements from motion capture data for modification or reuse. A typical approach is to construct a graph for different types of walk cycles and generate synthesized motion accordingly [14]. Work has also been done in analyzing and classifying motion capture data so that users can easily understand the underlying semantic meanings of movements [15] or to provide an alternative interface for controlling animated objects [16]. Keyframe extraction is also commonly seen in video editing and video compression [17]. More advanced, automatic approaches have been studied in recent years for video summarization [18] and fast browsing[19]. These techniques inspired us to consider how keyframes could be extracted from user interaction during navigation of a volume visualization.

In recent years, the topic of formalized storytelling has emerged as a means of helping users communicate, understand, and interpret datasets. Wohlfart and

Hauser applied the concept of storytelling to volume visualization [20], treating the creation of a visualization as a story, and presenting sequential story nodes using an animated interactive system. Lu and Chen produced an interactive storyboard for presenting time-varying data sets [21]. Yu et al. represented events from time-varying data as tree-like hierarchical structures for the purpose of creating narratives [12]. Finally, Diepenbrock et al. developed an image-based navigation approach for collision-free flying through internal volume structures [22]. Similarly, we would like to enable scientists and other domain experts to communicate findings and tell stories about data using animation with minimal effort.

## 7 Summary

To summarize, we have devised a semi-automatic method of generating animations based on user inputs during navigation through a 3D volume data set. We conducted a pilot user study comparing this method to traditional keyframe-based animation. Task completion times indicate that users need to spend only half the time creating animations using our method as compared to keyframes, and subjective user ratings indicate that the animations generated by our method are of comparable quality to those created using keyframes.

As future work, it would be interesting to see whether these findings are replicable in completely unsupervised situations – that is, without predefined target images. Answering this question will require working with domain experts, who are already accustomed to seeing and analyzing the types of data sets that will be presented. In particular, it may be possible to generate a meaningful overview animation of a data set using only the interaction history of a user exploring a data set. This would lead to a method of generating expository animations of data sets without any additional time or effort on the part of users, beyond the initial exploration.

## References

1. Heer, J., Robertson, G.: Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics* 13, 1240–1247 (2007)
2. Woodring, J., Shen, H.-W.: Incorporating Highlighting Animations into Static Visualizations. In: *Proceedings of SPIE Electronic Imaging*, vol. 6495, p. 649503 (2007)
3. Wu, Y., Qu, H., Zhou, H., Chan, M.-Y.: Focus + Context Visualization with Animation. In: Chang, L.-W., Lie, W.-N. (eds.) *PSIVT 2006*. LNCS, vol. 4319, pp. 1293–1302. Springer, Heidelberg (2006)
4. Sikachev, P., Rautek, P., Bruckner, S., Gröller, M.E.: Dynamic Focus+Context for Volume Rendering. In: *Proceedings of Vision, Modeling and Visualization*, pp. 331–338 (2010)
5. Lundstrom, C., Ljung, P., Persson, A., Ynnerman, A.: Uncertainty Visualization in Medical Volume Rendering Using Probabilistic Animation. *IEEE Transactions on Visualization and Computer Graphics* 13, 1648–1655 (2007)

6. Parent, R.: *Computer Animation: Algorithms and Techniques*, 3rd edn. Morgan Kaufmann (2012)
7. Shiratori, T., Park, H.S., Sigal, L., Sheikh, Y., Hodgins, J.K.: Motion capture from body-mounted cameras. *ACM Transactions on Graphics* 30, 1 (2011)
8. Moltedo, L., Morigi, S.: ANIMA: An Interactive Tool for Scientific Data Animation. *Computer Graphics Forum* 12, 277–288 (1993)
9. Akiba, H., Wang, C., Ma, K.-L.: AniViz: A Template-Based Animation Tool for Volume Visualization. *IEEE Computer Graphics and Applications* 30, 61–71 (2010)
10. Mühler, K., Preim, B.: Reusable Visualizations and Animations for Surgery Planning. *Computer Graphics Forum* 29, 1103–1112 (2010)
11. Wu, Y., Xu, A., Chan, M.-Y., Qu, H., Guo, P.: Palette-style volume visualization. In: *Proceedings of the Sixth Eurographics/IEEE VGTC Conference on Volume Graphics, VG 2007*, pp. 33–40. Eurographics Association, Aire-la-Ville (2007)
12. Yu, L., Lu, A., Ribarsky, W., Chen, W.: Automatic Animation for Time-Varying Data Visualization. *Computer Graphics Forum* 29, 2271–2280 (2010)
13. Hsu, W.-H., Zhang, Y., Ma, K.-L.: A Multi-Criteria Approach to Camera Motion Design for Volume Data Animation. *IEEE Transactions on Visualization and Computer Graphics* 19, 2792–2801 (2013)
14. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. *ACM Transactions on Graphics* 21, 473–482 (2002)
15. Müller, M., Baak, A., Seidel, H.-P.: Efficient and robust annotation of motion capture data. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2009*, pp. 17–26 (2009)
16. Yamane, K., Ariki, Y., Hodgins, J.: Animating non-humanoid characters with human motion data. In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2010*, pp. 169–178 (2010)
17. Girsensohn, A., Boreczky, J., Wilcox, L.: Keyframe-based user interfaces for digital video. *Computer* 34, 61–67 (2001)
18. Truong, B.T., Venkatesh, S.: Video Abstraction: A Systematic Review and Classification. *ACM Transactions on Multimedia Computing, Communications, and Applications* 3, 3:1–3:37 (2007)
19. Correa, C.D., Ma, K.-L.: Dynamic Video Narratives. *ACM Transactions on Graphics* 29, 88:1–88:9 (2010)
20. Wohlfart, M., Hauser, H.: Story telling for presentation in volume visualization. In: *Proceedings of the 9th Joint Eurographics/IEEE VGTC Conference on Visualization, EUROVIS 2007*, pp. 91–98. Eurographics Association (2007)
21. Lu, A., Shen, H.-W.: Interactive Storyboard for Overall Time-Varying Data Visualization. In: *PacificVIS 2008: Proceedings of IEEE Pacific Visualization Symposium*, pp. 143–150 (2008)
22. Diepenbrock, S., Ropinski, T., Hinrichs, K.: Context-aware volume navigation. In: *2011 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 11–18 (2011)

# A Deep Dive into Decades of Baseball's Recorded Statistics

Antonio G. Losada, Roberto Theron, and María Vaquero

Department of Computer Science and Automation,  
University of Salamanca. Plaza de los Caídos s/n, Salamanca, Spain  
`{theron, alosada, mvaquero}@usal.es`

**Abstract.** Sports data are usually stored in databases and made available as plain tables that, in the best of the cases, allow the users to sort them by a given column. Although this technique is the most used to show the present state of a competition or its final outcome, it does not provide all of the information an analyst may require. Hence, this work aims to provide a new method of visualizing full seasons of baseball by means of a heatmap that holds every game of the season in it. Taking advantage of filtering and interaction, we explore the possibility of overlapping seasons to provide a deeper analysis of teams' results over selected time spans.

**Keywords:** visual analysis, sport visualization, baseball.

## 1 Introduction

The growth that baseball has experienced in terms of available statistical analysis methods has not been followed by a wide use of information visualization techniques, which is in its early days in the sports' field, leaving fans and analysts without suitable ways of exploring all of the data that baseball generates day in day out.

Regarding the baseball game, the official Major League Baseball (MLB) schedule establishes that every team must play 162 games (sometimes 161 or 163 on special cases). A full season with 30 teams playing it, results in some overwhelming five thousand games to analyze. If the dataset involves more than one season, that numbers would rocket to huge amounts of data, making it impossible to analyze by exploring and sorting plain tables. Some websites cover huge loads of statistics, but to the best of our knowledge, none of them provide advanced methods to facilitate the comprehension of the data they show. While the stored data alone can be useful to find information and answer simple questions, it is not sufficient in the case of more complex inquiries. The cognitive load that would require the act of remembering big sets of information organized in tables claims for a visualization that ease this task.

This paper aims to provide a new method to visualize data out of a basic set of statistics generated with each match played by a couple of teams. Despite having developed a whole application focused on the performance analysis of



individual teams and their players from 1900 to 2013, this paper is devoted to present a visualization that enables the analysis of any number of seasons and gives an example that covers the 21st century of American Baseball.

## 2 Related Work

The earliest visualizations regarding baseball came from Graham Wills [21], who proposed a series of plots with multiple charts to represent baseball statistics and group data. Cox and Stasko [6] provided new methods of showing game and players information allowing to sort and filter it by multiple categories. The Chernoff Faces [4][9] have been also applied to baseball visualization [15]. Edward Tufte [19], a pioneer in information visualization, has also applied his research to baseball and Ric Werme [20] widely introduced the *running graphs* showing teams' season progressions. Other authors, such as Ben Fry, have made available some concrete visualization about baseball as can be seen in [5]. Most websites have been using little charts to show information, like [1] and [2], where some heatmaps can be found. The way of showing rankings have changed over time and some systems such as Slope Graphs [11][18], Rank Charts and the R2S2 proposal [12] have left the plain ranking tables obsolete although they remain as the primal method of visualizing this information. Table Lens [3] and MagicLens [14] have provided methods for exploring large sets of data without requiring large scale devices due to its organization, focus and zooming. Other techniques, such as TreeMaps [8], have been widely used to represent information in the field of sports as SportVis [6] does. To represent knockout tournament developments, D. Tan et al. developed a system as an alternative to the normally used brackets called AdaptiviTree [16], which saved screen space showing the same information much more compacted. In others sports, like basketball, K. Goldsberry had an impact on the NBA with his CourtVision [7], which uses a heat map and size-variable allocated to the position of the attempt squares to represent the shooting performance of pro players in terms of the numbers of shoots shot and converted. Hockey has presented also some evolutions, with SnapShot [13] being one of the most interesting as it represents radial heat maps that represent information regarding shot positioning. Innovative techniques based on motion or video recording have also been proposed for basketball, allowing to track the player movements via GPS or analyzing video to extract data regarding rebounds [17][10], while some general purpose systems as Tableau<sup>1</sup> or Spotfire<sup>2</sup> have been used to produce interactive visual representations of baseball data.

## 3 Problem Description

Our work was aimed at providing the possibility of analyzing big sets of data regarding matches played by teams in one or multiple seasons. Due to the amount

---

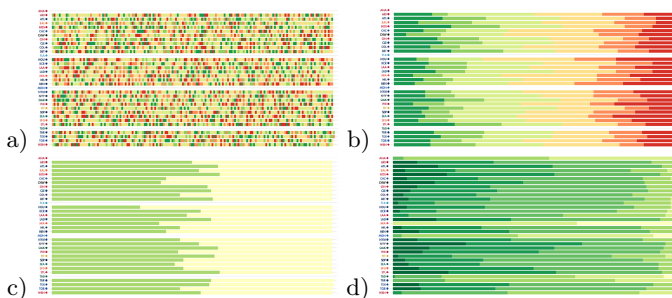
<sup>1</sup> [www.tableausoftware.com](http://www.tableausoftware.com)

<sup>2</sup> [spotfire.tibco.com](http://spotfire.tibco.com)

of games played in a season, another concern was the required screen space to represent them. Considering the dataset presented in this paper, we had to deal with 68,956 games from 2000 to 2013. Our approach to the problem was to base our representation on run differences or victories for each match. A season can be easily conveyed as a matrix in which each row represents a team and each column a game. This, multiplied by the number of teams participating in the season, generates a full matrix that represents every game played during that year, therefore, a full season. To highlight the differences between results, a color scheme can be used, marking the games won or lost by more or less points. To deal with several seasons, by computing average season results for the selected time span, seasonal matrices can be overlapped, providing the analysts with a whole vision of the team’s performance. This approach is combined with multiple views and detailed charts, creating an all-in-one interactive exploration of seasons, teams and players.

## 4 System Description

StatClub uses visualization techniques to expose patterns and allow people to find relationships between the represented data and its actual meaning, regarding many different aspects of the game. In this paper, we focus on the heatmap analysis we have enabled, which we describe in terms of data management, visual encoding and interaction.



**Fig. 1.** Same season with different horizontal match order. (a) chronological; (b) run difference; (c) victories in one season; and (d) victories in several seasons.

### 4.1 Data Management

StatClub retrieves and stores data from the Internet. The stored data contains information about the almost sixty-nine thousands games played between 2000 and 2013 in the MLB, while the full system allows the user to explore any team from 1900 to 2013 retrieving the detailed data on the fly. Every team that played during our time span (00-13) is prepared to keep a list of all the games it played

during the fourteen years (or less in case of disappeared or recently introduced teams). Once the data is set up, the matrix is created based on the last season (2013 in our case) ordering the teams alphabetically on the vertical axis and matches by game number (from 1 to 162) on the horizontal one (see Figure 1.a).

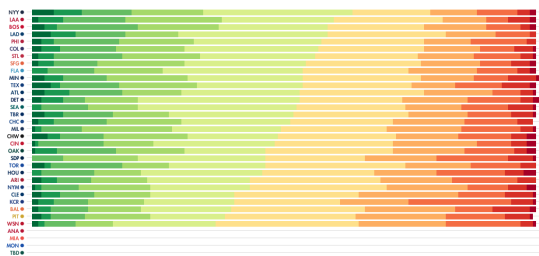
## 4.2 Visual Encoding and Interaction

Besides the heatmap, information relative to the block on focus, which may contain one or more matches depending on the number of selected seasons, is also shown. In the main area, every team that played at least one game from 2000 to 2013 is shown in the vertical list at the left color-coded by franchise. The row next to the team's name represents the matches played by the team in the season selected, and the color of each block varies depending on the computed run difference (runs scored minus the opponent runs). The assigned color depends on the level of analysis applied (discussed later).

When more than one season are analyzed, the heatmap will convey the overlapped performance during the selected time span. Therefore, the colors will change to represent the average run difference of the two or more matches that share the same game number in the overlapped seasons. If the horizontal order is set to represent the victories obtained, accumulated in one or more seasons, the colors will vary only from green to yellow, since the fewest victories the team could have earned is zero. In order to reduce the cognitive load, transitions are used every time the heatmap is sorted.

Figure 1 shows how these three horizontal sorting methods differ. While the default sorting (Figure 1.a) contains all the match information in a way that can be useful to find patterns in certain moments of the season, the other two sorting methods provide further useful information. Sorting by run difference (Figure 1.b) the user can understand which teams surely dominated the league and which ones ended the season with a poor register but losing a lot of games by one or two points in what can be seen or understood as bad luck. The third sorting method (Figure 1.c) removes peripheral information (in the case of one season) and centres the visualization in the difference between won and lost games. The last case (Figure 1.d), which is similar to the previous one but extending the data from one to fourteen seasons, represents the number of victories each team has gotten in a given game number, showing the most powerful teams through history.

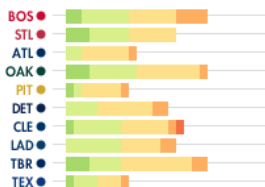
Another way of organizing data is vertically (see Figure 2). While the default order is alphabetical (due to its simplicity and ease of understanding), teams can be sorted by wins (in one or several seasons). Figure 2 conveys the relationship between a vertical organization, where the most winning teams are atop of the list, and a horizontal one, where the wins are at the left (in contrast with the losses, at the right). Note how the four teams without games (and therefore, no registered wins) in the selected season are positioned down the list. Also, note the two blocks that go out of the limits of the heatmap at the right side, conveying the fact that those teams played one extra game during the season. It does not necessarily mean that the block out of the matrix limits is the



**Fig. 2.** Vertical organization of teams by number of wins in a season

extra game (163), since the team could have won it and therefore it would be on the left (green) side of the corresponding row.

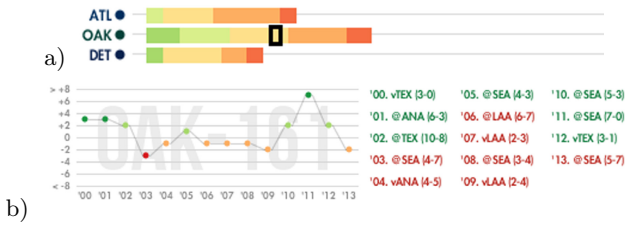
One of the last aspects of our visual encoding is related to the used color gradient, whose full spectrum can be seen in Figure 2, as it shows the deepest level of analysis (11 color-coded bins for match run difference, from  $< -10$  to  $> 10$ ).



**Fig. 3.** 2013 Season top 10 winning teams and the matches they played where they scored between 5 and 10 runs and allowed between 6 and 10

Beyond horizontal and vertical sorting and the time span selection, we have enabled several filters. By combining heatmaps with filtering (see Figure 3), we provide a simple yet powerful analysis tool that permits the user to answer questions such as how many runs a team needs to win a game no matter how many the rival scores (discussed later as a case study) or who has more luck by winning more games by one or two runs.

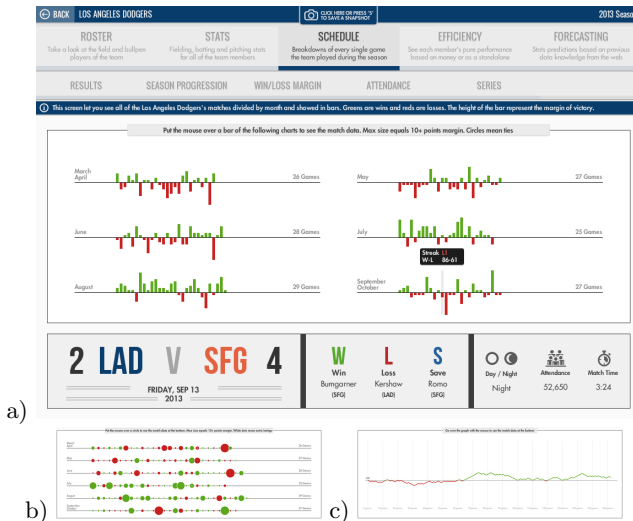
Another interactive possibility of the application is related to the match blocks, providing the users with detailed information. At the user's demand, any block can be selected, making it possible to track its evolution over time thanks to the animated transitions. As shown in Figure 4.a, the selected block is highlighted by a thick black border (the 161st game, in the figure). Figure 4.b details all the matches that the selected block holds. The number of matches contained per block will vary depending on the number of seasons being overlapped in a given moment. On the background both the team's name and the game number under analysis are shown. Matches are represented as little circles in a position that conveys the year when the games took place and its outcome.



**Fig. 4.** (a) Highlighted match block. (b) Block's matches represented in a time vs. run-difference graph.

Details can be found at the right side list, which shows the results for each one of the games represented in the graph, including the opponent and if they were disputed at home or as a visitant.

The last interactive option is accessible via the team names list at the left of the matrix. Hovering over the team's name the corresponding row is highlighted. When only one season is shown in the graph due to the applied filters, the teams become clickable and therefore available for a more accurate and individualized analysis.



**Fig. 5.** (a) Season results view to allow a game by game exploration. (b) Win/Loss margins expressed in a horizontal flow. (c) Results progression over a complete season.

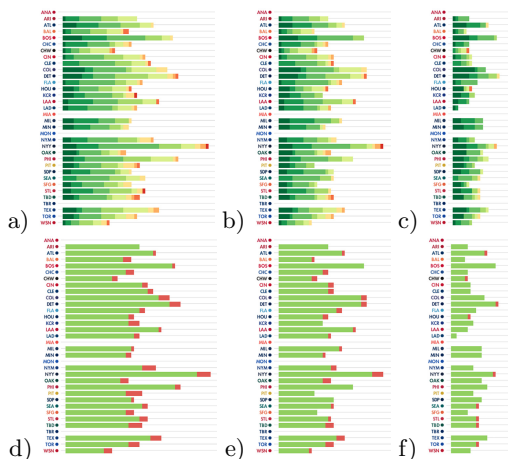
Figure 5 shows an example of the kind of detailed information available, where the 2013 season of Los Angeles Dodgers was selected. Figure 5.a permits the exploration of the schedule: through a combination of monthly bar charts (matches), which represent the margin of victory for each game in green (over

the axis, wins) or red (under the axis, losses) following the scheme used in our heatmap. For further information, each bar can be explored to see the outcome of the game, the winning and loser pitchers (and the one who got the save) and some miscellaneous information related to the time of the day, the attendance or the match duration. The little black box that pops up when hovering a game is related to the streak the team is maintaining up to that match and the season W-L register in terms of won and lost games up to the selected date. Figure 5.b and 5.c show two other available views related to the team’s results.

## 5 Case Studies

In this section we present two case studies showing the potential of our approach to answer complex questions and foster a deep analysis on the available data.

We begin aiming at general questions in the first case to then deepen into more concrete queries directly related with real baseball historic seasons or moments for which we try to find meanings by exploring our visualization.



**Fig. 6.** Matches where the analyzed team scored 8 or more (a, d), 9 or more (b, e) and 11 or more (c, f) runs. The level of analysis can be changed from 11 channels (a, b, c) to 3 channels (d, e, f).

### 5.1 Forecasting Win Probability Based on Scored Runs

The user wants to find answers to questions like how many runs a team needs to win a game no matter the runs allowed, or what is the maximum number of runs a team can endurance without automatically losing the game if it gets to score a certain number of them.

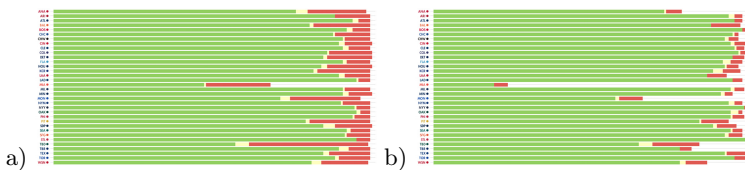
The user starts by finding a relationship between scored runs and won matches in a single season (2007, to be precise) to discover how many runs guaranteed a victory in that season. To do this, the user filters the matches to show only the ones where a certain number of runs were scored by each team. After that, results are sorted by run difference, thus grouping all of the remaining matches at the left of the screen. Figure 6 shows the results of filtering by scored runs, specifically by 8 or more (a), 9 or more (b) and 11 or more (c) runs scored. To clarify the results, the same filter has been applied but with less depth of detail, so the wins and losses can be easily distinguished (d, e, f).

As expected, the more restrictive the filter, the less matches meet the study requirements. While there were a lot of games played where a team scored 8 or more points and less with 9 or more, the results show less than half the matches (compared to Figure 6.a) racking up 11 or more points, which is infrequent.

Now the user focuses on the lost games. Figures 6.d-e (8-9 or more runs) still contain a small amount of losses, but Figure 6.f, where the team got to score at least 11 runs, make the losses almost negligible: only 9 games (red blocks) were lost by the 30 analyzed teams. So the user reach the conclusion that a team which scored 11 runs in 2007 had a probability of winning higher than 95%.

To find the concrete number of runs which guaranteed the win with a hundred percent of probability the user raises the filter up to fifteen runs, which is kind of exaggerated but has its explanation in a game played between the Chicago White Sox and Minnesota Twins that ended on a 14-20 defeat of the former, disturbing the results.

Finally, the user wanted to find the answer not for a season but for the fourteen seasons played in the 21st century of baseball. The results are shown in Figure 7, which covers 2000-2013 and represents the games where teams scored 4 or more (a) and 7 or more (b) runs. The amount of games, given the long span, has increased, showing that much lower scores than the 11 runs needed in 2007 can carry a great number of wins along with them. Four runs (Figure 7.a) may be seen as a little of a risk but seven (Figure 7.b) are more than enough to bring the win home in most of the cases.



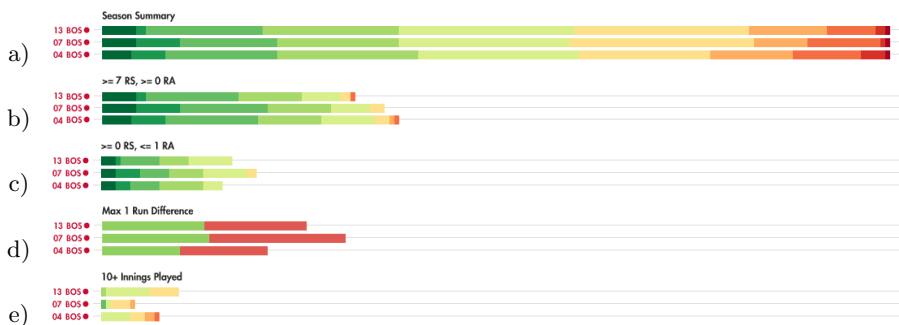
**Fig. 7.** Matches where teams scored 4 or more (a) and 7 or more (b) runs during a span of fourteen years (2000 to 2013)

## 5.2 Boston Red Sox' World Series Races

Our second study aims to compare three historic moments of the same franchise, the Boston Red Sox. The Bostonian team has won three World Series in the last

fourteen years, the most by any team in that period, and the user will be looking to compare the three teams (from now on BOS04, BOS07 and BOS13) to gain more knowledge about their strengths and weaknesses.

First, the user would inspect the full season results of each team. This will give an early glimpse of which teams performed better, won more games by more points or suffered the biggest defeats, giving way to inferring team's attributes such as offence or defence. To get the most out of this visualization the user decides to have the games sorted by run difference in each season with the greatest depth of detail (see Figure 8.a): the overall results are very similar, differing in just two victories among the three teams. Getting into more detail thanks to the selected depth scheme, we can appreciate how BOS13 and BOS07 teams deployed a better defensive performance over the course of the season (less defeats by 3 or more points, orange-red blocks) while they all performed quite similar in terms of offensive production. The only difference in that department, albeit being almost trivial, is the low big-score production of BOS13.



**Fig. 8.** Five visualizations of data from the '13, '07 and '04 Boston Red Sox, which ended winning the World Series those seasons

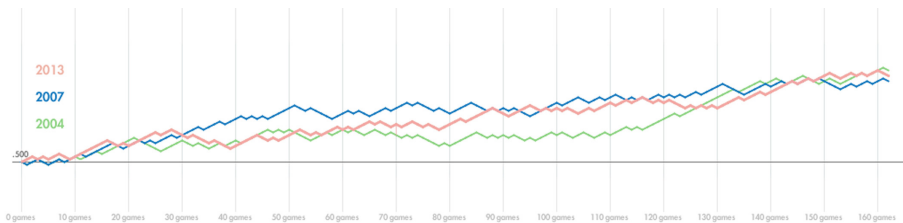
Now that the starting point suggests that BOS13 and BOS07 teams seem to have great defenses, with BOS04 having more problems, the user can keep exploring. Offensive and defensive production analysis seems appropriate, which will endorse the initial user's thoughts on the situation. In Figure 8.b, the user decides to show only the matches where the Red Sox scored at least 7 runs, which is more than a good score and should (as a conclusion of the first case study) be enough to win the game most of the times. The green blocks confirm the big offensive production that BOS04 put up as they won more than the games BOS13 played at all and almost the same number of BOS07 total matches (in both cases with at least 7 runs, counting both wins and losses). Furthermore, it is proved that BOS04 lacked defense ability, as they played for the most losses given these circumstances. Also, it can be deduced that the league is turning into a low-scoring game as can be seen in Figure 8. which shows that for each year there are less games with great scores.



Moving onto pure defense, the user analyzes only those games where the teams allowed one or none runs. Figure 8.c shows how the pattern and conclusions found earlier keep appearing. BOS07 is the team which held its rivals to one run the most times (longest bar) and BOS04 the one which achieved it in the least cases (shortest bar). Attending to the offensive production details given by color code applied, BOS13 shows again that it is not atop of the three in terms of scoring (dark green).

Another factor to take into account is the luck each team had during the season. Although it cannot be measured and may be seen as something outside the game, baseball is a sport where luck is important and can be present in almost every game, as a bad call by an umpire can end up deciding a ball game. The matches where luck impacts the most is in close ones, so the user can apply this principle and filter the games to show only the ones that ended with just a one run difference (see Figure 8.d). Very similar results stand out. If the three bars were normalized, the proportion would be close and almost equal for every team. Thus, BOS04 can be considered as a kind of an all-or-nothing team (big wins or big losses), hence low one run difference games. BOS13 and mostly BOS07 used to hold the rivals more and that could have led to more wins in close matches if luck had been favourable to them.

Next, the user wanted to analyze only games that went into overtime, namely, that went further than the official 9 innings (Figure 8.e). BOS04 again lost games by wide margins in the overtime, something that does not happen usually and could be related to defensive issues. BOS07 had little chances in the overtime but ended losing most of those games, which may stand for its inability of performing under pressure or in close games situations.



**Fig. 9.** Season progression for the three Bostonian teams analyzed in this case study

Finally, and taking advantage of the remaining views of our tool, the user generated a line chart (see Figure 9) that shows the progression of each of the three teams, in order to assess how their seasons went in terms of streaks of wins and losses. BOS07 sustained the most regular progression, making it the most reliable team in a season despite being the one with the worst record of the three (two wins less). The red line, which stands for BOS13, shows four distinguishable periods (games 20 to 40, 40 to 80, 80 to 130 and 130 to 162); each of them starts with a long streak of wins that makes the team go up in the

rankings and the last one is long enough to make a huge impact in the overall season. On the contrary, BOS04 proves to be not so reliable by showing a low winning percentage for almost three quarters of the season, relying its success in an outburst at the end of the season that rocketed its ranking to the best of the three analyzed teams.

## 6 Discussion and Future Work

Baseball is one of the sports in which more statistics are recorded. Starting from a basic approach based on games results, we have proposed a new visualization system to analyze every game of a single or multiple seasons in the same screen, in a simple yet powerful way not explored to date.

Matrices provide a good structure for sorting based on multiple criteria and it favours filtering; in combination with the heatmap technique it allows to highlight concrete events and discover patterns with ease. This added to the interactive possibilities the visualization contains makes it a strong tool for game analysis that provides an in-depth experience as it is embedded in a multiple-views baseball statistics analysis system.

Although we have applied our tool to a dataset of fourteen years of baseball history, it could be easily expanded. Furthermore, our approach could be transferred to other team-based sports such as football or basketball since they use the same schedule oriented organization and the matches are based in points and, therefore, they can be measured by point difference. Besides that, other future works may include adding more filtering and ordering options to provide more flexibility to the user, including one-team historic matrices (row by row rather than overlapped) to enable easy season comparisons or improving the interaction with the blocks creating new methods to show information about the matches.

## 7 Conclusion

We have proposed a visualization that makes use of both a matrix and a heatmap combined to create a tool that allows the exploration of thousands of matches by just representing blocks on the screen in a reduced space.

The discussed case studies show how our simple yet powerful visualization can foster the exploration of vast amounts of data and be of great help in gaining knowledge related to dynamics occurring during periods of time expanding several decades. Our proposal is valid for multiple sports, and even it can be transferred to other event-based problems.

**Acknowledgments.** This research was supported by the Spanish MINECO grant FI2010-16234.

## References

1. Allen, D.: 45 up, 45 down (July 2009), [www.fangraphs.com/blogs/45-up-45-down/](http://www.fangraphs.com/blogs/45-up-45-down/)
2. Allen, D.: One of the game's stranger hitters (2009), [baseballanalysts.com/archives/2009/08](http://baseballanalysts.com/archives/2009/08)
3. Bier, E.A., Stone, M.C., Pier, K., Buxton, W., DeRose, T.D.: Toolglass and magic lenses: the see-through interface. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, pp. 73–80. ACM (1993)
4. Chernoff, H.: The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association* 68(342), 361–368 (1973)
5. Fry, B.: Salary vs. performance (2005), <http://benfry.com/salaryper/>
6. Cox, A., Stasko, J.: Sportsvis: Discovering meaning in sports statistics through information visualization. In: Compendium of Symposium on Information Visualization, pp. 114–115. Citeseer (2006)
7. Goldsberry, K.: Courtvision: New visual and spatial analytics for the NBA. In: MIT Sloan Sports Analytics Conference (2012)
8. Johnson, B., Shneiderman, B.: Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In: Proceedings of the IEEE Conference on Visualization, Visualization 1991, pp. 284–291. IEEE (1991)
9. Lee, M.D., Reilly, R.E., Butavicius, M.E.: An empirical evaluation of chernoff faces, star glyphs, and spatial visualizations for binary data. In: Proceedings of the Asia-Pacific Symposium on Information Visualisation, vol. 24, pp. 1–10. Australian Computer Society, Inc. (2003)
10. Maheswaran, R., Chang, Y.H., Henehan, A., Danesis, S.: Deconstructing the rebound with optical tracking data. In: MIT Sloan Sports Analytics Conference (2012)
11. Park, C.: Slopegraphs, <http://charliepark.org/slopegraphs/>
12. Perin, C., Vernier, F., et al.: R2s2: a hybrid technique to visualize sport ranking evolution. In: What's the Score? The 1st Workshop on Sports Data Visualization (2013)
13. Pileggi, H., Stolper, C.D., Boyle, J.M., Stasko, J.T.: Snapshot: Visualization to propel ice hockey analytics. *IEEE Transactions on Visualization and Computer Graphics* 18(12), 2819–2828 (2012)
14. Rao, R., Card, S.K.: The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 318–322. ACM (1994)
15. Reisner, A.: What's the matter with chernoff faces? (2006), <http://www.alexreisner.com/baseball/chernoff-faces>
16. Tan, D.S., Smith, G., Lee, B., Robertson, G.G.: Adaptivtree: Adaptive tree visualization for tournament-style brackets. *IEEE Transactions on Visualization and Computer Graphics* 13(6), 1113–1120 (2007)
17. Theron, R., Casares, L.: Visual analysis of time-motion in basketball games. In: Taylor, R., Boulanger, P., Krüger, A., Olivier, P. (eds.) *Smart Graphics*. LNCS, vol. 6133, pp. 196–207. Springer, Heidelberg (2010)
18. Tufte, E.: Open forum on slopegraphs, [http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg\\_id=0003nk](http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0003nk)
19. Tufte, E.R.: Beautiful evidence, vol. 1. Graphics Press Cheshire, CT (2006)
20. Werme, R.: Runnings - baseball standings on the run (2014), <http://wermenh.com/runnings.html>
21. Wills, G.J.: Selection: 524,288 ways to say “this is interesting”. In: Proceedings of the IEEE Symposium on Information Visualization 1996, pp. 54–60. IEEE (1996)

# A Sketch-Based Generation System for Oriental Cloud Pattern Design

Ming-Te Chi, Chen-Chi Hu, and Yu-Jyun Jhan

Department of Computer Science, National Chengchi University, Taiwan  
mtchi@cs.nccu.edu.tw, {102753501,97753033}@nccu.edu.tw

**Abstract.** Cloud patterns are essential design elements used in oriental art that harmonize well with other ornament patterns. The expression of oriental cloud patterns uses simple curves to abstract a natural phenomenon. This study analyzed the structures of traditional cloud patterns, and proposed a system that can create a similar style of structuring cloud patterns. The proposed system allows users to sketch simple strokes as an initial step in creating a cloud shape, and subsequently, the users can automatically generate various results that conform to the oriental cloud pattern.

**Keywords:** Non-photorealistic rendering, Oriental cloud pattern, Pattern generation.

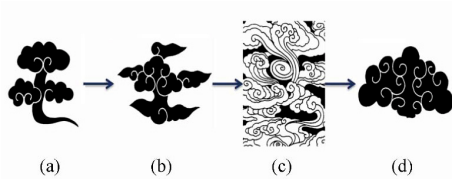
## 1 Introduction

Cloud patterns are common decorative elements used to abstract the natural cloud into a set of stylized curves. This pattern is an important decorating style in the Eastern part of the world. The torch used in the 2008 Summer Olympics was filled with cloud patterns. This kind of oriental cloud pattern is different from the western style. The western cloud patterns are classified by appearance. Accordingly in atmosphere science, the major cloud patterns are categorized as cirrus, cumulus, and stratus. In contrast, oriental cloud patterns, brimming with mysterious cultural features, express the shapes of clouds. The means of assisting users to create cloud patterns is a challenge and an interesting research topic in computer graphics. The non-photorealistic rendering (NPR) technology focuses on analyzing and integrating the principles of various arts and skills to establish a systematic method of reproducing stylization.

We endeavor to provide a sketch-based system that can generate oriental cloud patterns. The basic idea of the system is to convert a simple sketch of a cloud pattern into a complex one. To achieve this objective, the significant cloud designs from an ancient artifact are analyzed. These patterns, which were conceived by early artists, imply the aesthetic principles and understanding of the phenomenon of nature of different periods. By identifying these principles, we can build an algorithm that can be employed to create a digital cloud pattern.

The data show that cloud patterns evolve over time. Cloud patterns can be classified into several types, namely, original spin, thunder, flow, cirrus,

introverted cirrus, combined, expanded complex, and ruyi clouds. Each period features several patterns; the design of the cloud pattern of the latter stage is more stable than that of the previous stage. Moreover, the design of cloud patterns has not dramatically changed since the emergence of the introverted cirrus cloud pattern. Most patterns were generated by combining various introverted cirrus cloud patterns. Based on these observations, we focus on the introverted cirrus cloud pattern and succeeding patterns for our proposed system. The introverted cirrus cloud pattern is a mature design, which along with the combined, expanded complex, and ruyi clouds, is considered as an modern cloud pattern (Fig. 1).



**Fig. 1.** Evolution of cloud patterns: (a)Introverted-cirrus cloud, (b)Combined cloud, (c)Expanded-complex cloud, and (d)Ruyi cloud

We develop a decorative cloud pattern generation system by analyzing existing oriental cloud patterns. A sketch-based system is proposed to facilitate the creation process, and to develop an intuitive interface in the formation of patterns. This paper comprises two major parts. The first part discusses the analysis of the basic generating rules and the combination of features from collected oriental cloud patterns. The second part presents the generation of the oriental stylized cloud patterns from a sketch contour based on the generating rules.

## 2 Related Work

The present project relied on studies that can be classified into two categories, namely, model of clouds and smokes and space division or point distribution.

Clouds or smokes are important in many applications, including games, animations, and movies, among others. This explains the existence of numerous computer graphic studies on cloud and smoke rendering. Most of these studies focused on the particle system-based approach used to stack primitive objects to build cloud and smoke [7], [8], [9], [10]. Fedkiw et al. [3] proposed a smoke simulation system, which was adopted by Selle et al. [12] to arrange different stencils along the cloud contour. The system allows the computation of the effective range that can rationalize the layout of objects. The system used by Selle et al. can produce smoke animation in cartoon style. Zhou et al. [4] proposed an

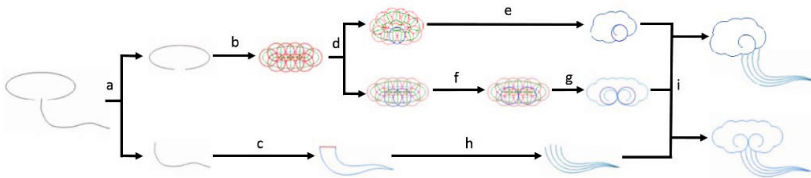
effective approximation to reduce the computation of compensated ray marching, which includes the concept of combining radial basis functions (RBFs) with residual field. Using RBFs can generate different sizes of spheres, and approximating the original data of the former can replace all calculations in a single approximation. Most of the above systems arrange different stencils along the cloud contour to depict a cloud pattern. This arrangement is adopted in this study to express the cloud contour and its generation.

NPR studies focused on generating stylization from a source image, in which the technique deals with space division and point distribution [5], [6], [11]. Hausner et al. proposed “Simulating decorative mosaics” [1], which combines direction field with centroidal Voronoi diagram (CVD), and generates a mosaic image that contains tiles of different colors from a source image. Kim et al. [2] adopted the Lloyd relaxation algorithms along with distance field to distribute points. This method can generate stippling images, in which the feature of the source image is preserved. In this study, the circle is the basic primitive pattern; the circles are constrained inside the user’s sketch contour. The CVD method is used to layout the position of circles in uniform distribution. The cloud patterns are generated by the repulsion between multi-circles.

### 3 Overview

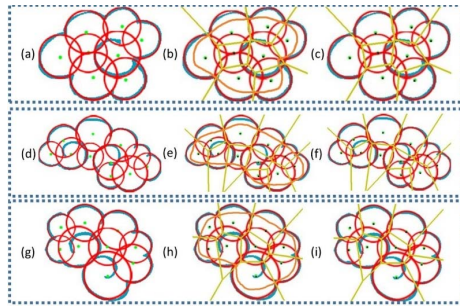
The flow chart of the proposed system is shown in Fig. 2. The system of generating oriental cloud patterns contains two components, namely contour, cloud body and cloud tail. Cloud body defines a region from the user sketch contour and arranges multi-circles inside the contour. Subsequently, cloud body shifts the circles to generate the cloud pattern from the outer cloud contour. Cloud tail employs the user input curve. In particular, this curve is smoothed and its range is calculated to generate the tail pattern.

Most cloud models drawn from the literature review are stacked with multi-small objects. The CVD, which is used to divide space into several regions, can also be used to simulate various natural patterns. Based on the presented ideas,



**Fig. 2.** Flow chart of generating oriental cloud patterns: (a) Structural classification: cloud body and cloud tail, (b) Arrangement of CVD circle, (c) Generation of cloud tail contour, (d) Distribution between the start and end circles, (e), and (g) Generation of contours, (f) Movement of circles until the circle becomes tangent, (h) Generation of inner curl, (i) Combination of cloud body and cloud tail

arranging the circles and generating the cloud pattern are feasible. Based on CVD, we formulated a modification that can allow the placement of circle and consequently generate a stylized cloud pattern. Fig. 3 illustrates the relationship between the modern cloud pattern and CVD. The three cloud patterns shown in Fig. 3(a), (d), and (g) are analyzed. The pattern is equipped with multi-circles. The circle centers are used as the seed in Voronoi distribution [Fig. 3(b), (e), and (h)]. The possible contour user may draw from the CVD and cloud shapes shown in Fig. 3(c), (f), and (i). The above observation shows that the CVD, with adjustment in the circle diameter, can be used to distribute circles uniformly in the sketch region. This hypothetical pattern is compared with existing cloud patterns. The results show that the user can easily setup the initial parameter, and draw an annular sketch curve for cloud body and a curve for tail pattern. The proposed system can automatically generate oriental cloud patterns.



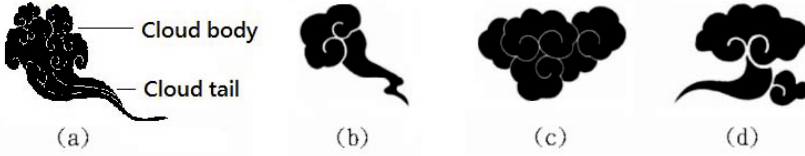
**Fig. 3.** Cloud patterns and the CVD: (a), (d), and (g) are the arrangements of origin circle; (b), (e), and (h) are the generation of CVD from (a), (d), and (g); (c), (f), and (i) are the movement of circles from (b), (e), and (h) inside the orange contours in (c), (f), and (i)

## 4 Generating Oriental Cloud Patterns

A cloud pattern can be divided into cloud body and cloud tail, as depicted in Fig. 4(a). The cloud body expresses the plump style, and the cloud tail depicts the flow direction. Cloud patterns can be arranged and combined based on these two components. The cloud body is the main structure of a cloud pattern. When combined with layers of the circle stack, various stylization of cloud bodies can be generated depend on the distribution of circles. The cloud tail can express the flow of a cloud; thus, the tail is generated when the cloud body is formed. When the cloud has no flow movements, the cloud body is expressed as static, and the cloud tail is not shown, as illustrated in Fig. 4(c).

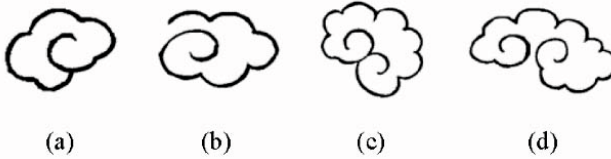
### 4.1 Cloud Body

The cloud body is mainly constituted by introverted hook curl. Thus, the structure of a cloud body is normally presented as a smooth, wavy curve. The introverted hook curl can be classified into single-hook curl (SHC) and double-hook



**Fig. 4.** Structure of an oriental cloud: (a) Complex structure formed by many cloud bodies and single cloud tail, (b) A cloud body with single cloud tail, (c) Multi-cloud bodies, and (d) Two cloud bodies with single cloud tail

curl (DHC), de-pending on the size of the hooks. Based on the images shown in Fig. 5, SHC and DHC have the features of merge and separation. Hence, the cloud body can be categorized into four types as follows: (a) Merged-SHC, (b) Separated-SHC, (c) Merged-DHC, and (d) Separated-DHC (Fig. 5).



**Fig. 5.** Analysis of a cloud body: (a) Merged-SHC, (b) Separated-SHC, (c) Merged-DHC, and (d) Separated-DHC

The cubic B-spline method is used to generate the cloud body. The cloud contour from the user input is smoothed, the start point is connected to the end point as a closed curve, and the circles are arranged inside the closed curve. Accordingly, generating the cloud body is divided into three steps, namely, circle arrangement, curl generation, and contour extraction.

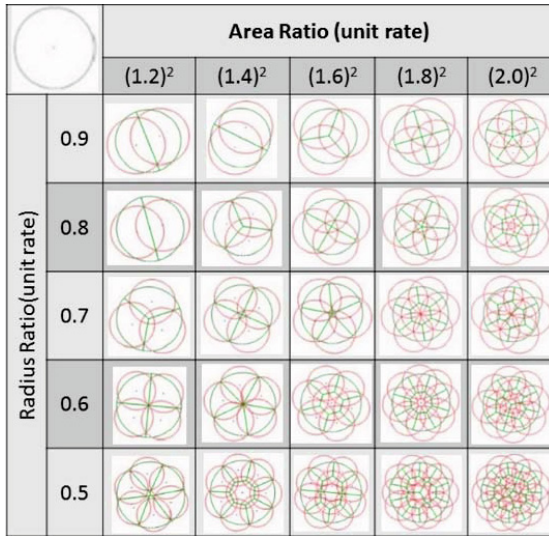
**Circle Arrangement.** To arrange the circles properly, the cubic b-spline method is used to smooth the cloud contour from the user input. The start point is connected to the end point as a closed curve, and the circles inside the closed curve are arranged.

Arranging the circles uniformly inside a specific contour is difficult. Thus, attraction and repulsion forces are used to balance the distribution of circles. The centers of the circles are treated as a point set by using the CVD method. At the CVD point distribution, the CVD line is the boundary of the force balance from two different circles. Consequently, the cloud contour from the user input becomes a force balance boundary between the outer circles.

The area of the cloud contour is used as the upper bound of the number of stacking circles. The number of the stacking circles, which can be arranged in



the area and determine the samples in CVD, is specific by user depend on the area. The initial distribution of circles within the range from the user input to the cloud contour is randomly sampled. The CVD is produced by using every point as the center of the circle and distributing the latter to the contour region. However, the centers of the circle outside the contour region are not used. The positions of the centroid in the Voronoi region are computed and replaced as the center of the circle. The steps are undertaken in a recursive manner as follows: 1. The CVD is processed. 2. The outer centers of the circles are removed. 3. The centroids are replaced as the centers of the circles. After the recursion, the circles are uniformly distributed. As a result, the small circle has the larger outward force and the size of the circle increases. By contrast, the large circle has a greater inward force inward and the size of the circle decreases. After the recursion convergence, each circle contains a uniform force in the cloud contour region, and the circles have the same size arrangement.



**Fig. 6.** Compression ratio diagram for the circle arrangement

To derive the bubble-like shape, the possible arrangement of circles inside the con-tour region is determined by the parameters. The results are shown in Fig. 6. Area ratio is the scale ratio in length unit of the cloud contour region. Radius ratio is the radius decline rate of the maximum inscribed circle in the cloud contour region. When the area ratio is between 1.42 and 1.82, the circular intersections are distributed near the contour. When the area ratio is less than 1.42, the circular intersections are distributed inside the contour. When the area ratio is greater than 1.82, the circular intersections are distributed outside the contour. Therefore, the cloud generating range is close to the cloud contour from the user input when the area ratio is set between 1.42 and 1.82.

**Curl Generation.** Curl is generated at the outer circles on the boundary. We need to extract the outer circles to outline the curl and contour of cloud body. The set of outer circles can be identified with the arrangement of circles.  $S_B$  is assumed as the outer circle set, and  $T_B$  is the total circle set. At the beginning, assume  $|S_B| = 0$  and  $|T_B| = n$  ( $n$  denotes the total number of circles). The algorithm begins at the circle near the starting control point from user sketch, the circle is the first selected circle ( $C_1$ ), it is placed in  $S_B$ , and removed from  $T_B$ . At the same time,  $|S_B| = 1$  and  $|T_B| = n - 1$ . A circle in  $T_B$ , which is adjacent to the circle in  $S_B$  and along with the direction of user sketch, is identified as  $C_2$  using the information of CVD.  $C_1$  in  $S_B$  is set as the original point, and the vector is computed as  $V_2$  from  $C_1$  to  $C_2$  in  $T_B$ . Another adjacent center of a circle ( $C_n$ ) can be identified in  $T_B$ . Moreover,  $C_n$  is also the last circle to the iteration steps.  $C_n$  is placed in  $S_B$  as the last circle, and  $C_n$  is removed from  $T_B$ . Thus,  $|S_B| = 2$  and  $|T_B| = n - 2$ . When the above settings are completed, the next steps are based on the iteration steps in algorithm 1. After executing algorithm 1,  $S_B$  contains the entire circles on the boundary.

According to the above steps,  $C_1$  is the beginning circle and  $C_n$  is the end circle at the outer circles. The process of generating curl is classified into three different cases. In the first case,  $C_1$  and  $C_n$  are at the same circle; hence, the curl is generated by con-touring from the end point along with the user input. If the end point is on the left of the original point, the curl is generated counterclockwise. However, if the end point is on the right of the original point, the curl is generated clockwise. This case is similar to the merged-SHC structure. In the second case,  $C_1$  and  $C_n$  are not connected. Thus,  $C_1$  and  $C_n$  generate clockwise and counterclockwise curls, respectively. Therefore, this case is identical to the separated-DHC structure.

In the third case,  $C_1$  and  $C_n$  intersect in one or two points; it will be merged-DHC structure. However, if  $C_1$  and  $C_n$  intersect in two points, it will cause the double hook curl overlap, and the overlap is what we want to avoid. The two possible examples are shown in Fig. 7. The circles are shown in red color with two points intersection, and we want to move them until they are tangent to each other (are shown in green color). The blue points are the points tangency between the adjacent circles. The yellow circles ( $C_1$  and  $C_n$ ) are the adjacent circles. The pink point is the middle point of the two red centers. Our goal is to move the circles until they are tangent at the pink point just like the green circles. If the adjacent yellow circles are at the same side [Fig. 7(a)], the red circles shift respectively to the lower left and lower right until the circles are tangent to each other at the pink point. If the adjacent yellow circles are at the different side [Fig. 7(b)], the left red circle shifts along with the vector to lower left, and the right red circle shifts along with the vector to upper right until the two circles are tangent to each other at the pink point. Therefore, the case constitutes the merged-DHC structure.

---

**Input:** total-circles  $T_B$ .

**Output:** searching-circles  $S_B$ .

**Procedure** *SearchCircles*(total-circles  $T_B$ )

// Initialization Step

**Foreach** circle in  $T_B$

Assume that  $C_1$  and  $C_n$  are adjacent circles

Prev =  $C_n$ ;

Center =  $C_1$ ;

$S_B = S_B \cup \{C_1, C_n\}$ ;

$T_B = T_B - \{C_1, C_n\}$ ;

//Search Step

**Repeat**

**Foreach** circle in  $T_B$

$\mathbf{V}_1 = Prev - Center$ ;

Next =  $C_x$ ;

Assume that  $\mathbf{V}_2 = Next - Center$ ;

$\Theta_x = \arccos \frac{\mathbf{V}_1 \cdot \mathbf{V}_2}{|\mathbf{V}_1| |\mathbf{V}_2|}$

Choose Next, arg max  $\Theta_x$ ;

$S_B = S_B \cup \{C_{Next}\}$ ;

$T_B = T_B - \{C_{Next}\}$ ;

Prev = Center;

Center = Next;

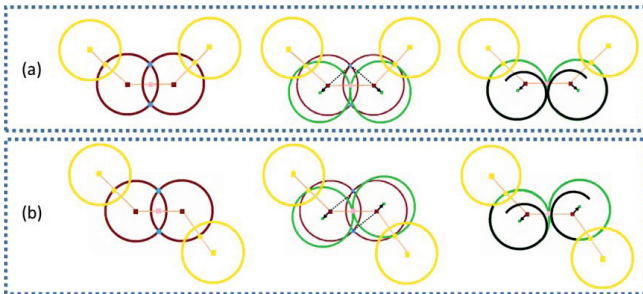
**Until** Center =  $C_n$ ;

**Output**  $S_B$ ;

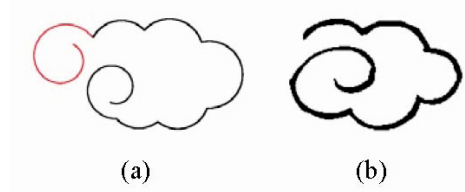
---

### Algorithm 1: Curl searching

However, if  $C_1$  and  $C_n$  are at the same side (left or right) in the second case, but one circle is far from the center of cloud [Fig. 8(a)], the upper curl is removed, and the separated-SHC structure is formed [Fig. 8(b)].



**Fig. 7.** Resolve the overlap in double hook curl. Case (a): Adjacent circles are in the same side. Case (b): Adjacent circles are in the different side.



**Fig. 8.** Special case: (a) Separated-DHC; removal of the red contour (b) Separated-SHC

**Contour Extraction.** After the circle placement is completed, the circles are converted into simple curves, completing the cloud body. First, the outer contour is traced and connected from the circles on the input boundary. Each circle in set  $S_B$  stores the positions of the center of the circle, left intersection, and right intersection. The hook type is based on the distance of the start and end points from the user specific contour. In case (1), only the single hook converts the circle into a spiral curve when the distance is small [Fig. 2(e)]. In case (2), the circles on the start and end points are converted to two hook in counterclockwise and clockwise spiral curves when the distance is large [Fig. 2(g)].

## 4.2 Cloud Tail

Cloud tail depicts the flow of cloud. The common types of cloud tail are: (a) merged-tail and (b) separated-tail (Fig. 9). To strengthen the wind force, mobility, and velocity features in the cloud tail, stylization is controlled by length, density, merged-tail, and separated-tail.



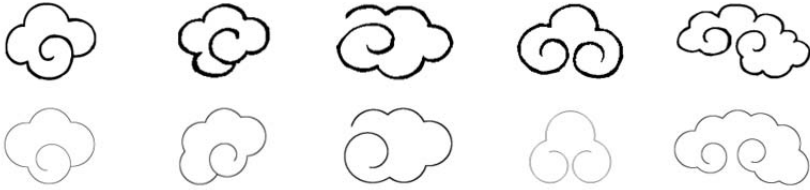
**Fig. 9.** Model of cloud tail: (a) Merged-Tail (b) Separated-Tail

To smooth the curve from the user input, the Bezier curve method is applied many control points on the curve. According to the observation of the tail structure, the tail is normally placed near the introverted hook curl. The direction of the start point of the tail is tangent to the introverted hook curl because the former is used to express the wind flow from the cloud tail. In generating cloud tail, the user specific tail contour is followed by using the start and end points of the cloud contour as the two initial points. Thus, separated-tail is generated

without changing the length and the region. By contrast, merged-tail is formed by changing the length through interpolation.

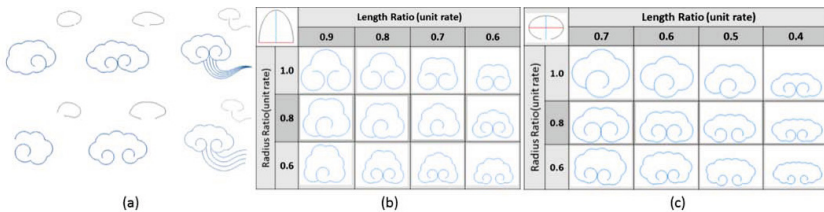
## 5 Results and Discussion

The proposed method was implemented by C++ on Windows 7. All experiments were evaluated on a PC with a 2.6 GHz CPU and 2 GB memory. On average, the proposed method takes 10 seconds to generate a pattern. Fig. 10 shows a comparison of traditional patterns and those produced by the proposed method. The proposed method can generate patterns that are similar to the traditional ones.



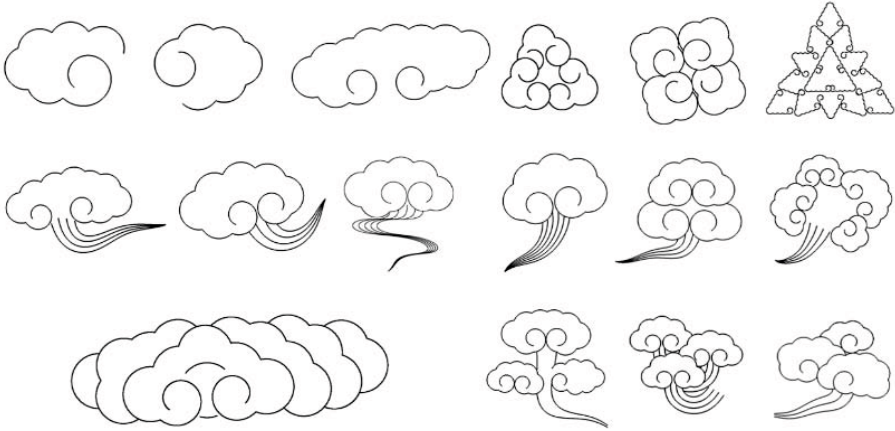
**Fig. 10.** Basic cloud patterns: Top: traditional patterns; Bottom: patterns produced by using the proposed method

A user can generate cloud patterns not only from sketch (Fig. 11(a)), but also from the parametric contour definition. Fig. 11 illustrates the effectiveness of the general contour achieved by adjusting the parameter. In Fig. 11(b), length ratio is the ratio of half ellipse bottom (red line) to half ellipse height (blue line), and radius ratio is the ratio of the maximum inscribed circle diameter of the cloud contour. In Fig. 11(c), length ratio is the ratio of semi-major axis (red line) to semi-minor axis (blue line) of the ellipse, and radius ratio is the ratio of the maximum inscribed circle diameter of the cloud contour.



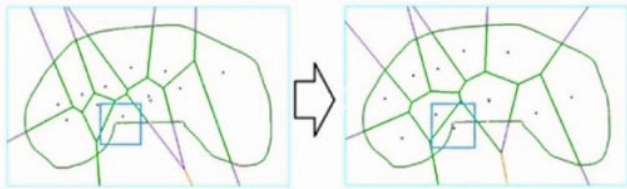
**Fig. 11.** Results. (a) Generating cloud patterns from sketch; and symmetric cloud pattern generated with different parameters (b) Half-ellipse, and (c) Ellipse.

In the proposed system, users can generate complex cloud patterns that are characterized with personal styles. Fig. 12 shows various results from using simple sketch, and the arrangement of cloud patterns from simple to complex.



**Fig. 12.** Complex cloud patterns under different arrangements

The proposed method, however, has one limitation. If the contour from the user input is concave, the center of the circle will go out of the contour range in the CVD recursion step (blue box in Fig. 13), inducing unsatisfactory results. Thus, the use of a concave contour as current system input is avoided.



**Fig. 13.** Concave contour

## 6 Conclusion and Future Work

A sketch-based oriental cloud pattern generating system is presented as an intuitive design tool for generating cloud patterns with few sketch curves. The structures of clouds from existing oriental cloud patterns are collected and analyzed. The modified CVD method is used to distribute circles inside the user sketch contour to compose the structure of a cloud body. A cloud tail style is also produced from the simple sketch curve. The generated cloud pattern can serve as a design element in games, computer animations, and decorations.

In the future, we plan to develop an interactive system with wind effect. The additional vector field will deform and move the cloud pattern to generate

dynamic cloud patterns. The enhancement of the style of the cloud pattern is part of this plan. The current system focuses on simple circles, in which various shapes can be used as the primitive pattern to compose the cloud body. Color and line width can also be employed to express the cloud feature. The recursive structure can also be considered in formulating a hierarchy level; thus, providing various stylizations.

**Acknowledgments.** This work was supported by the National Science Council, Taiwan under NSC-102-2221-E-004-008.

## References

1. Hausner, A.: Simulating decorative mosaics. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 573–580. ACM (2001)
2. Kim, D., Son, M., Lee, Y., Kang, H., Lee, S.: Feature-guided Image Stippling. *Computer Graphics Forum* 27(4), 1209–1216 (2008)
3. Fedkiw, R., Stam, J., Jensen, H.W.: Visual simulation of smoke. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 15–22. ACM (2001)
4. Zhou, K., Ren, Z., Lin, S., Bao, H., Guo, B., Shum, H.Y.: Real-time smoke rendering using compensated ray marching. *ACM Transactions on Graphics (TOG)* 27(3), 36 (2008)
5. Balzer, M., Schlömer, T., Deussen, O.: Capacity-constrained point distributions: a variant of Lloyd’s method 28(3), 86 (2009)
6. Ebeida, M.S., Davidson, A.A., Patney, A., Knupp, P.M., Mitchell, S.A., Owens, J.D.: Efficient maximal poisson-disk sampling. *ACM Transactions on Graphics (TOG)* 30(4), 49 (2011)
7. McGuire, M.: A real-time, controllable simulator for plausible smoke. Tech Report CS-06-03, Brown Univ., Providence, RI (2006)
8. McGuire, M., Fein, A.: Real-time rendering of cartoon smoke and clouds. In: Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering, pp. 21–26. ACM (2006)
9. Rana, M.A., Sunar, M.S., Shamsuddin, S.M.: Particles Cloud Modeling Algorithm for Virtual Environment. *Asian Journal of Information Technology* 5(5), 555–565 (2006)
10. Wang, N.: Realistic and fast cloud rendering. *Journal of Graphics Tools* 9(3), 21–40 (2004)
11. Fattal, R.: Blue-noise point sampling using kernel density model. *ACM Transactions on Graphics (TOG)* 30(4), 48 (2011)
12. Selle, A., Mohr, A., Cheney, S.: Cartoon rendering of smoke animations. In: Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering, pp. 57–60. ACM (2004)

# D-Sweep: Using Profile Snapping for 3D Object Extraction from Single Image

Pan Hu, Hongming Cai, and Fenglin Bu

School of Software, Shanghai Jiao Tong University, Minhang District, China

**Abstract.** The perception of an object from a single image is hard for machines but is much easier for humans as humans often have prior knowledge about the underlying nature of the object. Combining the power of human perception with the computation capability of machines has led to some key advances for ill-posed problems such as 3D reconstruction from single image. In this work we present D-sweep, a novel method for modeling 3D objects using 2D shape snapping and 3D sweep. The user assists recognition and reconstruction by choosing or drawing specific 2D shapes and placing them on either end of the object in the image. The machine first snaps the shape to the end of the object by fitting its projection to the automatically detected contour lines. Once the profile is determined, the user can sweep it along a specific 3D trajectory to reconstruct the 3D object.

**Keywords:** 3D Acquisition, User Interaction, Image Processing.

## 1 Introduction

The task of extracting three dimensional objects from a single photo is very difficult and even ill-posed as some information is lost from 3D scene to 2D image. However, this task is of great practical relevance since 1) photos are easy to find especially when only one photo for a particular object is needed and 2) photos can serve as a guideline in the modeling process, reducing the workload of professional 3D modelers

In this paper we introduce an interactive technique to extract 3D objects from a single photograph. D-sweep stands for "Describe and sweep". Our approach largely follows classical 3D sweep methods but takes advantage of human's perceptual abilities and prior knowledge on the object. The user helps to recognize and describe the profile as well as the main axis of the object to constrain the solution space while the machine performs the rest of the task which requires high accuracy or complex computations. The information the user provides bridges the gap between what is given in the image and what is actually needed to reconstruct the object in 3D. Textures are derived from the photograph and preserved as well as the object's geometry and structure.

Our modeling process contains three consecutive steps. The user first identifies the photograph and chooses appropriate 2D primitives that compose shapes such as circles, triangles and rectangles. Alternatively, the user can define a 2D



irregular shape himself by drawing it with the mouse. The user then drags the primitive to one end of the object and the system automatically matches curves of the primitive to contour lines that are automatically detected in the image. Once the fitting process is over and the 2D profile of the object is defined, the user can sweep it to create the final 3D object. Throughout the sweep, the program dynamically adjusts the progressive profile by sensing the pictorial context in the photo and automatically snapping to it.

Our approach allows the user to assist the recognition and the extraction of 3D objects by explicitly defining their profiles and their curved axis. Meanwhile, the rest of the task is accomplished by an automatic, computational process. Using this technology makes it possible for both professionals and non-professionals to efficiently extract 3D objects from photographs. Despite the non-linearity and non-convexity of the fitting problem, the optimization process is indeed quite fast.

The remainder of this paper is organized as follows. We first briefly survey some related work and some current systems that can be used to extract 3D objects from photographs (section 2). We next investigate the issue of information loss from 3D scenes to 2D photos and explain the necessity of using human perceptual abilities to solve the problem (section 3). These observations and reasoning lead to a novel variant of 3D sweep, which we describe at length in section 4, 5 and 6. We then present our experimental results and discuss some possible future directions in section 7. Section 8 summarizes our contributions and presents our conclusions.

## 2 Related Work

**Traditional Sweep-Based Modeling.** Sweeping a 2D area (usually referred to as a sweep template) along a specific 3D trajectory to create 3D models is a well-known technique in computer aided design [1, 2] and a large body of work has been done in this area. Early computer-aided-design systems [3] usually used boundary representations for modeling sweep surfaces. However, these sweep solids lacked a robust mathematical foundation, making self-intersecting sweeps simply invalid. Angelidis et al. proposed a volume preserving modeling technique called Swirling-Sweepers [4], which allowed arbitrary stretching while avoiding self-intersection. An implicit modeling method with no topological limitations on the 2D sweep profile was proposed by Schmidt and Wyvill [5], allowing surfaces to be blended smoothly. These methods all model 3D objects with sweeps but none of them uses images or sketches to guide the modeling process.

**Image-Based Modeling.** Methods for interactive modeling from multiple photographs or video sequences have been proposed in [6–8]. These methods require users to explicitly mark edges or polygons or place 3D primitives in multiple photographs or frames before the systems could automatically align them and extract models, whereas in our problem setting there is only one single image

for one particular object. Therefore, these methods could not be applied in this circumstance.

Extracting a 3D object from a single image requires a certain degree of semantic understanding of the object. To this effect, [9, 10] are proposed. In these methods, the user has to explicitly place primitives and add semantic annotations to them in order to model a 3D shape. Unlike our work, these techniques require explicit annotations and extensive manual operations.

Semantic understanding could also be integrated into the modeling system in the form of prior knowledge. Given a photograph of an object and a database of 3D models in the same class, Xu et al. [11] presented a system that creates a 3D model to match the photograph requiring only minimal user interaction. Rother and Sapiro [12] proposed a probabilistic framework of 3D object reconstruction from a single image that encodes prior knowledge about the object class in a database and projects it to 2D when it is used, casting the 3D reconstruction problem to a statistical inference problem. In contrast, our method does not rely on any database of similar models.

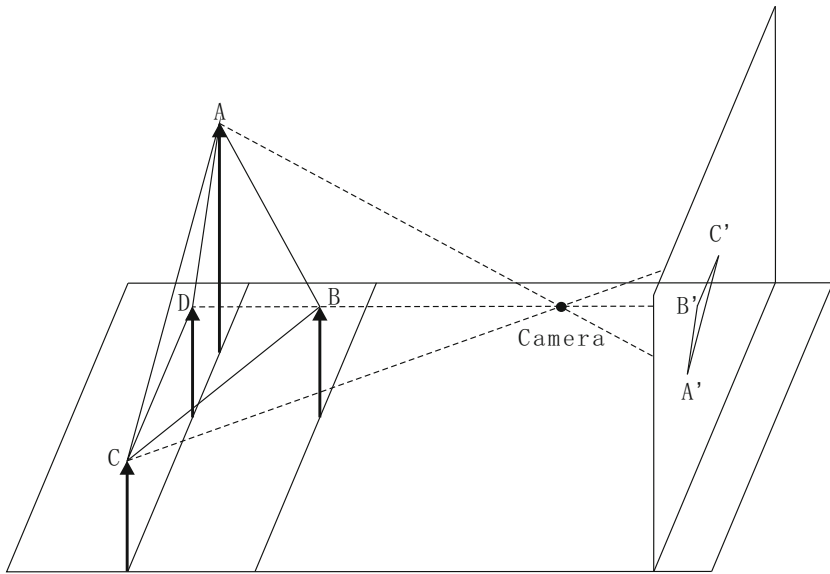
Our work is closely related to the recent work of Chen et al. [13] that uses three strokes provided by the user to generate a 3D component that snaps to the shape’s outline in the photograph with each stroke defining one dimension of the component. However, this approach only works for a limited number of primitives such as generalized cylinders and cuboids. In our work a larger range of 2D profiles, including irregular ones are supported.

**Sketch-Based Modeling.** Sketches can be used to assist 3D modeling process. Igarashi et al. introduced Teddy [14], a sketching interface for 3D freeform design. This inspired a large body of follow-up work; refer to [15] for a detailed survey. These methods used a so-called sketch-rotate-sketch workflow and allowed users to build 3D objects by sketching.

Our approach is inspired by that of Shtof et al. [16], which models 3D objects from sketches rather than by sketching. In their work, the user assists recognition and segmentation by choosing and placing specific 3D geometric primitives on the relevant parts of the sketch. However, our approach tries to snap 2D profiles instead of 3D primitives. Moreover, we do not fit shapes to sketches, but to contour lines automatically detected in real images. Another difference is that [16] supports a limited number of primitives such as generalized cylinders, spheres and boxes whereas our approach is capable of more complex shapes.

### 3 Analysis

When a photo is taken by a camera, a perspective projection occurs, mapping three-dimensional points to a two-dimensional plane. As the dimensionality reduces, some important information is lost. For example, parallelism and perpendicularity might not be preserved. In another word, angles might change after the projection. The extent to which a specific angle changes, relies on many factors including the positions of the object and the camera, as well as their

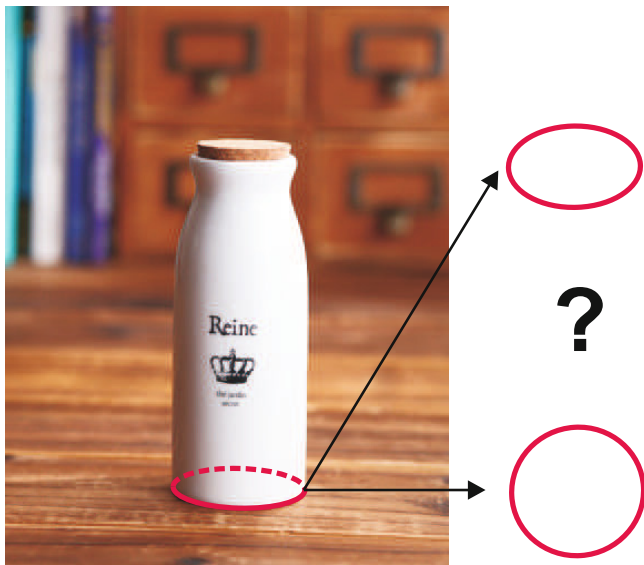


**Fig. 1.** Two different triangles, namely  $\triangle ABC$  and  $\triangle ADC$  are projected to the same shape:  $\triangle A'B'C'$

orientations. Consequently, multiple different shapes could be projected to one same 2D shape on the resulting photo under certain conditions. See an example shown in Figure 1.

As shown in Figure 1, having  $\triangle A'B'C'$  on an image, it is hard for a machine to automatically determine the actual location of the triangle in 3D space. The triangle could be either  $\triangle ABC$  or  $\triangle ADC$ , or some other shape. This kind of ambiguity makes the task of extracting 3D models from single photos extremely difficult for machines. On the contrary, the cognitive task of understanding the geometry and structure of a shape, its projection and their relations is often simple for humans. This is because humans often have prior knowledge on the objects that we want to extract and can thus draw some inference based on their experience. Figure 2 shows an ordinary photo of a milk bottle. We assume that bottom contour can be perfectly recognized by a certain contour detection algorithm and that the object we want to extract is symmetrical. The assumption we make is a strong one, yet a machine still cannot tell whether the 2D profile of the object is an ellipse or a circle. In contrast, upon seeing the photo, we humans can rapidly identify that the object to be extracted from the photo is a milk bottle. Milk bottles tend to have circular bottoms rather than elliptical ones. Now what remains to be done is to pass the information we perceive to the machine and let it do the rest of the work.

The 3-sweep system proposed by Chen et al. [13] implicitly defines different user-perceived geometrical constraints on 2D profiles by using different gestures. Two kinds of simple 2D shapes are supported, namely circles and squares. Unlike



**Fig. 2.** Determining the 2D profile of a simple object is difficult for machines

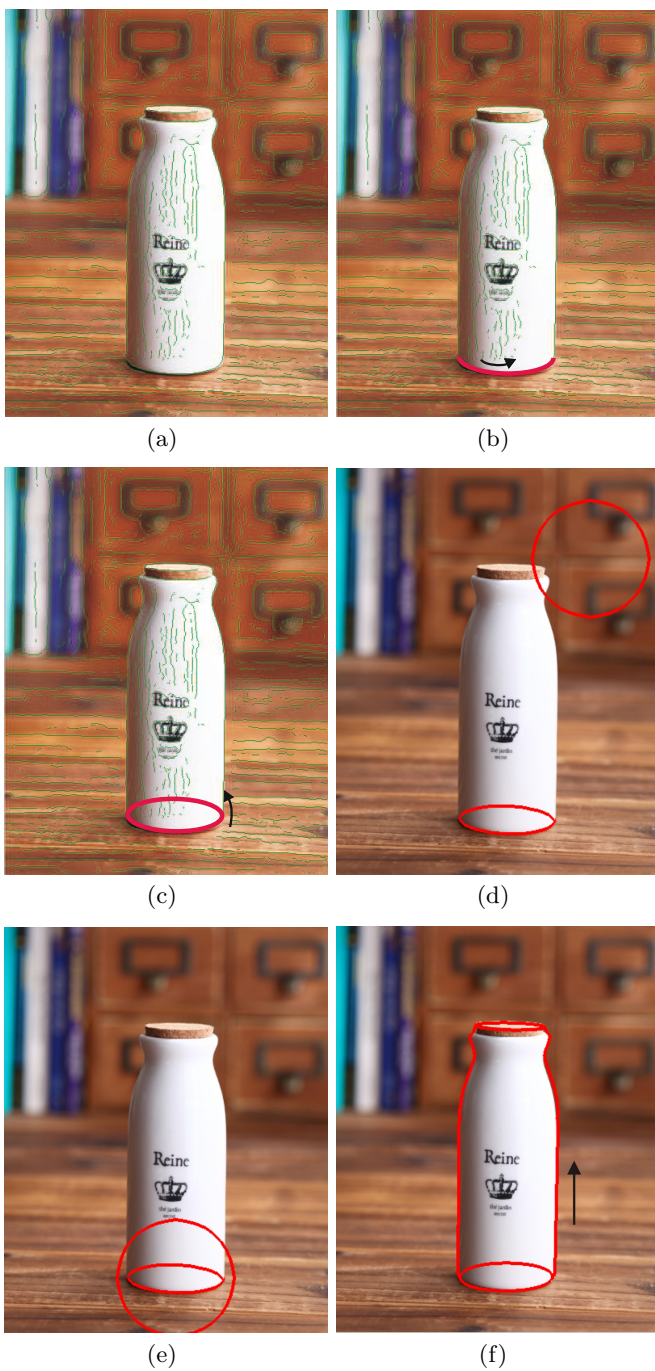
this method, our approach allows the user to directly define 2D profiles of the object to be extracted and thus supports a larger range of 3D objects. We next provide an overview to our D-sweep technique.

## 4 Overview

Our interactive modeling approach takes as input a single photo such as the one in Figure 2. Our goal is to extract 3D component whose projection matches (a part of) the object in the given photo. Creating multiple 3D components one by one and using them to construct more complex 3D models is, however, beyond the scope of our paper (see, for instance, [13] and [16]). We focus on single component fitting. By explicitly describing a 2D profile and sweeping it along a specific trajectory, the user can efficiently construct a 3D component that is consistent with its counterpart in the photograph.

Although the user interacts with the photo by describing 2D profiles and sweep trajectories, D-sweep also relies on automatically snapping shapes to object outlines created from image edges. To detect object contours and build candidate object outlines from them, we use the hierarchical edge feature extraction method proposed by Arbelaez et al. [17]. Each pixel of the image is associated with a numerical value, indicating the probability of this pixel being part of a contour. We then apply the technique proposed by Cheng [18] to link the detected contour pixels into continuous point sequences.

An overview of the workflow of our system is presented in Figure 3. First, an image is loaded into the system and automatic contour detection takes place,



**Fig. 3.** Overview of our method

showing the user multiple candidate object outlines. The user then marks the profile contour, which will later be used for geometric snapping, by casually tracing over it, as shown in Figure 3(b). This semantic task is very simple for the user to accomplish while machines can hardly distinguish bottom contours from others. It is possible that the chosen bottom outline is incomplete with the other part of it sheltered by the object itself, which is indeed the case in Figure 2. To handle this issue, we check the completeness of the chosen curve by simply comparing its starting and ending position. If they are far away the system asserts that the contour is incomplete. Under the assumption of centrosymmetry and ignoring perspective distortion, the system will carry out an auto-completion to the existing contour, rotating it by 180 degrees around the center of the bottom surface. The center lies at the midpoint of the line segment connecting starting and ending point of the existing contour.

Once the appropriate bottom contours are chosen (and completed), the user picks up a 2D shape from the pre-defined primitive list or he can draw one by himself, as shown in Figure 3(d). The user then approximately scales the 2D shape to the size of the object’s profile, and drags it to the appropriate position (Figure 3(e)). When the user releases the 2D shape, an optimization using the fitting objective function is performed so as to determine both the orientation and the size of the 2D profile.

Finally the user defines the main axis of the 3D component using a straight or curved stroke. Our method for this step largely follows that of Chen et al. [13]. The user sweeps the 2D profile along a curve that approximates the main axis of the 3D object. During the curve drawing procedure, copies of the 2D profile are placed along the curve and most of them are snapped to the object’s side outline. Side contours are not always correctly detected. In places where no appropriate side outline can be found, we simply clone the last profile and translate it to the current position. Figure 3(f) shows the result of this final operation. The 3D model for a milk bottle is successfully built.

## 5 Contour Marking

The profile contour of the object has to be chosen manually since the system cannot automatically distinguish it from other lines. We provide a smart interface to help users accomplish this task in a simple and straightforward manner. First, the user clicks somewhere on the image and the system iterates over all contour lines to find the nearest contour point, as well as the contour line on which it resides. The contour point found in this operation is marked as the starting point of the final contour line. Second, the user casually traces over the chosen contour line and marks vertex points of the profile using left clicks. As the cursor moves, the point that 1) is closest to the current cursor position and 2) resides on the previously determined contour line is marked as the ending point of the current line. Third, the user ends the whole contour marking process by using a simple right click. The contour will then be automatically completed, as is previous discussed. In situations where the bottom shape is asymmetric, the user can

manually complete its contour using line segments. More details on how profile contours are defined can be found in the supplementary video.

Once the profile contour is marked, we calculate the geometric center  $c'_{xy}$  of the marked vertices as follows:

$$c'_{xy} = \frac{\sum_{i=1}^N s_i}{|S|} \quad (1)$$

where  $S = \{s_i\}$  represents the set of vertices on the contour line.

In case the contour line is a curved one and no vertex is marked, we calculate  $c'_{xy}$  as the arithmetic average of all the points on it.

## 6 Curve Fitting

**Circle.** To perform snapping, the following objective function  $\phi$  is constructed:

$$\phi(c, n, d) = \sum_{i=1}^N [(n_z(c_{xy} - s_i))^2 + (n_{xy} \cdot (c_{xy} - s_i))^2 - (rn_z)^2]^2 \quad (2)$$

where  $S = \{s_i\}$  represents the object bottom contour. For the user-defined circle,  $n = \{n_x, n_y, n_z\}$  denotes its normal direction;  $c = \{c_x, c_y, c_z\}$  defines its center;  $r$  represents its radius. This objective function is based on the observation that a 2D point lies on the projection of a 3D circle if, for a certain  $z$  coordinate (in the implementation we let  $n_z = 1$  to simplify calculation), it resides on the plane defined by the center and the normal direction of the 3D circle, and its distance from the center equals the radius  $r$ .

**Polygon.** A polygon can be uniquely determined by its vertices. Therefore, we represent polygons with their vertex points. As is mentioned before, vertex points are marked manually in the curve choosing step. Since we know exactly where these points are on the profile contour, we formulate the following objective function based on the reason similar to that explained for function (2):

$$\phi(c, n, d) = \sum_{i=1}^N [(n_z(c_{xy} - s_i))^2 + (n_{xy} \cdot (c_{xy} - s_i))^2 - (dr_i n_z)^2]^2 \quad (3)$$

where  $S = \{s_i\}$  represents the set of vertices on the contour line. For the user-defined polygon,  $n = \{n_x, n_y, n_z\}$  denotes its normal direction;  $c = \{c_x, c_y, c_z\}$  defines the geometric center of its vertices;  $R = \{r_i\}$  represents the distances from its vertex points to the center  $c$  (which are constants once the polygon is drawn);  $d$  defines the actual size of a polygon (which is to be optimized).

**Optimization.** To solve this constrained optimization problem we use a different strategy to that of [16], which involves in using an augmented Lagrangian method [19] and L-BFGS [20]. Augmented Lagrangian methods are used to

replace constrained optimization problems by unconstrained problems and L-BFGS is used to solve the latter. The optimization problem that we formulate is essentially an unconstrained one. We further simplify the problem by letting:

$$c_{xy} = c'_{xy} \quad (4)$$

Now since there are only three unknown factors to be found both for function (2) and (3), we simply use BFGS method to solve them. Our experimental results show that this optimization procedure is extremely rapid and can be carried out in real-time.

## 7 Results

**Modeling from Single Image.** Figure 4 shows several modeling results. In the left column we show the input photos, in the middle column we show the extracted 3D models represented by lines, and in the third column they are textured. Textures are mapped using a similar strategy to that introduced in [13]. For a certain 3D point of the object, the 2D coordinates of its projection on the image plane can be directly used as its texture coordinates. As there is no texture information regarding the back side of the extracted object, we simply mirror the front texture to the back.

3-Sweep method proposed in [13] can only handle 3D components with square or circle bottoms. The method introduced in [16], similarly, is only capable of generalized cylinders and spheres. In reality, many objects cannot be decomposed or categorized to such simple primitives. For example, the pyramid in Figure 4(d) has a triangular bottom while the vase in Figure 4(g) has a hexagonal bottom. These objects, and many other objects with polygonal bottoms, cannot be modeled by previous methods. However, with our system, users can model such objects in an efficient and elegant manner. Details about how our system works can be found in the supplementary video.

**Limitations.** Our work has several limitations. Firstly, although polygons and circles are supported as profiles in our method, many objects have irregularly curved profile contours, making it impossible for our system to work properly. Adopting the generalized sweep template introduced in [5] can extend our application range. Secondly, our method does not allow non-uniform scale of the 2D profile during the sweep. For example, a toothpaste tube that is round at one end and flat at the other cannot be modeled by our method. Defining independently both ends of an object, as well as its main axis, might help in this situation, but leads to more complex user controls.

In our work we assume that the object is neither too close to the camera, nor too far away from it. In other words, the angle of view should not be too large or too small. We further assume that the main axis of an object is mainly visible and parallel to the projection plane. Objects extruding along the viewing direction are hard to model. Another assumption is that object contours, especially those





**Fig. 4.** Modeling objects. Left: input photos. Middle: extracted 3D models represented by lines. Right: extracted 3D models with their textures.

along the main axis of the object, need to be correctly detected. Our system cannot recover objects with fuzzy edges. Also, the back side of an object cannot be texture mapped if it does not have a symmetric texture.

## 8 Conclusions

We have presented a novel interactive technique which can model 3D man-made objects from a single photograph by combining the cognitive ability of humans with the computational accuracy of computers. Compared with previous work, our approach supports a considerably larger range of 2D profiles. Users can not only choose a 2D profile from a predefined shape list, but also efficiently define one. Contour marking is also done in a flexible way, allowing users to specify bottom contours efficiently. The concept of modeling by sketching is thus integrated in our approach. In future, we hope to allow modeling of more complex/natural 3D objects. We also wish to add inter-part semantic constraints on the shapes as in some previous work.

**Acknowledgments.** This research is supported by the National Natural Science Foundation of China under No. 61373030, 71171132.

## References

1. Requicha, A.: Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys* 12(4) (1980)
2. Foley, J.D., Van Dam, A., van Dam, A., S.K., Hughes, J.F., Phillips, R.: *Introduction to Computer Graphics*. Addison-Wesley (1993)
3. Requicha, A., Voelcker, H.: Solid modeling: a historical summary and contemporary assessment. *Computer Graphics and Applications* 2(2), 9–24 (1982)
4. Angelidis, A.: Swirling-sweepers: Constant-volume modeling. In: *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, pp. 10–15 (2004)
5. Schmidt, R., Wyvill, B.: Generalized sweep templates for implicit modeling. In: *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pp. 187–196 (2005)
6. Debevec, P., Taylor, C., Malik, J.: Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In: *Proceedings of ACM SIGGRAPH*, pp. 11–20 (1996)
7. Sinha, S., Steedly, D., Szeliski, R.: Interactive 3D architectural modeling from unordered photo collections. *ACM Transactions on Graphics (TOG)* 27(159) (2008)
8. Van Den Hengel, A., Dick, A.: VideoTrace: rapid interactive scene modelling from video. *ACM Transactions on Graphics (TOG)* 26(3) (2007)
9. Gingold, Y., Igarashi, T., Zorin, D.: Structured annotations for 2D-to-3D modeling. *ACM Transactions on Graphics (TOG)* 28(148) (2009)
10. Lau, M., Saul, G., Mitani, J., Igarashi, T.: Modeling-in-context: User design of complementary objects with a single photo. In: *Proceedings of the Seventh Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pp. 17–24 (2010)

11. Xu, K., Zheng, H., Zhang, H., Cohen-Or, D., Liu, L., Xiong, Y.: Photo-inspired model-driven 3D object modeling. *ACM Transactions on Graphics (TOG)* 30(80), 1–10 (2011)
12. Rother, D., Sapiro, G.: 3D Reconstruction from a Single Image. Submitted to *IEEE Transactions on Pattern Analysis and Machine Learning*, IMA Prepr International (2009)
13. Chen, T., Zhu, Z., Shamir, A., Hu, S., Cohen-Or, D.: 3-Sweep: Extracting Editable Objects from a Single Photo. *ACM Transactions on Graphics (TOG)* 32(195) (2013)
14. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3D freeform design. In: *Proceedings of ACM SIGGRAPH*, pp. 409–416 (1999)
15. Olsen, L., Samavati, F.F., Sousa, M.C., Jorge, J.A.: Sketch-based modeling: A survey. *Computers Graphics* 33(1), 85–103 (2009)
16. Shtof, A., Agathos, A., Gingold, Y., Shamir, A., Cohen-or, D.: Geosemantic snapping for sketch-based modeling. *Computer Graphics Forum* 32(2.2), 245–253 (2013)
17. Arbelaez, P., Maire, M.: Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Learning* 33(5), 898–916 (2011)
18. Cheng, M.: Curve structure extraction for cartoon images. In: *Proceedings of the 5th Joint Conference on Harmonious Human Machine Environment*, pp. 13–25 (2009)
19. Nocedal, J., Wright, S.J.: *Numerical Optimization*. Springer (2006)
20. Liu, D., Nocedal, J.: On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45, 503–528 (1989)

# Fixit: A 3D Jigsaw Puzzle Game Using Multi-touch Gestures

Yi-Hsiang Lo, Che-Chun Hsu, Hsin-Yin Chang,  
Wen-Yao Kung, Yen-Chun Lee, and Ruen-Rone Lee

Department of Computer Science,  
National Tsing Hua University,  
Hsinchu, Taiwan  
{s9962123, s9962244, s9962202, s9962315,  
s9962336}@m99.nthu.edu.tw, rrllee@cs.nthu.edu.tw

**Abstract.** We propose Fixit, a 3D jigsaw puzzle game using touch gestures on Android mobile devices. The system consists of a back-end authoring process and a front-end gaming control. The authoring process generates the puzzle pieces, which are cut and refined by our algorithms from the input 3D model. The gaming control provides an intuitive graphical user interface, which utilizes touch screen multimodal gestures and allows users to reconstruct the 3D jigsaw puzzle pieces into the original 3D model.

**Keywords:** 3D jigsaw puzzle, multi-touch gesture, graphical user interface.

## 1 Introduction

2D jigsaw puzzles have been widely adopted as a kind of casual game to rebuild the original picture from a set of small 2D jigsaw puzzle pieces. In order to enhance the difficulty, the number of pieces is increased. People spend a lot of time in putting the right piece in the right place. The joy comes from revealing the final picture gradually. The step of solving jigsaw puzzles is by figuring out the position, the color, the orientation, and the integration to the pieces already “glued” together. Despite the fun of playing with it, the operations of putting a piece into the half-finished picture are simply 2D operations—translation and rotation. In the implementation of a jigsaw puzzle on a PC game or on a mobile game, users can only operate on a piece of jigsaw puzzle at a time. There are no or less operations we can apply to the half-finished picture. However, while we are playing the actual jigsaw puzzles, we always pick up two half-finished parts at the same time, rotating and observing, trying to figure out if they can be put together. Thus, in order to improve the gaming experience, we should have some operations that help us to deal with both half-finished pieces simultaneously. Those hand operations also need to be simulated by mouse, keyboard, or touch screen gestures operations.

With one more dimensionality, 3D jigsaw puzzles change the way of operations we can apply in playing the game in comparing to the 2D ones. Basically, it is similar to

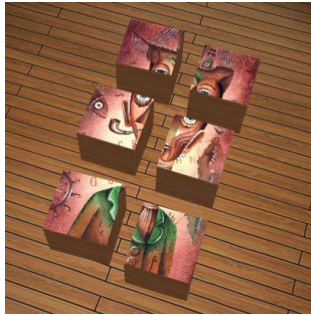
the 2D case in trying to glue a set of 3D jigsaw puzzle pieces together and resulting in a complete 3D model or scene. The primary operations are still the translation and rotation, despite they are 3D operations. However, the player usually has to pick up two puzzle pieces or partial macro puzzle pieces and try to search for the clues, such as profile, cross section, color, texture, etc., by rotating the two selected pieces simultaneously. Upon finding a possible match, attach one piece to another one by translating them close together, and see if they are perfectly match or not. It is different from 2D jigsaw puzzles in that you have to operate and examine on two parts, a piece or a macro piece, simultaneously instead of just a single piece in common 2D case.

Mobile devices with touch screen have become the most popular platforms for people to use in daily life. In order to truthfully reflect the operations in playing a 3D jigsaw puzzle game, some user-friendly multimodal gestures on touch panels are introduced in our system. The gestures on touch panels are so intuitive that user can easily map the gestures to the similar operations in real game, including simultaneous control.

## 2 Related Work

The 3D jigsaw puzzle games in the popular online application markets can be roughly categorized into three classes: cube puzzle, puzzle globe, and interlocked puzzle [1].

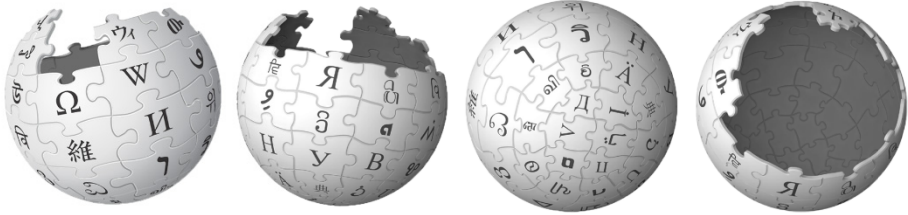
Cube puzzle (Fig. 1) is made up with cubes, which are colored or image-textured. They can only use the six surfaces of each cube to put on texture images, and this restriction makes cube puzzle lack of variety.



**Fig. 1.** Cube puzzle (Source: Picture Cubes by Buagaga, <http://www.kongregate.com/games/buagaga/picture-cubes>)

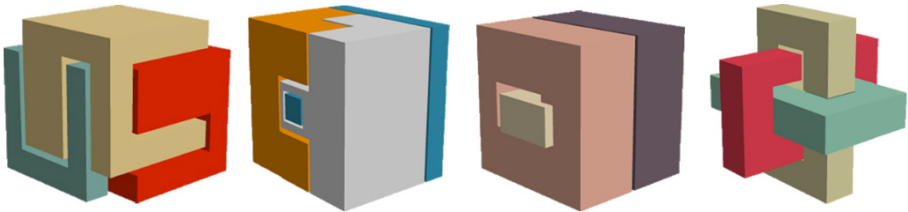
Puzzle globe (Fig. 2) is like a 2D jigsaw puzzle but warped into a hollow sphere. Textures are covered on the surface of the sphere, and then the hollow sphere is divided into several small curved jigsaw puzzle pieces. Although the final result is a 3D sphere, but the operations are similar to the ones used in 2D jigsaw puzzle. Moreover, since it is not a solid model, the matching occurs only at the silhouette of each jigsaw puzzle piece. *3D Polyomino Puzzle* [2] is another kind of 3D hollow jigsaw puzzle but

is not necessary to be in sphere type. Both puzzle globe and 3D polyomino puzzle can be regarded as a kind of 2D jigsaw puzzle due to that the restoration is solely on the model surface. The interior contributes nothing to the matching process.



**Fig. 2.** Puzzleglobe (Source: the logo of Wikipedia)

There is another type of puzzle games called *Interlocked puzzle*[3](Fig. 3) that gained its fame on the app market recently, and the further researches have been introduced in *Making Burr Puzzles from 3D Models* [4] and in *Recursive Interlocking Puzzles* [5]. An interlocked puzzle is constituted by several specially designed pieces which are interlocked by each other. Rather than restoring the original model or picture from the jigsaw puzzle pieces, users are to figure out a specific sequence to take apart those interlocked pieces. In most of the cases, there is only one correct sequence to separate them, and then, in reverse sequence, to restore them back. It is different from the traditional jigsaw puzzle games which have no specific sequence to decompose it or rebuild it.



**Fig. 3.** Interlocked puzzle

Despite different types of 3D puzzle games, we focus on providing a 3D jigsaw puzzle game where the 3D puzzle pieces were generated by cutting a 3D model without any specific sequence. A group of intuitive touch screen gestures were developed to simulate the real-life operations that people apply on actual 3D jigsaw puzzles.

### 3 System Overview

Our system, as shown in Fig. 4, can be divided into three parts: the Model Data, the Touch-Screen Gesture, and the Game System. The Model Data part is used to generate the 3D jigsaw puzzle from an input 3D model. In this part, the input model will be

cut into pieces, and the adjacency information between pieces is stored in a matrix for later access. In Touch-Screen Gesture part, we design a user-friendly layout with multi-touch gestures supported to provide means in controlling and operating on the selected puzzle pieces. Finally, in the Game System part, some glueable check and merging algorithms were introduced to provide connecting between adjacent pieces or macro pieces.

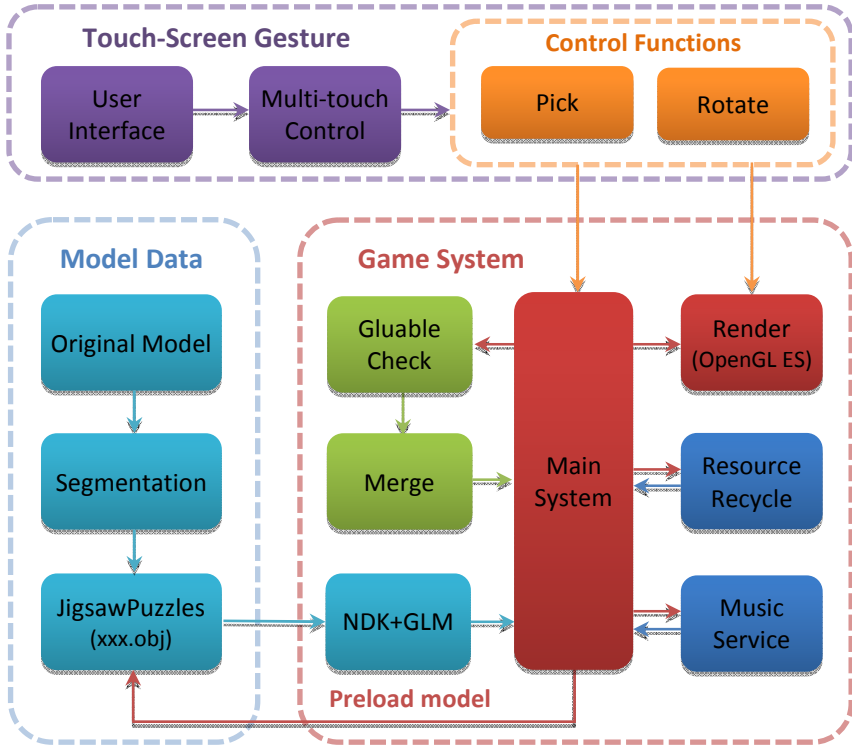


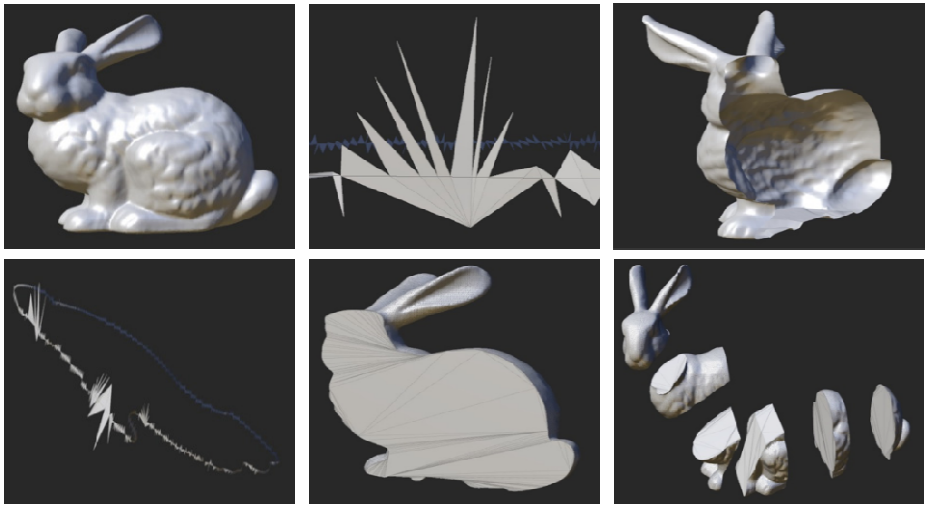
Fig. 4. System diagram

## 4 Model Data

The 3D jigsaw puzzle pieces are generated from an input 3D model by an authoring tool. Inside the authoring tool, a segmentation algorithm is proposed to cut the 3D model manually. After the segmentation, the resulting puzzle pieces are stored as individual models with an additional information file in which the adjacency graph is saved. When the user selects a model to play with, the corresponding puzzle data, including all puzzle pieces and the adjacency relation, will be loaded into the game system for display and match manipulation. The details will be discussed in the following subsections.

### 4.1 Segmentation

There have been some researches on automatic segmentation [6,7,8].For example, *Feature Sensitive Mesh Segmentation* [8] provides a way to automatically divide the model surface patches into different segments with specific constraints.However, it did not cover the interior of the 3D model after segmentation. Another example is the *Chopper* [9] which dealt with a large 3D model and decomposed it into smaller parts that are printable to fit in a 3D printer.Both methods require a lot of computing time in the optimization process. We, on the contrary, decided to perform the segmentation manually in order to simplify the authoring process and as well as to keep the flexibility on controlling the shapeof generated 3D jigsaw puzzles.



**Fig. 5.** Segmentation (top row: the input 3D model, triangles cutting, surface model after sub-division; bottom row: the circle of cutting edge, triangulation, output 3D puzzles)

First, a 3D plane is specified and positioned in a proper place to cut the model. There are triangles being intersected by the specific plane (see the middle of top row in Fig. 5).In order to extract all the triangles being intersected, we simply calculate the value of plane equation with respect to the vertices of each triangle. Thus, assume that the 3D plane equation is

$$f(x, y, z) = ax + by + cz + d \tag{1}$$

Then, we can categorize all the triangles into ten categories: +++, ++-, ++0, +--, +-0, +00, -00, --0, ---, and 000, where '+', '-', and '0' represent  $f(x, y, z) > 0$ ,  $f(x, y, z) < 0$ , and  $f(x, y, z) = 0$ , respectively.For the cases of '+++' and '---',there is no intersection occurs and the triangles are classified into positive and negative groups with respect to the plane just checked. For the cases of '++-' and '+--', the triangles have to be subdivided into new triangles and classified into positive or



negative groups as well. For the remaining cases with ‘0’ in the category, the intersection occurs at the triangle vertex (one ‘0’s) or vertices (two or three ‘0’s), the triangle is also subdivided in that the intersected vertices are duplicated for both positive and negative group. After this step, we should be able to subdivide the model into two surface meshes such as the one shown at the right of top row in Fig. 5. We need a further step to generate the surfaces that cover up the interior of the divided surface models such as the one shown at the middle of bottom row in Fig. 5. We have all the vertices on the cutting edge being sorted in monotonic sequence shown in left of bottom row in Fig. 5. Then, a concave polygon triangulation is applied to fill up surface generated by the intersection of plane and model. Although there were many triangulation algorithms [10] we can apply, a simple divide and conquer triangulation algorithm is implemented to perform the task. After these steps, a 3D model has been cut into two sub-models with the cutting cross sections being filled with triangles. For each sub-model, the segmentation process can be applied recursively to obtain more sub-models. Another reason to make the segmentation manually is that it can prevent from generating similar sub-models that are indistinguishable.

## 4.2 Loading Model Data

In order to improve the speed of loading model data from SD card into memory, Android Native Development Kit (NDK) is used for communicating between C and Java on Android system. The model data were first read in via GLM library. Then while the model data are built and vertices and faces arrays are stored as specific data structure in C, we send the array pointers to Java and convert them into the Java one through NDK. It can greatly improve the time in loading model data by 50 to 90 percent avoiding unnecessary communication between the I/O port and the Android virtual machine. Fig. 6 shows the flow to adopt NDK in the process of loading model data.



Fig. 6. Using NDK to connect GLM in C and Android Java

## 5 Touch-Screen Gesture

In order to better map the operations of real 3D jigsaw puzzle game onto the touch-screen gestures, a carefully designed GUI and intuitive multi-touch gestures are introduced.

## 5.1 Graphical User Interface

As we mentioned in previous sections that playing 3D jigsaw puzzle is different from the case of playing 2D jigsaw puzzle. Both hands are commonly required to operate on two individual puzzle pieces (or macro puzzle pieces) and searching for possible feature match between these two puzzle pieces. Besides, a complete model is consisting of several 3D puzzle pieces. Player might pick up one puzzle piece (or macro puzzle piece) and examine it through rotating the puzzle piece. He may also try to pick up two puzzle pieces at the same time and examine them by rotating both objects individually or simultaneously. In order to fit the operations on a mobile device with multi-touch gestures support, the layout of display and operation area have been carefully designed.

First, in order to let the player utilize both hands simultaneously, the landscape orientation is adopted so that both thumbs can be operated on the touch panel easily when holding the device or using two fingers of both hands to operate as well when lay down the device.

Second, the *Cover Flow*<sup>1</sup> interface is chosen for model or puzzle piece (macro puzzle piece) selection. Fig. 7 shows a snapshot of our game at model selection. Instead of randomly placing the models or puzzles pieces on a desk and showing all of them on the screen, the *Cover Flow* interface is a common technique to display multiple objects on a limited space. Users can use the left or right swipe gestures to browse the models and use the up or down swipes to change the background images. Also, because of the limited memory in mobile devices, we have implemented the model preloading process to minimize the delay in loading models. We will discuss it in the section of Game System. The browsing interface for puzzle pieces is similar to the model browsing except that you can have two selections on two puzzle pieces for later matching process.



Fig. 7. Cover flow for model browsing

<sup>1</sup> *Cover Flow* was conceived by artist Andrew Coulter Enright and originally implemented by an independent Macintosh developer, Jonathan del Strother.

Third, for manipulating on two puzzle pieces individually, the screen has been divided into left and right operating zones for the active regions of left and right puzzle pieces, respectively. So, it is intuitive to have the left thumb and right thumb to control the left and right puzzle pieces, respectively, when holding the mobile device as long as the thumbs are located in the correct operating zones. Fig. 8 shows the active operating zones in green outlines. As for the yellow outlines, they are the regions for clockwise or counter-clockwise rotation control.

Besides the GUI interfaces mentioned above, there are also some icons or hints for changing configuration, game level, or help in case you get lost in not knowing how to fix the model.

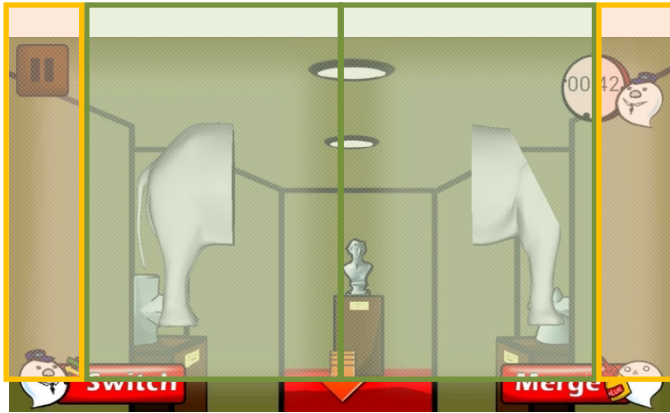


Fig. 8. Graphical user interface with multi-touch gestures supported

## 5.2 Multi-touch Gestures

The most essential operation in playing 3D jigsaw puzzle game is to search for the features that match two selected puzzle pieces so that they can be glued together to form a macro puzzle piece. Searching the features is commonly realized by rotating and examining the objects simultaneously. Gluing objects can be realized by a translation. In our system, as long as the two puzzle pieces meet the requirement of matching, they can be merged to a larger macro puzzle piece by tapping on the merge button. There is no translation operation required. For the supporting of 3D rotation operations, after selecting two potential puzzle pieces for merging, the left-right swipe indicate a rotation in  $y$ -axis; the up-down swipe indicate a rotation in  $x$ -axis; and the up-down swipe at the yellow outlines regions indicate a rotation in  $z$ -axis. No matter what orientation a puzzle piece is, it can be easily apply the supported rotation operations to the desired orientation quickly.

Common multi-touch gestures, such as scale or rotation, operate on a designated object when the subject of interest has been selected. However, in our case, we have to manipulate two subjects independently and simultaneously. So, the multi-touch gesture will be interpreted into two individual single touch operations for controlling the individual subjects at the same time. The operating zones defined in our system

help to separate a multi-touch gesture by the locations of the touches. By checking the difference between the new multi-touch locations to the previous multi-touch locations, it can be easily to identify what operations are performed with respect to the individual operating zones. The corresponding operations are then applied to the individual puzzle pieces to achieve independent control.

The common operating scenario is to use both fingers (usually they hold the device with both hands and manipulate with their thumbs) to select two jigsaw puzzles and manipulate them simultaneously. Rotate them individually until possible common match features, such as silhouette, color, texture, or cross section, are found. A merge operation can be applied to form a new macro puzzle piece. Then, the user can either to pick a new puzzle piece (or a macro puzzle piece) and trying to match with current macro puzzle piece, or simply to place the macro puzzle piece back to the cover flow and select two new puzzle pieces for matching. The process continues until all the puzzle pieces are glued together into a single completed model, just like playing a real 3D jigsaw puzzle.

## 6 Game System

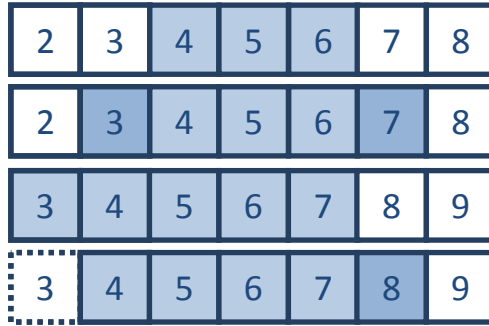
In our system, we have the model preloading step to improve the browsing performance and as well as memory management to prevent from memory leak. Besides, a merging algorithm to determine whether two puzzle pieces can be merged into one macro puzzle piece or not. The following sections illustrate how they work in the system.

### 6.1 Model Preload

The *Cover Flow* interface is a great way for model viewing. However, if we have ten models which are ready to show, for example, we could not simultaneously load the ten models into memory due to the limited resources in mobile devices. Fortunately, in *Cover Flow* we can just load the current selected model and the nearest two models at its left and right. Although, in our design, we only need three models or pieces to show on the screen, to prevent from display flashing and delaying while browsing the models, five models were preloaded for smooth transition when switching models in the *Cover Flow*. We first load the three models that are going to directly show on the screen. When the user is wondering whether left or right swipe is chosen for model selecting, the next two models at the left and right are loaded quietly. After user swipe left, for example, we quickly release the left-most model and preload the next model at the right (see Fig. 9). Thus, a smooth model transition is achieved and the memory is also well managed to prevent from overflow or leak.

### 6.2 Gluable Check and Merge

Gluable check is to determine whether two puzzle pieces can be merged to form a larger puzzle piece or not. It consists of a neighbor check and an orientation check.



**Fig. 9.** Model preloading steps (load three models first, preload the next two models at left and right, after left swipe gesture, release the 3<sup>rd</sup> model and preload the 8<sup>th</sup> model)

Only when a merge button is tapped, the gluable check is performed. For the neighbor check, in the segmentation process, we already saved the adjacent information of each jigsaw puzzle in the adjacency matrix as shown in Fig. 10. We can simply check whether two selected jigsaw puzzles are neighbor or not in the adjacency matrix. If two puzzle pieces are merged, then the new adjacency relation of the merged macro puzzle piece will be a union of the original two puzzle pieces. After ensuring the neighboring condition of the two selected puzzle pieces, we then check whether their orientations are matched with each other. Initially, all the puzzle pieces have the same orientation after segmentation. When a model is selected, all the puzzle pieces are randomly rotated to simulate the condition in spreading the pieces on the desk with arandom orientation. A rotation matrix is associated to each puzzle piece. When a user apply a sequence of rotation operations trying to make the resulting two puzzle pieces in approximately the same orientation and the merge button is tapped, the orientation check is performed. Each time a rotation operation is applied, the associated rotation matrix is updated. We retrieve the rotation matrix of the right puzzle piece and perform the inverse rotation to both right and left puzzle pieces. If the difference of orientation of the resulting left and right puzzle pieces is within an acceptable threshold (15 degree in angle), we may assume that these two puzzles pieces pass the orientation check. Fig. 11 illustrates the process of orientation check.



**Fig. 10.** Illustration of neighbor check and adjacency matrix

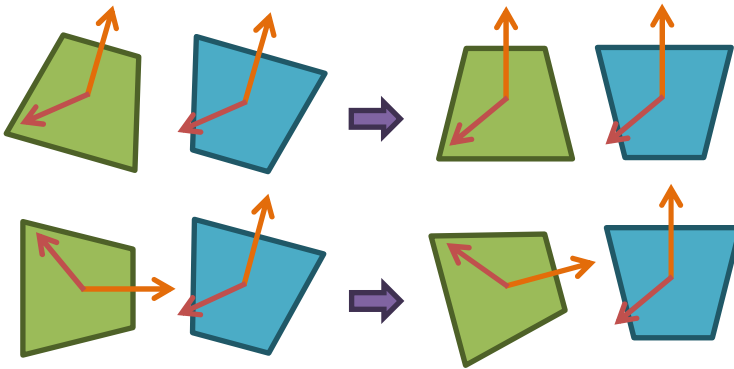


Fig. 11. Illustration of orientation check (top row: match; bottom row: mismatch)

Once the gluable check is passed, that is both neighbor check and orientation check are all passed, the two puzzle pieces are able to merge into a macro pieces. The count of the remaining pieces reduces every time a merge occurs. The game is accomplished until there is only one completed puzzle piece left.

## 7 Conclusions and Future Works

Playing a real 3D jigsaw puzzle game in real life is different from playing it in a mobile device with multi-touch gestures support in that only 2D display and touch gestures are provided. A good mapping in translating the real operations into an intuitive multi-touch gestures operations can greatly speedup the adaptation of playing the game in mobile devices. We have introduced a 3D jigsaw puzzle system, Fixit, on the mobile devices with an authoring tool to decompose a 3D model into a set of 3D jigsaw puzzle pieces; an intuitive multi-touch gestures that are easily used for performing the similar operations in real 3D jigsaw puzzle game; and a gaming system to realize the game in display, operations, and puzzle matching. We believe the implementation of a real world game into a mobile game requires a careful design in the operations it provides. The closer the operations to the real operations in real world, the larger the adoptions it will receive.

There are still some future works which require for further study. First, the authoring tool is currently a manually process. An automatic real-time jigsaw puzzle generator is definitely a promising aspect of development. Second, the gaming system provides only the shape features, such as silhouette, profile, and cross section, but lack of color and texture features. It should be able to improve the system by providing these common features. Moreover, other gestures may also apply for further improving the operations that mapped.

**Acknowledgements.** This project was supported in part by the Ministry of Science and Technology of Taiwan (MOST-103-2220-E-007-012) and the Ministry of Economic Affairs of Taiwan (MOEA-103-EC-17-A-02-S1-202).

## References

1. Erich's 3D Jigsaw Puzzles,  
<http://www2.stetson.edu/~efriedma/rubik/3jigsaw/>
2. Lo, K.Y., Fu, C.W., Li, H.W.: 3D Polyomino Puzzle. *ACM Transactions on Graphics (TOG)* 28, 157 (2009)
3. We Create Stuff: Interlocked, <http://interlocked.wecreatestuff.com/>
4. Xin, S.Q., Lai, C.F., Fu, C.W., Wong, T.T., He, Y., Cohen-Or, D.: Making Burr Puzzles from 3D Models. *ACM Transactions on Graphics (TOG)* 30, 97 (2011)
5. Song, P., Fu, C.W., Cohen-Or, D.: Recursive Interlocking Puzzles. *ACM Transactions on Graphics (TOG)* 31, 128 (2012)
6. Attene, M., Katz, S., Mortara, M., Patané, G., Spagnuolo, M., Tal, A.: Mesh segmentation—a comparative study. In: *IEEE International Conference on Shape Modeling and Applications*, pp. 7–18 (2006)
7. Kalogerakis, E., Hertzmann, A., Singh, K.: Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics (TOG)* 29(4), 102 (2010)
8. Lai, Y.K., Zhou, Q.Y., Hu, S.M., Martin, R.R.: Feature Sensitive Mesh Segmentation. In: *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, pp. 17–25 (2006)
9. Luo, L., Baran, I., Rusinkiewicz, S., Matusik, W.: Chopper: Partitioning Models into 3D-Printable Parts. *ACM Transactions on Graphics (TOG)* 31, 129 (2012)
10. Lamot, M., Balik, B.: An overview of triangulation algorithms for simple polygons. In: *IEEE International Conference on Information Visualization*, pp. 153–158 (1999)

# Towards an Intelligent Framework for Pressure-Based 3D Curve Drawing

Chan-Yet Lai and Nordin Zakaria

High Performance Computing Centre  
Department of Computer and Information Sciences,  
Universiti Teknologi PETRONAS  
Malaysia

chanyet@gmail.com, nordinzakaria@petronas.com.my

**Abstract.** The act of controlling pressure through pencil and brush appears effortless, but to mimic this natural ability in the realm of electronic medium using tablet pen device is difficult. Previous pressure based interaction work have explored various signal processing techniques to improve the accuracy in pressure control, but a one-for-all signal processing solutions tend not to work for different curve types. We propose instead a framework which applies signal processing techniques tuned to individual curve type. A neural network classifier is used as a curve classifier. Based on the classification, a custom combination of signal processing techniques is then applied. Results obtained point to the feasibility and advantage of the approach. The results are generally applicable to the design of pressure based interaction technique and possibly unlock the potential of pressure based system for richer interactions.

**Keywords:** 3D curve drawing, pressure based interaction, sketch based interface.

## 1 Introduction

As pen and touch based systems become prevalent, pressure input is now featured in many everyday devices (mobile, touch screen, digitizer and tablet computer) but with limited usage and little has known on the effectiveness of this additional input dimension. To date little work has been done on the use of pressure as an additional input channel for 3D curve drawing. Commonly, most of the previous pressure based studies [1-5] explored the use of pressure for discrete target selection, high precision parameter control [6], concurrent execution of selection and an action [7], and simultaneous manipulation of object orientation and translation [8]. In our earlier work [9], we explored a user interaction technique where pen pressure is used to generate 3D curve from a single viewpoint. The hardware we focus on for the study to provide pressure input is the pen device commonly used in the tablet system. While this technique provides direct interaction to specify a 3D curve quickly, this simple direct mapping of pressure to depth distance does not provide good pressure control. Human performance of controlling pen pressure varies at different pressure levels and under different pressure interaction scenarios. Existing one-for-all signal processing solutions fail to support different types of curve. We propose a framework which incorporates pressure-based 3D curve drawing technique that is capable of processing the recognized curve with its custom combination of signal processing techniques to deal with the unique pressure interaction issues for each curve type. The key to



this capability is a neural network classifier that has been trained to classify curves based on its pressure profile, allowing for processing steps that has been tuned for each curve class to be applied (Figure 1). Due to the use of neural network, an ‘artificial intelligence’ device, in a sense, the framework we propose here is ‘intelligent’.

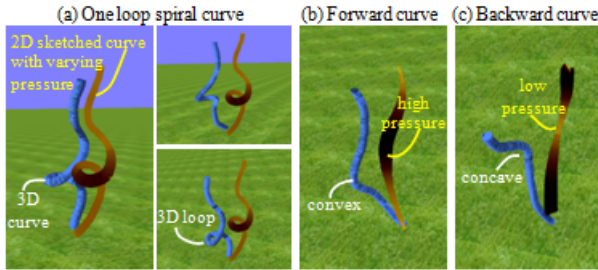


Fig. 1. The application of signal processing techniques tuned to individual curve type

## 2 Related Work

### 2.1 Pressure Based Interaction

Analog force data is converted to large number of discrete digital values through the Analog to Digital Converter (A-to-D) [1, 5]. Studies have shown that human is not capable of differentiating the granularity of this range of pressure values. The pressure based interaction studies done by researchers since 1996 [1-4, 10] have all reported that at low pressure spectrum, the difference in pressure levels sensed by the digital instrument is far greater than that the user would have expected. While at high pressure spectrum, unintended minor force exerted from finger tips tremor produces magnified variation in pressure signal. The main sources of pressure signal noise are input device background noise, physical environment noise, and hand tremor motion [11]. Ramos & Balakrishnan [6] demonstrated that the use of low pass filter and hysteresis filter work very well in mitigating signal noise and stabilizing the pressure signal. Nonetheless they concluded that there is still room for improvement by using more sophisticated filtering techniques. Common strategies to improve accuracy in pressure control include the use of a combination of noise filtering techniques [6], discretizing the raw pressure values [1, 4-6, 8] into a maximum number of discrete pressure level [1-3, 5], identifying the last intended pressure level [6], well designed visual feedback [2, 3, 6, 8, 11], and the study of human’s perception of pressure [2, 3, 6, 10, 11]. In summary, existing work on pressure-based interaction [1-6, 8, 10, 11] concluded that the adequate control of pressure is tightly coupled to the choice of pressure signal stabilization techniques, transfer function, the maximum number of discrete pressure level allowed, and visual feedback.

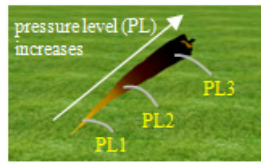
### 2.2 Classification of Waveform Using Neural Network

Artificial neural network (NN) enjoyed very extensive applications in various domains that require automated recognition or classification. The decision making in NN is holistic [12]. A great variety of NN has been experimented in the field of medical for effective classification of electrocardiogram (ECG) [12, 15-19, 21], electroencephalograph (EEG) [20], and Doppler waveform [13, 14] to diagnose disease. In financial market, it has been explored for Elliot waveform recognition [22] to predict future trends. All reported good recognition rate with correct classification in over 90% of the cases and as high as 100% classification rate were reported for Doppler [13, 14] and Elliot waveform classification [22]. As a comparison,

alternative waveform classifiers such as beat classifier, digital filter, linear and non linear methods were evaluated, but all generalize poorly [16, 17, 19, 21]. The characteristics of bio-signals are non stationary, contaminated with noise, and have large variation in the morphologies of the waveform within the same patient [12, 16, 17, 21]. Successful applications of neural network in previous works [12-22] for bio signal and waveform classification have demonstrated that the generalization ability of neural network is relatively robust in the presence of noise and the variation of waveform.

### 3 Pressure-Based 3D Curve Drawing

We assume the 3D curve interaction technique in [9]. User makes use of pen pressure to sketch a 2D curve with varying thickness. The system maps the thickness at each point along the 2D curve to a depth distance. Hence, the thicker the 2D curve at a particular point, the closer the corresponding 3D point to the camera. With this technique an approximate 3D curve can be sketched and edited directly. More details on the interaction scheme is in [9]. In this study, the amount of pressure applied is indicated by real-time feedback that mimics the effect of a paint brush, as shown in Figure 2.



**Fig. 2.** The shade and brush stroke size increase as the pressure increases.  $PL1 < PL2 < PL3$ .

### 4 Controlling Pressure

We investigate the viability of having different processing steps for different curve type by examining the effect of applying existing signal processing techniques on a basic set of curves. This basic set of curves is one loop spiral curve, forward (convex) curve and backward (concave) curve. The choice of basic set of curves is inspired by the naturally formed curves commonly found in botanical shape. The existing signal processing techniques experimented are low pass filter, sigmoid function, fisheye function and hysteresis filter.

Drawing a forward curve requires fine pressure control at high pressure range to specify the convexity of the curvature. We observed that the application of high pressure spans across the neighboring pressure signals of intended convex area. This observation is in agreement with Mizobuchi et al. 2005 [2] findings which found that it is difficult for participants to control and maintain constant pressure at high pressure level. On the contrary, fine pressure control at low pressure range is needed in specifying the concavity of a backward curve. Numerous previous works [1-3] reported that at low pressure range, participants demonstrated poor pressure control, the pen is perceived to be ‘too sensitive’. These results agree with our earlier observation, which shows that it is difficult to control low pressure range to achieve the desired concave curvature. The challenge faced in drawing a one loop spiral is to perform symmetric bi-directional pressure control to create the 3D circular shape of the loop. Fingertips tremors greatly distort the circular shape users intend to draw.

Other than the curve specific pressure control challenge described above, there are mechanical issues with the pressure sensitive pen. Unwanted pressure input is captured when the pen lands on the sensing surface before it reaches the desired start drawing pressure, which we refer

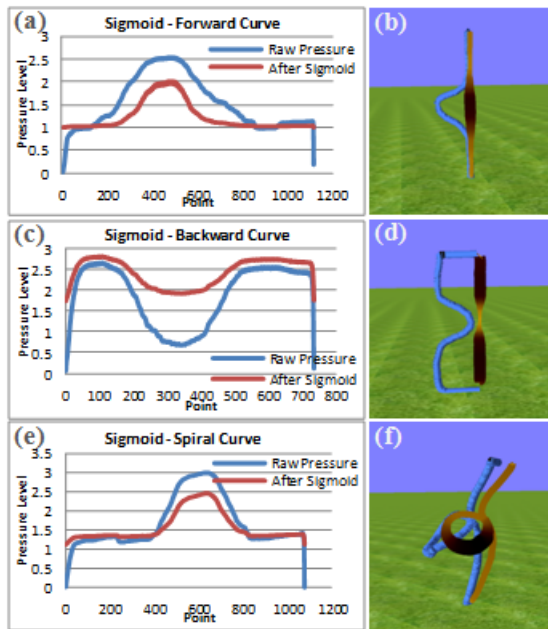
as a landing effect in this study. Pressure is being captured throughout the action of lifting the pen. As a result it leaves a trail of sudden drop in pressure value. We refer to such effect as landing effect. Similar landing effect was reported in [6]. Both landing and lifting effects described have produced significant distortion to the beginning and ending of all curves.

#### 4.1 Low Pass

The use of low pass filter has been proposed by Ramos et al. 2005 [6] to mitigate the background noise of the pressure input device. Through our experimental observation, the application of low pass filter smooths the pressure signal but introduces phase shift at the same time. Therefore, low pass filter is used for minor smoothing in all three curves.

#### 4.2 Sigmoid

It has been highlighted in previous works [4, 6, 10] that the use of parabolic-sigmoid transfer function improves user feel when applying force through pressure sensitive input devices. This transfer function has slow response at low pressure levels, linear behavior at middle pressure levels and slow response at high pressure levels. This effect is similar to the effect we want to achieve to mitigate the difficulty in pressure control at low and high pressure levels. The conventional sigmoid function was modified to include a contrast factor and threshold value. The modified sigmoid function can be tuned for fine pressure control at high or low pressure range.



**Fig. 3.** Pressure profiles processed with sigmoid function and 3D curves generated

Through a series of experiments, we found that when high contrast factor and high threshold value were used, the application of high pressure is focused and the pressure near the lower end of its range reaches a plateau value, as shown in Figure 3a and 3b. The use of low contrast

factor and low threshold value stabilizes pressure value at low pressure range and the variation in pressure input near the upper end stays considerably constant, as shown in Figure 3c and 3d. Unfortunately none of the configurations succeed in providing good symmetric bi-directional pressure control to create a one loop spiral. The circular shape of the loop narrowed as it moves towards the convex area as a result of focused pressure control (see Figure 3f).

### 4.3 Fisheye and Hysteresis

Fisheye transfer function is the only transfer function that has been evaluated for bi-directional pressure control, mainly for discrete target selection [5] and zooming purposes [23]. Both studies [5, 23] concluded that fisheye transfer function exhibits greater accuracy in bi-directional pressure control as compared to other transfer functions. In this study, the discrete fisheye function proposed in [5] was modified for continuous pressure control. Ramos & Balakrishnan 2005 [6] proposed the use of hysteresis filter to mitigate muscular tremor and environmental noise. Our hysteresis implementation is based on the hysteresis filter proposed in [6].

Our pilot study shows that both techniques share the commonality of stabilizing the pressure signal by suppressing it. Figure 4 compares the application of both techniques on each curve type. Consistent with previous research findings in both [5] and [23] the use of fisheye function provides good bi-directional pressure control. Hysteresis filter achieves pressure stabilization with larger pressure suppression as compared to fisheye function. User tends to exhibit better control of pressure due to the reduced sensitivity in pressure response. The major drawback is that the reduced sensitivity in pressure response creates a false impression that the intensity of the convexity and concavity created does not conform to the amount of pressure applied. Fisheye function provides better sense of pressure mapping to depth distance and is able to create rounded curvature. The experimental findings while preliminary suggest that pressure suppression is necessary to increase the accuracy in pressure control.

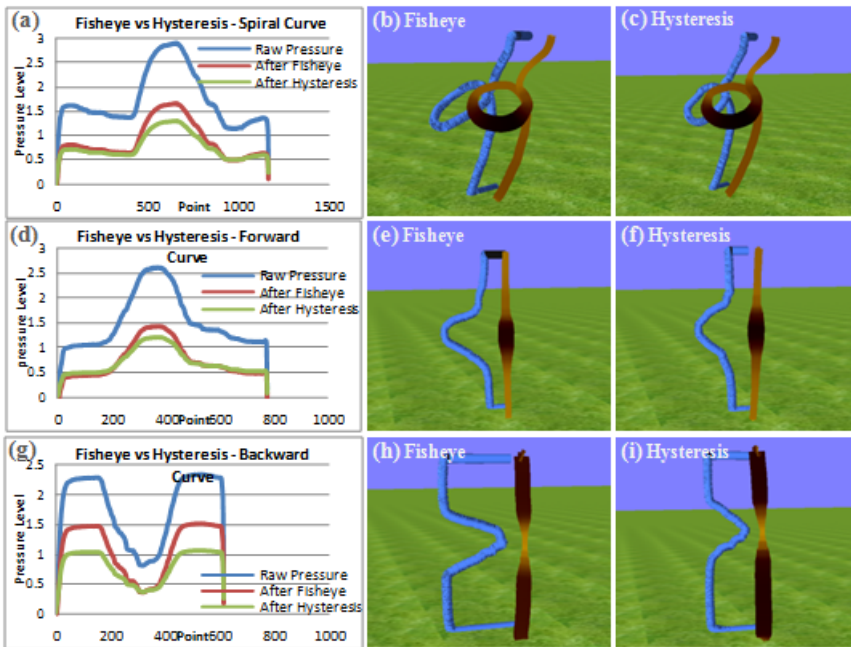


Fig. 4. Comparison of the effect of fisheye function and hysteresis filter

### 4.4 Combination of Signal Processing Techniques

#### Spiral Curve.

The existing signal processing techniques even when they are used in combination are not adequate to mitigate the pressure interaction challenges faced in sketching a spiral curve. We designed a heuristic approach to be used in combination with existing signal processing techniques. The pressure signal first passes through a low pass filter then a heuristic approach is adopted to smooth the pressure values of the curve outside the loop towards the vicinity of the median pressure value to ensure that the start and end of the loop always cross at the same depth distance. Observations in our studies reveal that the intended pressure at area where the curve crosses itself to form the loop is usually the median pressure value of the sorted spiral curve pressure profile. Lastly, fisheye transfer function is utilized to reduce the pressure differences moderately. Pressure suppression (fisheye function) is performed last to avoid over processing the pressure which will lead to losing drawing details. Figure 5 shows the effect of applying the various signal processing steps on the pressure profile. Figure 6 shows 3D one loop spiral curves generated from the processed pressure profile. The spiral curve in Figure 6a1 to 6a4 has been rotated counter clockwise around its y-axis. From the side view, the curve crosses itself to form a loop. It can be clearly seen from various angle of top view, the loop has a circular shape, as shown in Figure 6a5 and 6a6. Figure 6b and c show variations of one loop spiral curves. Their side view and top view are presented in Figure b2 & c2 and b3 & c3 respectively.

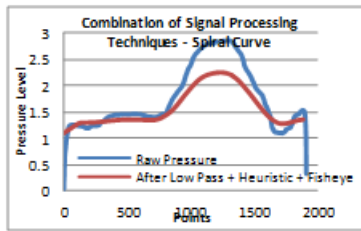


Fig. 5. Effect of the combination of processing steps on the spiral curve pressure profile

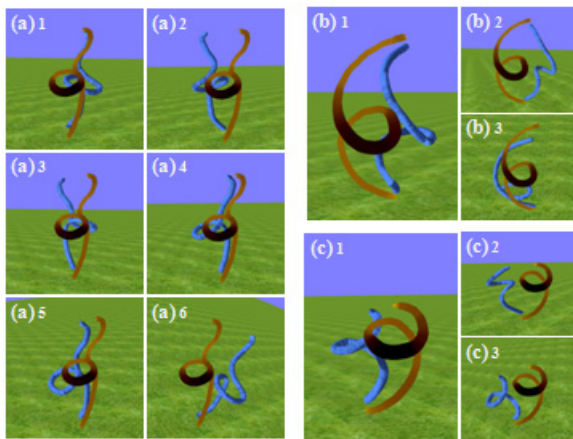


Fig. 6. Examples of 3D spiral curves created through the processed pressure signals

### Forward Curve.

The combination of signal processing techniques is applied in the following order: i) low pass filter ii) sigmoid function and iii) fisheye function. We present an example of how the signal processing stages affect the final pressure profile in Figure 7. Examples of 3D forward curves which had their pressure profiles processed with the proposed combination of signal processing techniques are presented in Figure 8. From the side view in Figure 8a2 and 8a3, it is obvious that the application of high pressure is focused. The stabilize pressure signals enable user to maintain a constant pressure level, which has been regarded in previous study as an impossible task without the help of well designed interactive techniques. Figure 8b shows variation of 3D forward curve created.

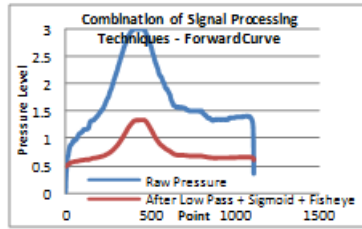


Fig. 7. Effect of the combination of processing steps on the forward curve pressure profile

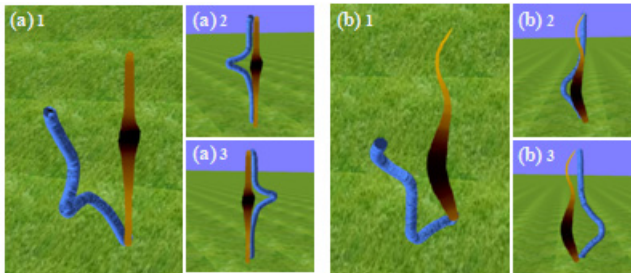


Fig. 8. (a)1 & b(1) Top view (a)2 & (b)2 Side view, rotated 90° clockwise around y axis (a)3 & (b)3 Side view, rotated 90° counter clockwise around y axis

### Backward Curve.

Following the success application of low pass filter, sigmoid function and fisheye function in combination to achieve good accuracy of pressure control in drawing a forward curve, the same experiment was repeated for the backward curve. With the right configurations of sigmoid and fisheye functions tuned towards backward curve, we are able to achieve fine accuracy of pressure control in specifying a backward curve but show no improvement on the landing and lifting effects. In addition to the processing steps applied, an algorithm was designed to mitigate the landing and lifting effects. Through our observation, the start drawing pressure is usually the first pressure value encountered after a steep increasing trend at the beginning of the pressure profile. While it is apparent from all pressure profile that when the pen lifting action occurs (stop drawing), from that point onwards the pressure values decrease abruptly. We replace the increasing trend and decreasing trend with the identified start drawing pressure and stop drawing pressure respectively. This simple algorithm works well in mitigating the landing and lifting effect. An example of the processed pressure profile is presented in Figure 9. Figure 10

presents examples of 3D backward curve created. Both side views in Figure 10a2 and 10a3 show that we are able to achieve high precision pressure manipulation at low pressure levels. Further the result demonstrates that it is feasible to maintain a constant high pressure level, a pressure control challenge which has been highlighted in [2].

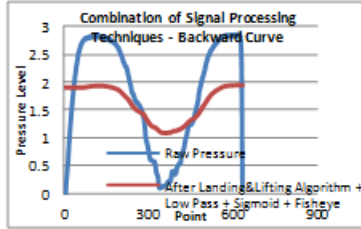


Fig. 9. Effect of the combination of processing steps on the backward curve pressure profile

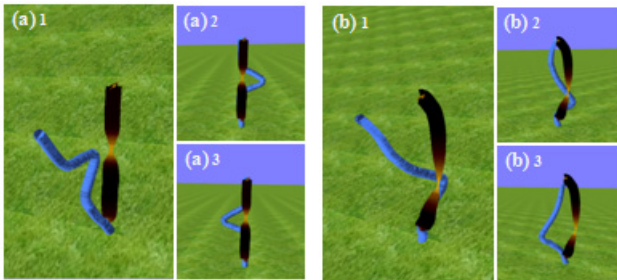


Fig. 10. (a)1 & (b)1 Top view (a)2 & (b)2 Side view, rotated 90° clockwise around y axis (a)3 & (b)3 Side view, rotated 90° counter clockwise around y axis

The findings above suggest that different signal processing technique is needed for different type of curve. Observation from the study shows that it is more difficult to control pressure in a decreasing manner as compared to increasing manner. This finding is in agreement with previous studies [1, 7, 24] which highlighted the limitation in human motor to control pressure from positive to zero. Interestingly, the current study finds that higher level of difficulty was experienced when controlling compound pressure in high → low → high manner (backward curve) as compared to low → high → low manner (forward and spiral curve).

## 5 Pressure-Based 3D Curve Recognition Using Neural Network

A total of 181 training data (65 forward, 67 backward and 49 spiral) was acquired from 5 volunteers. The training data was pre-conditioned with Fast Fourier Transform (FFT). The network consists of 50 input neurons and 3 output neurons, each represents a curve class. In contrast to earlier studies [14, 17, 19-22], however, our training dataset does not converge well using the conventional backpropagation training algorithm. Lower error rate was obtained using adaptive batch training algorithm. Similar to previous waveform pattern recognition studies [12-14, 16-18, 20], the optimum network architecture was determined via experiment. The architecture of the neural network was examined using one and two hidden layers. The optimum architecture obtained is 50:35:3. It is able to achieve 100% correct classification rate for backward curve, 98% classification accuracy for forward curve and 95% of the spiral curves are successfully classified. In our study, one hidden layer is adequate to solve the learning

problem without losing its generalization ability. This finding corroborates the ideas of Sternickle 2002 [19] and Subasi et al. 2005 [20], who suggested that one hidden layer is adequate to solve most of the problem. The optimum number of hidden neurons obtained is 35, this result agreeing with the approximate number of hidden neurons determine through the rule-of-thumb methods which suggested that the optimum number of neurons should be between the number of input neurons and output neurons; approximately  $2/3$  of the total input neurons and output neurons [26] and should not be greater than twice the number of input neurons [25].

## 6 Conclusion

In this paper, we propose a *draw-recognize-process* framework for 3D curve drawing that exploit curve pressure profiles. This is in fact the underlying principle of sketch-based modelling system (such as [27]). We introduce the use of context dependent signal processing and filtering techniques to facilitate the use of pressure to draw 3D curves with a pressure sensitive pen. The contribution is significant especially when taking into consideration the difficulty of projecting 2D curves to 3D based on its pressure profiles. Our current implementation supports three curve patterns: convex, concave, and spiral. It is probable that other curve patterns can be supported in a similar manner.

One limitation of the current system is that our neural network does not support curves that can change in style in one continuous trace. However, the results of the preliminary study show that the crafted combination of signal processing techniques is able to handle variation of curve of its type. An interesting direction would be to extend our work to support application customized sketch-based interface in modelling targets such as trees.

## References

1. Cechanowicz, J., Irani, P., Subramanian, S.: Augmenting the mouse with pressure sensitive input. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2007)
2. Mizobuchi, S., Terasaki, S., Keski-Jaskari, T., Nousiainen, J., Ryyanen, M., Silfverberg, M.: Making an impression: force-controlled pen input for handheld devices. In: CHI 2005 Extended Abstracts on Human Factors in Computing Systems. ACM (2005)
3. Ramos, G., Boulos, M., Balakrishnan, R.: Pressure widgets. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2004)
4. Ren, X., Yin, J., Zhao, S., Li, Y.: The adaptive hybrid cursor: A pressure-based target selection technique for pen-based user interfaces. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4662, pp. 310–323. Springer, Heidelberg (2007)
5. Shi, K., Irani, P., Gustafson, S., Subramanian, S.: PressureFish: a method to improve control of discrete pressure-based input. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2008)
6. Ramos, G., Balakrishnan, R.: Zliding: fluid zooming and sliding for high precision parameter manipulation. In: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology. ACM (2005)
7. Ramos, G.A., Balakrishnan, R.: Pressure marks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2007)



8. Shi, K., Subramanian, S., Irani, P.: PressureMove: Pressure Input with Mouse Movement. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) *INTERACT 2009*. LNCS, vol. 5727, pp. 25–39. Springer, Heidelberg (2009)
9. Lai, C.-Y., Zakaria, N.: Pressure-based 3D curve drawing. In: Taylor, R., Boulanger, P., Krüger, A., Olivier, P. (eds.) *Smart Graphics*. LNCS, vol. 6133, pp. 156–159. Springer, Heidelberg (2010)
10. Barrett, R.C., Olyha Jr., R.S., Rutledge, J.D.: Graphical user interface cursor positioning device having a negative inertia transfer function. Google Patents (1996)
11. Ramos, G.A.: Pressure-sensitive pen interactions. University of Toronto (2008)
12. Rajendra Acharya, U., et al.: Classification of heart rate data using artificial neural network and fuzzy equivalence relation. *Pattern Recognition* 36(1), 61–68 (2003)
13. Ceylan, M., Ceylan, R., Dirgenali, F., Kara, S., Ozbay, Y.: Classification of carotid artery Doppler signals in the early phase of atherosclerosis using complex-valued artificial neural network. *Computers in Biology and Medicine* 37(1), 28–36 (2007)
14. Ceylan, R., Ceylan, M., Ozbay, Y., Kara, S.: Fuzzy clustering complex-valued neural network to diagnose cirrhosis disease. *Expert Systems with Applications* 38(8), 9744–9751 (2011)
15. Engin, M.: ECG beat classification using neuro-fuzzy network. *Pattern Recognition Letters* 25(15), 1715–1722 (2004)
16. Osowski, S., Linh, T.H.: ECG beat recognition using fuzzy hybrid neural network. *IEEE Transactions on Biomedical Engineering* 48(11), 1265–1271 (2001)
17. Özbay, Y., Ceylan, R., Karlik, B.: A fuzzy clustering neural network architecture for classification of ECG arrhythmias. *Computers in Biology and Medicine* 36(4), 376–388 (2006)
18. Özbay, Y., Tezel, G.: A new method for classification of ECG arrhythmias using neural network with adaptive activation function. *Digital Signal Processing* 20(4), 1040–1049 (2010)
19. Sternickel, K.: Automatic pattern recognition in ECG time series. *Computer Methods and Programs in Biomedicine* 68(2), 109–115 (2002)
20. Subasi, A., Erçelebi, E.: Classification of EEG signals using neural network and logistic regression. *Computer Methods and Programs in Biomedicine* 78(2), 87–99 (2005)
21. Tezel, G., Özbay, Y.: A new neural network with adaptive activation function for classification of ECG arrhythmias. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) *KES 2007, Part I*. LNCS (LNAI), vol. 4692, pp. 1–8. Springer, Heidelberg (2007)
22. Kotyrba, M., Volna, E., Brazina, D., Jarusek, R.: Elliott waves recognition via neural networks. In: *Proceedings 26th European Conference on Modelling and Simulation, ECMS* (2012)
23. Mandalapu, D., Subramanian, S.: Exploring pressure as an alternative to multi-touch based interaction. In: *Proceedings of the 3rd International Conference on Human Computer Interaction*. ACM (2011)
24. Rekimoto, J., Schwesig, C.: PreSenseII: bi-directional touch and pressure sensing interactions with tactile feedback. In: *Extended Abstracts on Human Factors in Computing Systems, CHI 2006*. ACM (2006)
25. Boger, Z., Guterman, H.: Knowledge extraction from artificial neural network models. In: *1997 IEEE International Conference on Systems, Man, and Cybernetics*. Computational Cybernetics and Simulation. IEEE (1997)
26. Karsoliya, S.: Approximating number of hidden layer neurons in multiple hidden layer BPNN Architecture. *Internat. J. Eng. Trends Technol.* 3(6), 714–717 (2012)
27. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3D freeform design. In: *ACM SIGGRAPH 2007 Courses*. ACM (2007)

# Analysis of Visual Elements in Logo Design

Wen-Hung Liao and Po-Ming Chen

Department of Computer Science, National Chengchi University  
Taipei, Taiwan  
{whliao, radination}@gmail.com

**Abstract.** A logo is a mark composed of graph or a combination of text and graph. Typical visual elements in a logo design such as layout, shape, color, composition, and typeface. The graphical mark can exhibit interesting properties by mixing the elements in creative ways. Most previous researches regarding the role of visual elements in logo design are of qualitative nature. In this paper, we propose to incorporate visual feature extraction and analysis algorithms commonly utilized in computer vision to compute proper index and investigate key visual elements in logo design, including complexity, balance and repetition. After analyzing large amount of logos collected from the internet, we find out that most logos are of low complexity, high balance and exhibit minor degree of repetition. We hope that the results obtained in this research serve as a catalyst to motivate further efforts in applying computer vision methods to the area of aesthetics and design.

**Keywords:** Logo Design, Aesthetic Measure, Visual Elements, Image Features.

## 1 Introduction

With the rapid advancements in technology, we have witnessed the extension and practical application of once academically oriented research into realms that affect our daily life. Computer vision is one such example. Continuing innovations in computer vision and image analysis have led to success in tasks such as face detection [1], object recognition [2], human computer interaction [3], and 3D scene reconstruction [4]. Although the ultimate goal of image understanding is yet to be realized, the progress of computer vision in the last few decades is indeed quite impressive. More application domains will certainly be exploited to take advantage of the powerful tool and algorithms already devised.

Within the scope of computer vision research, art analysis is one subject that has gained attention only recently. Stork has taken computational approaches to the study of paintings and drawings [5]. New insights into historical paintings have been gained through rigorous mathematical analysis, revealing details that were imperceptible to human observers. Assessing the aesthetic quality of photo using computer vision techniques is also a surging topic due to the prevalence of image capturing devices [6]. More generally, researchers in the area of computational aesthetics attempt to develop computational methods that can make aesthetic judgments as humans do [7].

This paper focuses on the analysis of visual elements for logo design. Logo is a form of visual art. It is created by mixing simple geometric shapes, text and color to form a mark that is usually associated with the identity of an organization or corporation. Most existing literatures on logo are concerned with the ‘design’ aspect. Few materials that dealt with the analysis of content are qualitatively oriented [8]. Therefore, the main objective of this study is to incorporate visual feature extraction and analysis algorithms to compute proper index and investigate key visual elements in logo design, including complexity, balance and repetition. To observe the roles of these visual features, we have collected around 26K logos from the Internet to perform quantitative analysis and comparative studies.

The rest of this paper is organized as follows. In section 2, we review past research in applying computer vision techniques to art analysis, with special emphasis on the extraction of relevant visual features. Section 3 presents three mid-level visual features employed in this investigation, namely, balance, complexity and repetition. We discuss how these features can be computed from the input image and why they are considered significant in the logo design. Section 4 depicts experimental results of analyzing visual features from the logo dataset. Section 5 concludes this paper with a brief summary and outlook on future investigation.

## 2 Related Work

Previous research in applying computer vision techniques to art analysis often involves the computation of image features and identification of their roles in art appreciation. Many visual features can be extracted from an image. They include low level features such as color, brightness, edge, corner and blob. Mid-level features or factors including balance, proportion, composition, and complexity have been shown to exhibit strong relationship with aesthetic judgments. For complex artistic expressions in paintings and drawings, preprocessing steps are usually required to obtain a robust mid-level representation. In contrast, the material examined in this research, i.e., logo, has a simpler structure. Therefore, mid-level features can be computed directly from the image. Such a simplification has made it feasible for us to explore a much larger collection of samples compared to those reported in previous work [9].

Balance is considered to play a critical role in the organizational structure of visual images. In [10], Wilson et al. proposed a method to compute balance scores for black-and-white stimuli. The balance score gauge the symmetry along four principal axes: horizontal, vertical and two diagonal directions. Additionally, inner and outer regions are defined to measure the distribution of black pixels in different areas. The experimental results indicate that the objective balance score is highly correlated to subjective preferences for the 130 images employed in their study. We also consider balance to be a key element in logo design. However, the original balance score is defined only for black-and-white images. As a result, we will modify and extend the definition of balance score to suit our purpose.

Complexity is also an important factor in aesthetic evaluation. Birkhoff defined the aesthetic measure as the ratio between order and complexity [11]. When interpreted

from the view point of information theory, Shannon entropy can be employed to estimate the complexity of a signal. In this research, we follow a similar approach to compute the entropy [12]. Yet we observe that the complexity of an image should be examined from multiple perspectives. Toward this end, we have defined three metrics to measure the complexity of a logo.

Repetition attempts to quantify the occurrence of similar visual components in an image. In logo analysis, this corresponds to the discovery of similar geometric shapes. There exist many methods to extract and describe geometric shapes from an image, including Hough transform, shape context [13], and Fourier descriptor. In this research, we chose Fourier shape descriptor due to its efficiency and invariance with respect to rotation, translation and scaling. A repetition score based on similarity measures of major constituent shapes will then be defined.

### 3 Visual Elements in Logo Design

Many mid-level visual features have been proposed to characterize the aesthetic properties of visual arts. Since we are restricting ourselves to the study of logo, we will consider three mid-level visual attributes that are deemed important in the composition of logo. We have deliberately left out one essential attribute: color in this study as we believe that this topic deserves further investigation to arrive at some concrete conclusions. In the following, we will introduce balance, complexity and repetition and define proper formula for computing these quantities from input images.

#### 3.1 Balance

Balance is often associated with symmetry. Indeed, the simplest form of balance is symmetry. However, balance can be achieved when the visual forces of asymmetrically arranged elements compensate for each other. It is therefore a more general concept than the notion of symmetry. The balance score defined by Wilson et al. is calculated using Eqs. 1-4, and the partition styles are depicted in Fig. 1.

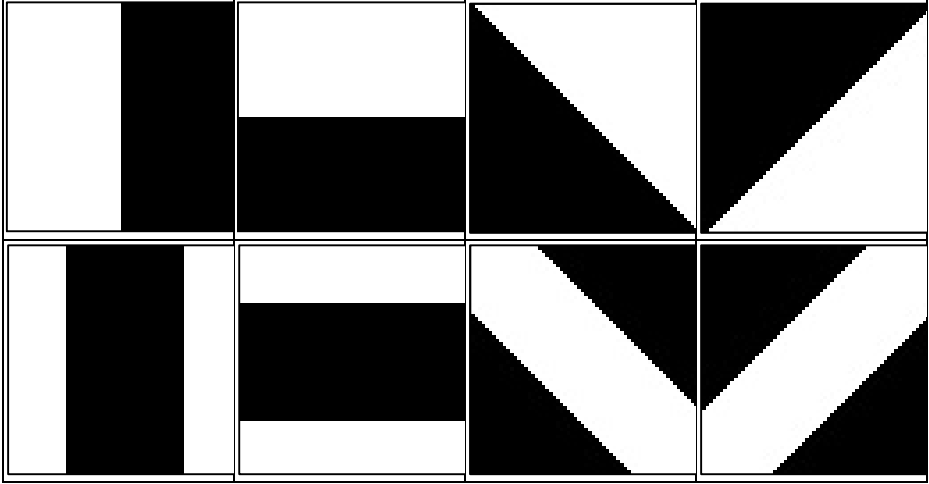
$$S_1 = \sum_{width} \sum_{height} 1 \text{ if } (x, y) \in A_1 \quad (1)$$

$$S_2 = \sum_{width} \sum_{height} 1 \text{ if } (x, y) \in A_2 \quad (2)$$

$$R_i = \frac{|S_1 - S_2|}{S_1 + S_2} \text{ (} i = 1, \dots, 8 \text{ corresponds to different partition styles)} \quad (3)$$

$$Balance = \frac{1}{8} \sum_{i=0}^7 R_i \quad (4)$$

The original formula defined above can only cope with black-and-white images. We propose the following modifications shown in Eqs. 5-6 to compute balance score from grayscale images.



**Fig. 1.** Eight region partitions for computing the balance score (white:  $A_1$ , black:  $A_2$ )

$$S_1 = \sum_{width} \sum_{height} \overline{g(x,y)} \text{ if } (x,y) \in A_1 \quad (5)$$

$$S_2 = \sum_{width} \sum_{height} \overline{g(x,y)} \text{ if } (x,y) \in A_2 \quad (6)$$

$$\text{where } \overline{g(x,y)} = 255 - \text{gray}(x,y)$$

For color images, we can utilize RGB color space and compute the balance of individual channel using the above formula. However, we will get three different balance scores in this manner. It is challenging to interpret the combined effect of these scores as the role of ‘color’ is coming into play. To exclude the color factor, we will adopt the HSV representation and define the balance score using only the S (Saturation) and V (Value) components in this research. The newly modified balance measure becomes:

$$R_i = \frac{|S_1 - S_2|}{S_1 + S_2} * \omega + \frac{|V_1 - V_2|}{V_1 + V_2} * (1 - \omega) \quad (7)$$

where

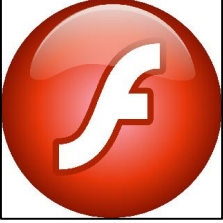
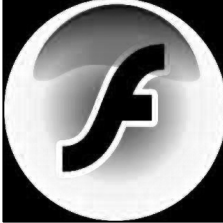
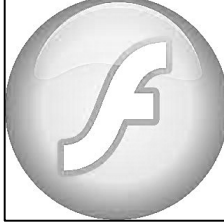



$$S_1 = \sum_w \sum_h s(x,y) \text{ if } (x,y) \in A_1 \quad (8)$$

$$S_2 = \sum_w \sum_h s(x,y) \text{ if } (x,y) \in A_2 \quad (9)$$

$$V_1 = \sum_w \sum_h \overline{v(x,y)} \text{ if } (x,y) \in A_1 \quad (10)$$

$$V_2 = \sum_w \sum_h \overline{v(x,y)} \text{ if } (x,y) \in A_2 \quad (11)$$

The symbol  $\omega$  in Eq. 7 denotes the weight assigned to the saturation component. In all our experiments, we set  $\omega$  to 0.5. Fig. 2 shows two logos and their corresponding balance score. Notice that the balance score thus defined indicates more balance when the value is closer to 0, and signifies imbalance when the value is approaching 1.

Original	Saturation	Value
		
		

**Fig. 2.** Top logo:  $R=\{0.005,0.13,0.112,0.106,0.06,0.106,0.135,0.163\}$ , Balance=0.102, Bottom logo  $R=\{0.154,0.416,0.181,0.370,0.048,0.081,0.064,0.12\}$ , Balance=0.180

### 3.2 Complexity

Complexity measure is directly related to entropy using information-theoretic modeling. Higher entropy corresponds to more complex configuration in an image. The discrete version of entropy can be expressed as Eq. 12:

$$H = -\sum_i p_i \log_2 p_i \tag{12}$$

In this study, we employ a top-down image partition algorithm to segment the logo into homogeneous regions with similar brightness value. A low complexity image will contain only a few large homogeneous blocks, and vice versa for a high complexity image. The partition can be applied vertically or horizontally. We will iterate through all columns and rows and compute the corresponding entropies, as shown in Eq. 13-14. The entropy  $H$  is used to determine whether the partition should continue. If the minimum of the sum of the entropies of two partitioned regions is smaller than a threshold  $\theta$ , the segmentation process stops. Otherwise the partition proceeds according to Eq. 15. The flowchart of the proposed image partition algorithm is depicted in Fig. 3.

$$E_{1i} = \lambda_{1i} \cdot H(I(0, i)) + (1 - \lambda_{1i}) \cdot H(I(i + 1, W))$$

$$\lambda_{1i} = (i + 1)/W \quad (13)$$

$$E_{2j} = \lambda_{2j} \cdot H(I(0, j)) + (1 - \lambda_{1i}) \cdot H(I(j + 1, H))$$

$$\lambda_{2j} = (j + 1)/H \quad (14)$$

$$P = \text{Argmin}(H * E_{1i}, W * E_{2j}) \quad i = 0 \sim W - 1, j = 0 \sim H - 1 \quad (15)$$

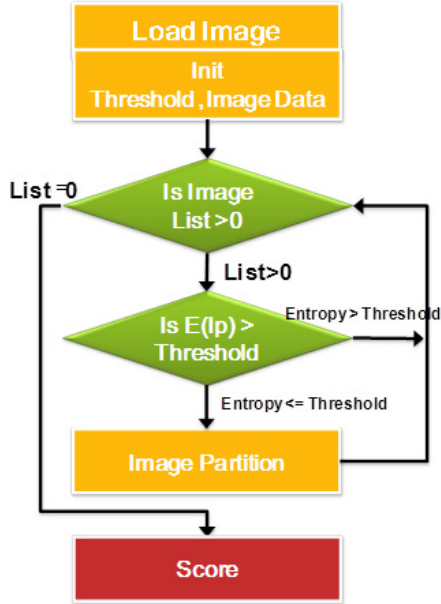


Fig. 3. Flowchart of the proposed image partition algorithm

Once the partition is done, the complexity score can be computed accordingly. However, we notice that there exist several possible interpretations of the partitioned result. The first complexity measure, denoted as *partition-based complexity*, is directly related to the distribution of the area of the partitioned segments and can be computed using Eq. 16. If the sizes of the partitioned regions are diverse, the image is thought to possess higher complexity.

$$C_p = H(\text{Size}(\text{Area}_i)) \quad (16)$$

Secondly, since the direction of partition is either vertical or horizontal, a homogeneous yet irregularly shaped region may be divided into smaller segments. It would be unfair to calculate the complexity using only the partitioned area. Instead, we can examine the distribution of the entropy of the segments and defined another complexity measure, denoted as *homogeneity-based complexity*, based on the homogeneity of the entropy of segmented regions as shown in Eq. (17).

$$C_f = H(H(Area_i)) \tag{17}$$

Finally, the total number of partitions with respect to the image size also reflects the complexity of a logo. Therefore, we define the third metric denoted as *area-ratio complexity* which can be computed using Eq. 18.

$$C_r = \frac{R}{N} \text{ where } R = \text{Total Number of Partitions}, N = W * H \tag{18}$$

Fig. 4 illustrates the necessity of having different complexity measures as a single metric may not effectively reveal the structure of the logo image.


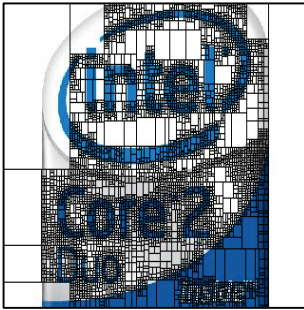
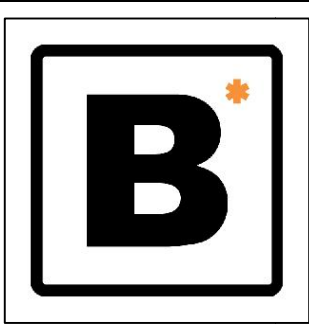
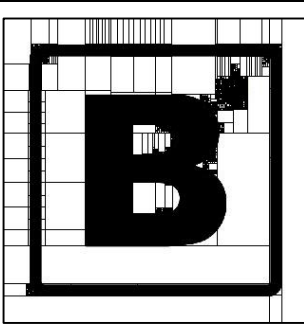
Input Logo	Partition Result	Complexity
		$C_p=3.94$ $C_f=5.41$ $C_r=3.7\%$
		$C_p=5.45$ $C_f=3.6$ $C_r=0.39\%$

Fig. 4. Complexity measures based on different criteria

### 3.3 Repetition

The repetition score can be computed by comparing the constitution of principal contours in a logo. We propose to use Fourier descriptor to represent the major contours and perform subsequent similarity analysis. However, topological structures have to be extracted from the images first. This is done using the following three steps:

1. Apply Gaussian smoothing to eliminate boundaries that are too close
2. Employ Canny edge detector to candidate boundary pixels
3. Use topological structural analysis to find the contours [14]



Since the number of contours thus extracted can be large, we use two criteria to reduce the number to a maximum of 16: (1) the area enclosed has to be greater than 0.5% of the total area, and (2) only the top 16 contours with largest enclosed area are retained. Fig. 5 depicts the final contours extracted from a logo image with gradual changes in the boundary.

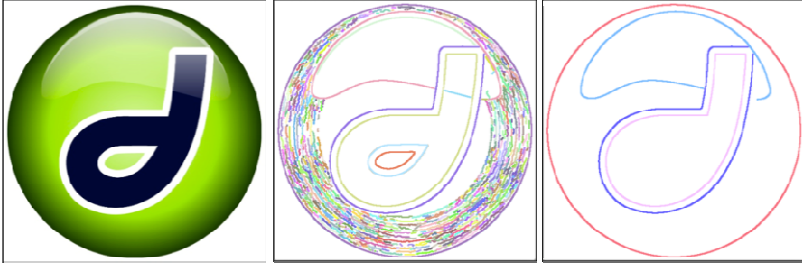


Fig. 5. Boundary extraction using the proposed approach

The 2-D boundary is converted into a 1-D representation using shape signature, where the centroid of the boundary is calculated and the distance between the centroid and the boundary is encoded, as illustrated in Fig. 6. The 1-D signal is then converted into frequency domain using Fourier transform. These coefficients will be normalized and the similarity between two shape contours will be calculated using earth mover’s distance [15]. The proper combination of representation scheme and distance metric will enhance robustness against rotation, translation and scaling.

Finally, to derive a score that quantify the degree of repetition, we need to have some basis of comparison. The statistics are obtained by computing the mean and standard deviation of the EMD of all samples in the dataset. If the deviation from the mean is less than half of the standard deviation (shown in Eq. 19), the two contours  $C_i, C_j$  are considered to be comparatively similar. The degree of similarity is quantized into four levels according to Eq. 20.

$$\Delta = \mu - \sigma/2 \tag{19}$$

$$SL = \begin{cases} 1 & EMD(C_i, C_j) < \Delta/4 * 1 \\ 2 & \Delta/4 < EMD(C_i, C_j) < \Delta/2 \\ 3 & \Delta/2 < EMD(C_i, C_j) < 3\Delta/4 \\ 4 & 3\Delta/4 < EMD(C_i, C_j) < \Delta \end{cases} \tag{20}$$

The repetition score is defined as the weighted summation of similarity levels according to Eq. 21. Fig. 7 gives two examples of computing repetition scores using the above procedure.

$$R(I) = \sum_{i=1}^4 \text{Count}(SL_i) * \omega_i, \omega = \{1, 1/4, 1/9, 1/16\} \tag{21}$$

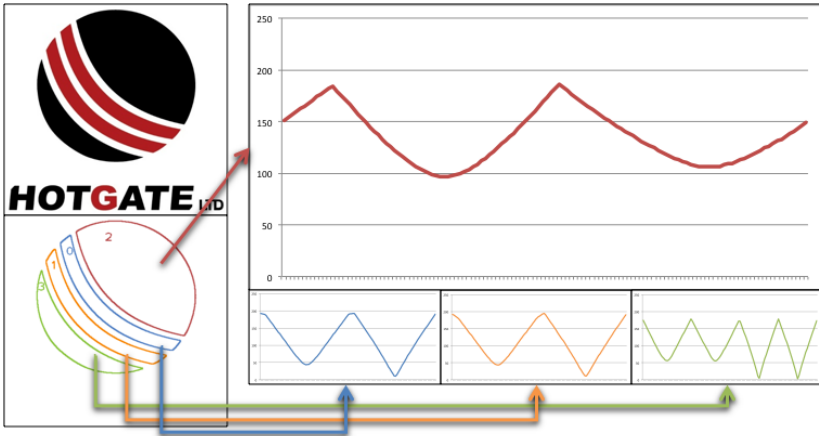


Fig. 6. Use centroid distance to cover 2-D boundary into 1-D representation

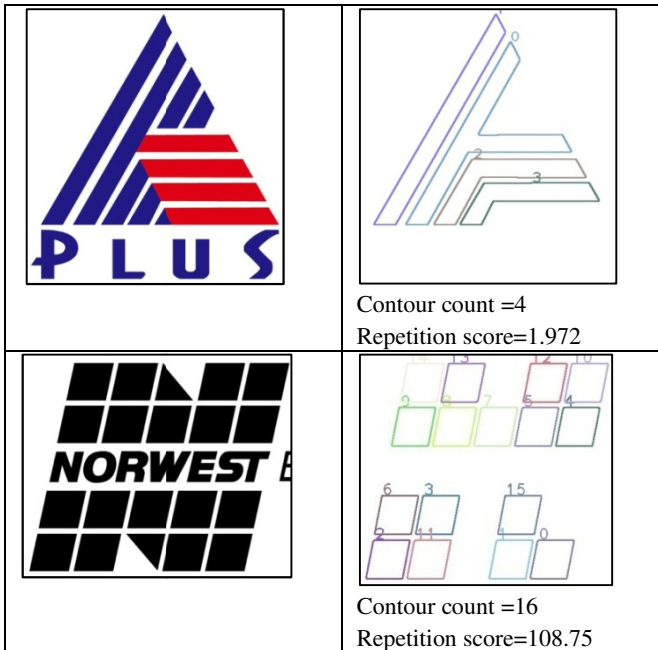


Fig. 7. Repetition scores of two logos

## 4 Experimental Results and Analysis

We have collected 26000 logos from the Internet. These logos are vector graphics, so that no artifact will be generated when resizing is applied. Fig. 8 depicts some logos used in our study. Three visual features defined in Section 3, namely, balance,

complexity and repetition will be computed. The results will be analyzed and discussed respectively. The computation of balance is independent of the size of the logo. Complexity analysis calls for normalization of image width and height. To measure shape repetition, preprocessing is needed to acquire dominant contours in the logo.



Fig. 8. Examples of logo used in this study

#### 4.1 Balance of Logo

There are three different ways to compute the balance score as explained in Section 3.1. Due to limitation of space, we choose to show the distribution of the balance score calculated using saturation and value components in Fig. 9. Generally speaking, most logos are balanced along the vertical, horizontal and diagonal directions. The balance is most prominent for the V-partition, i.e., left and right regions. The distribution of the average balance score takes a Gaussian-like shape, with a peak around 0.2.

#### 4.2 Complexity of Logo

Figs. 10-12 depict the distributions of partition-based complexity, homogeneity-based complexity and area-ratio complexity, respectively. Both partition-based and area-ratio complexity have Gaussian-like characteristics, whose mean value is on the lower side of the axis. Homogeneity-based complexity measure exhibits two peaks, which suggests that the designers either use very simple configuration or moderately complex patterns when creating the logo.

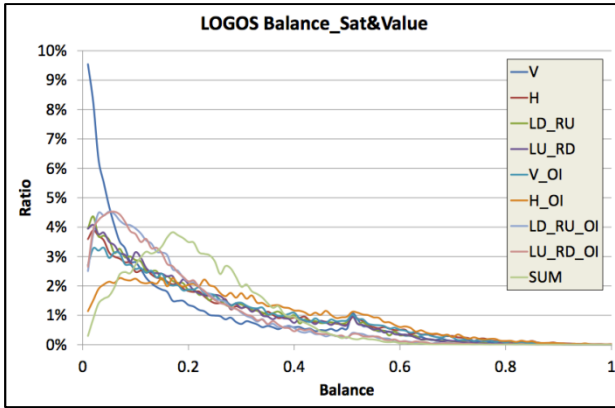


Fig. 9. Distribution of balance scores using saturation and value (8 directions and final score)

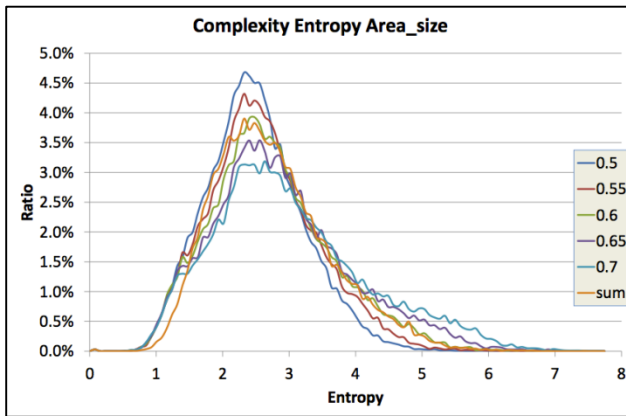


Fig. 10. Distribution of partition-based complexity scores using different thresholds

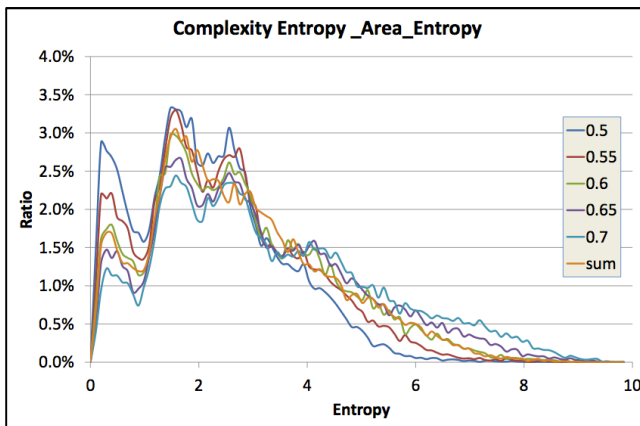


Fig. 11. Distribution of homogeneity-based complexity scores using different thresholds

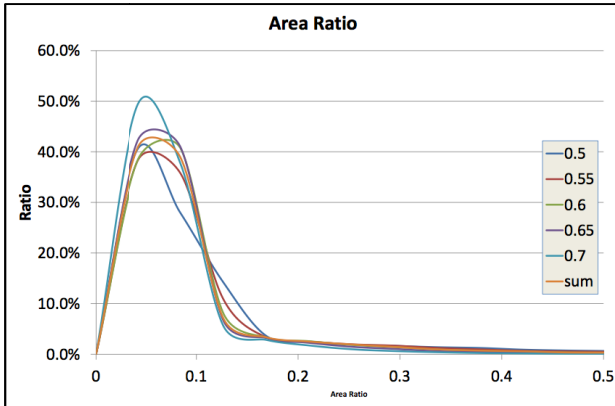


Fig. 12. Distribution of area-ratio complexity scores using different thresholds

### 4.3 Repetition in Logo Design

Fig. 13 illustrates the distribution of the repetition scores computed from the 26000 logos in our dataset. More than 75% of the logos have a repetition score smaller than 0.1.

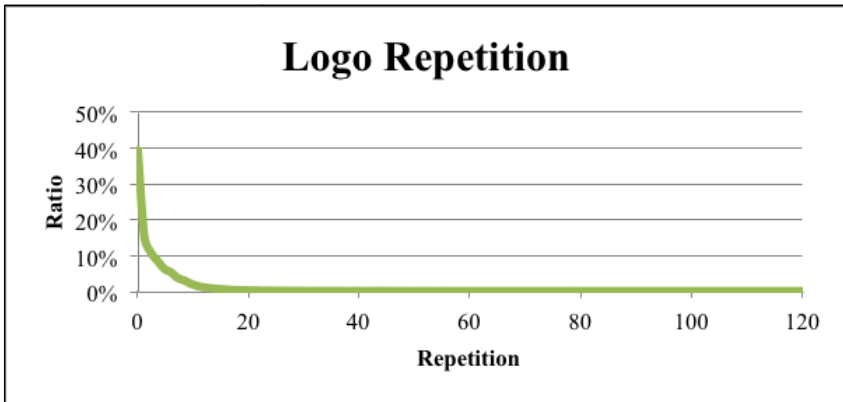


Fig. 13. Distribution of repetition scores

## 5 Conclusions

In this paper, we investigate the design of commercial logos from a quantitative analysis viewpoint. We have identified three key visual elements in logo design, namely, balance, complexity and repetition. We have also delineated procedures to compute these scores from the input image. Our finding from the analysis of 26000 samples indicates that logos are of low complexity, high balance and exhibit minor degree of repetition. Future work includes the search for more relevant mid-level features and methods to compute them from the input for more complicated design and artwork such as paintings and drawings.

## References

1. Viola, P., Jones, M.J.: Robust Real-Time Face Detection. *International Journal of Computer Vision* 57(2), 137–154 (2004)
2. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60(2), 91–110 (2004)
3. Kisanin, B., Pavlovic, V., Huang, T.S. (eds.): *Real-Time Vision for Human-Computer Interaction*. Springer (2005)
4. Hartley, R., Zisserman, A.: *Multiple View Geometry in Computer Vision*, 2nd edn. Cambridge University Press (2003)
5. Stork, D.G.: Computer Vision and Computer Graphics Analysis of Paintings and Drawings: An Introduction to the Literature. In: Jiang, X., Petkov, N. (eds.) CAIP 2009. LNCS, vol. 5702, pp. 9–24. Springer, Heidelberg (2009)
6. Joshi, D., Datta, R., Luong, Q.T., Fedorovskaya, E., Wang, Z., Li, J., Luo, J.: Aesthetics and Emotions in Images: A Computational Perspective. *IEEE Signal Processing Magazine* 28(5), 94–115 (2011)
7. Greenfield: On the Origins of the Term “Computational Aesthetics”. In: Neumann, L., Sbert, M., Gooch, B., Purgathofer, W. (eds.) *Computational Aesthetics*, pp. 9–12 (2005)
8. van der Lans, R., Cote, J.A., Cole, C.A., Leong, S.M., Smidts, A., Henderson, P.W., Bluemelhuber, C., Bottomley, P.A., Doyle, J.R., Fedorikhin, A., Moorthy, J., Ramaseshan, B., Schmitt, B.H.: Cross-National Logo Evaluation Analysis: An Individual-Level Approach. *Marketing Science* 28, 968–985 (2009)
9. Redies, C., Amirshahi, S.A., Koch, M., Denzler, J.: PHOG-derived Aesthetic Measures Applied to Color Photographs of Artworks, Natural Scenes and Objects. In: Fusiello, A., Murino, V., Cucchiara, R. (eds.) *ECCV 2012 Ws/Demos, Part I*. LNCS, vol. 7583, pp. 522–531. Springer, Heidelberg (2012)
10. Wilson, A., Chatterjee, A.: The Assessment of Preference for Balance: Introducing a New Test. *Empirical Studies of the Arts* 23, 165–180 (2005)
11. Birkhoff, G.D.: *Aesthetic Measure*. Cambridge Massachusetts’ University Press (1933)
12. Belongie, S., Malik, J., Puzicha, J.: Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(24), 509–521 (2002)
13. Rigau, J., Feixas, M., Sbert, M.: An Information-Theoretic Framework for Image Complexity. In: *Computational Aesthetics*, pp. 177–184 (2005)
14. Suzuki, S.: Topological Structural Analysis of Digitized Binary Images by Border Following. *Computer Vision, Graphics, and Image Processing* 30, 32–46 (1985)
15. Rubner, Y., Tomasi, C., Guibas, L.J.: The Earth Mover’s Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40(2), 99–121 (2000)

# Controllable and Real-Time Reproducible Perlin Noise

Wei-Chien Cheng, Wen-Chieh Lin, and Yi-Jheng Huang

Department of Computer Science,  
National Chiao Tung University,  
1001 University Road, Hsinchu, Taiwan  
{ayumiqmazaky,jennyhuang914}@gmail.com, wclin@cs.nctu.edu.tw

**Abstract.** Perlin noise is widely used to render natural phenomena or enrich the variety of motion in computer graphics; however, there is less attention on controlling Perlin noise. We present an approach to modify and control the value of Perlin noise function, which closely follows a user-specified pattern while preserving the original statistical properties of the noise. The problem is formulated as a multi-level optimization process, in which the optimization is performed from low frequency to high frequency bands. Our approach can easily achieve global and local control in designing texture patterns and reproduce same patterns without re-optimization.

**Keywords:** Controllable noise, Perlin noise.

## 1 Introduction

Procedure-based approaches synthesize textures by directly producing complex, interesting patterns using a program executed before or during the rendering process [1]. They are particularly useful when the storage of data is limited or the acquisition of data is difficult. They are also very efficient for generating large-scale synthesis. Among existing procedure texture synthesis approaches, noise functions [2–5] have been one of the most widely used techniques due to its flexibility for generating multidimensional patterns and efficiency on the needed storage. For example, Perlin noise [2] has been used to generate sound, image textures, solid textures, height field of terrains, ocean surfaces, and smoke. It has also been applied to computer animation to add the randomness of motion [6] or to generate turbulence vector field for fluid animation [7].

Unlike white noise which is composed of unconstrained random numbers, Perlin [2] constructs noise from different bands, each of which is limited to a range of frequencies. By summing up noise at different bands weighted by amplitude, a variety of patterns can be produced, such as wood, marble, cloud [1]. Perlin noise function is simple and efficient, and provides adequate spectral control.

An ideal noise function should have the following properties: band-limited, stationary, isotropic, reproducible, and aperiodic. These properties have motivated researchers on improving the noise quality by developing a more precise random number table to reduce the bias and band-limited.

Although many approaches have been proposed to describe complex patterns or natural phenomena by composing multiple noise functions, there is less attention on controlling noise functions. It is desirable and useful to generate a stochastic texture with some embedded patterns that meet users' demands in many applications of procedural texture synthesis and terrain synthesis. As modifying noise value arbitrarily would destroy the statistical properties of a noise function, we need an approach that can control a noise function while preserving its statistical properties. In this paper, we propose an approach, targeting at interactive applications, to generate controllable noises that closely match a user's demanded value without losing its statistical properties through an optimization process. Furthermore, we make a controllable noise function reproducible without re-running a lengthy optimization process by sampling the distribution of a random number table that fulfills the user-specified pattern and statistical properties.

The contributions of this paper are: (1) proposing an approach that can generate a user-demanded noise pattern through an optimization process while preserving the statistical properties of the random number table; (2) developing a multi-level optimization algorithm that can effectively reduce the computational time by propagating the optimizing values from lower levels to higher levels; (3) reproducing the same noise patterns without re-running the computationally expensive optimization process.

## 2 Previous Work

Perlin noise function is widely used to generate procedural textures such as cloud, marble, fire, woodgrain [1, 8, 9]. It can also be used to disturb the movement of an object so that animated motions look more natural [6]. Perlin and Neyret [10] further suggested a method to carry out dynamic swirling and flowing textures by rotating the gradient and sampling a noise with an offset to generate pseudo-advection. Perlin noise has also been implemented on GPU shader [11]. Lewis [9] proposed another approach to generate natural patterns based on Wiener interpolation. His approach can generate noise with arbitrary energy spectral, but is more complicated than Perlin noise. Perlin improved the original Perlin noise function by applying a new interpolation approach that gives  $c^2$  continuity and proposing a gradient vector table that avoids the non-uniform distribution of gradients and speed up the computation [3]. Due to the versatile applications of Perlin noise, there are new procedural noise functions proposed recently, e.g., wavelet noise [4] and Gabor noise [5]. These procedural noise approaches focus on solving the aliasing problem.

There have been few approaches dealing with the control of noise functions. Lewis [13] proposed the stochastic subdivision construction, which can be used to control the autocorrelation and spectral properties of the synthesized random functions. Yoon et al. proposed a method to control Perlin noise by modifying the pseudo-random gradient table to satisfy a user's demand [12]. Later, they integrated other statistical tools to measure the stability of the gradient table



during an optimization process to meet the user specified pattern [14]. Their methods can maintain the property of the noise function, including uniformly distributed, non-periodic and band-limited. Both approaches proposed by Yoon et al. [12,14] cannot achieve reproducibility. They need to re-run the optimization procedure, which is a time-consuming, to generate the same noise pattern. On the other hand, our approach can reproduce similar noise patterns by simply re-sampling from a distribution of gradient tables that have been optimized for satisfying the user-control pattern and preserving the statistical properties of a noise function.

### 3 Approach

An overview of our approach is shown in Figure 1, which consists of the preprocessing and the noise reproducing stages. In the preprocessing stage, our system extracts noise values from a user-specified pattern image and uses them as the initial guess of the optimization solver. The optimization process will then try the user-specified noise with the statistical properties. Because the fractal sum of noise is composed of low to high frequency noises, the gradient vectors are shared by low to high frequency noises. We propose a hierarchical optimization process to reduce the computational time by classifying the gradient vectors according to their frequency band. Once we get the optimal gradient vectors (the component of the noise composition), we estimate the gradient vector distribution so that we can re-sample the gradient vectors from the estimated distribution without re-running optimization. We apply a hierarchical clustering algorithm that clusters the optimized solutions to get an initial guess of the number of components in a Gaussian mixture model. We iteratively use the Gaussian mixture model to fit the optimal solution set until we get an appropriate number of components for reproducing the same pattern of noise. In the reproducing stage, we can reproduce a noise with similar patterns without losing the statistical properties in real time.

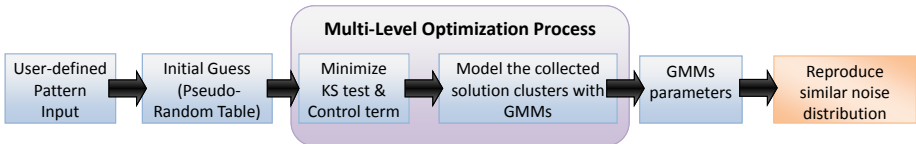


Fig. 1. System Overview

#### 3.1 Perlin Noise Function

Perlin noise function utilizes a hypercubic lattice on which a gradient vector is assigned at each grid point. A gradient vector is a unit-length vector randomly sampled from an  $N$ -dimensional space with the uniform distribution. For a 2D noise,  $N$  equals to two; for a solid noise,  $N$  equals to three and so on. These

gradient vectors are usually pre-sampled off-line and stored in a gradient table  $G$  that is hashing indexed with a permutation table  $P$ . Figure 2 illustrates the definition of Perlin noise in the 2D case. Let  $(x, y)$  be the coordinate of a 2D point  $v$ , and  $v_1, v_2, v_3, v_4$  the four nearest grid points of  $v$ . We denote position vectors  $\vec{v} = (x, y)$ ,  $\vec{v}_1 = (\lfloor x \rfloor, \lfloor y \rfloor)$ ,  $\vec{v}_2 = (\lfloor x \rfloor + 1, \lfloor y \rfloor)$ ,  $\vec{v}_3 = (\lfloor x \rfloor, \lfloor y \rfloor + 1)$ , and  $\vec{v}_4 = (\lfloor x \rfloor + 1, \lfloor y \rfloor + 1)$ , where  $\lfloor \cdot \rfloor$  is the floor function. The noise value at a grid point is defined as the inner product of the fractional vector  $\vec{v} - \vec{v}_i$  and the gradient vector  $\vec{g}_i \in G$ . For a non-grid point, the noise value is obtained by interpolating the noise values at four nearest grid points that enclose the non-grid point,

$$Noise(x, y; G) = \sum_{i=1}^4 W(x - \lfloor x \rfloor)W(y - \lfloor y \rfloor)\vec{g}_i \cdot (\vec{v} - \vec{v}_i), \quad (1)$$

where  $W(t) = 6t^5 - 15t^4 + 10t^3$  is the interpolation function proposed in [3]. For an  $N$ -dimensional noise,

$$Noise(x_1, \dots, x_N; G) = \sum_{i=1}^{2^N} W(x_1 - \lfloor x_1 \rfloor) \cdots W(x_N - \lfloor x_N \rfloor)\vec{g}_i \cdot (\vec{v} - \vec{v}_i), \quad (2)$$

where  $2^N$  is the number of lattice points closest to an  $N$ -dimensional point  $\vec{v} = (x_1, x_2, \dots, x_N)$ . To simulate the self-similarity commonly seen in natural patterns, a fractal sum is used,

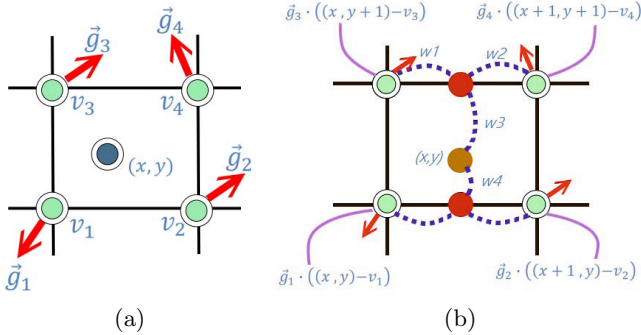
$$F_{sum} = \sum_{l=1}^F \frac{Noise(f_l x, f_l y; G)}{a_l}, \quad (3)$$

where  $F$  is called *octave*.  $f_l$  and  $a_l$  are the frequency and amplitude of band  $l$ , respectively. The fractal sum combines low frequency noise with larger weighting, i.e.,  $f_l < f_{l+1}$  and  $a_l < a_{l+1}$ .  $f_{l+1} = 2f_l$  and  $a_{l+1} = 2a_l$  are usually used.

### 3.2 Goodness of Fit Test

As our goal is to modify the gradient vectors in table  $G$  to match the user-specified noise value while maintaining the uniform distribution of gradient vectors, we need some statistical tools to measure the uniformity of the distribution of the gradient vectors. The goodness of fit of a statistical model is used to describe how well the observed data fit the expected distribution. We briefly introduce two goodness of fit tests in this section.

**Chi-Square (CS) goodness of fit test** [15] is used to determine whether a set of data comes from a specified distribution by measuring the difference between the data and those sampled from the specified distribution. Let  $X = X_1, X_2, X_3, \dots, X_M$  be the observed  $M$  data. To compute the distribution from the observed data, we divide the set  $X$  into  $K$  bins and define  $O_k$  as the set of observed data in the  $k$ th bin. Let  $E$  be the specified distribution and  $E_k$  the



**Fig. 2.** (a) Grid points with assigned gradient vectors. (b) Noise value is interpolated from the surrounding dot product value.

observed distribution for the  $k$ th bin. Chi-square test measuring the similarity between the set  $O$  and set  $E$  is defined as follows:

$$CS(X, E) = \sum_{k=1}^K (|O_k| - |E_k|)^2 / |E_k|, \quad (4)$$

where  $|O_k|$  is the number of observed data in the  $k$ th bin and  $|E_k|$  denotes the number of samples in the  $k$ th bin. For Perlin noise, we need the direction of gradient vectors to be uniformly distributed. Thus,  $|E_k|$  equals to  $M/K$ .  $CS$  is the difference between the observed distribution and specified distribution. The smaller  $CS$  is, the better the distribution of observed data matches the specified distribution.

**Kolmogorov-Smirnov (KS) goodness of fit test** quantifies the distance between the empirical distribution function of the sample data and that of the specified distribution. It is defined as follows:

$$KS(X, E) = \text{Max}|F_X(x) - F_E(x)|, \quad (5)$$

where  $F_X$  is the empirical distribution function, which is a cumulative distribution function (CDF) for  $M$  independently and identically distributed observations  $X_i$ . If the observed distribution matches the specified distribution, the curves of the two cumulative distribution function (CDF) should be similar. Thus, the maximal distance between the two curves must be very small. Because the CDF of the uniform distribution is approximately a straight line, the CDF of the gradient vectors must be yield to a straight line in order to satisfy the uniformity.

Chi-square test requires the expected number in each bin to be sufficiently large. An expected number greater than 5 for each bin is usually needed [16–18]. Chi-square test is adopted in [14]. On the other hand, the KS test does not have the restriction on the expected number for each bin. Furthermore, as described by Mitchell [19] who compared the two tests, KS test is considerably advantageous when the samples are scattered throughout a relatively large discrete

category. We experimented both tests and found that the KS test is indeed a better measurement of the goodness of fit test. As the result, we apply the KS test to measure the uniformity of the distribution of gradient vectors.

### 3.3 Noise Optimization

We formulate the noise control problem as an optimization problem. The goal is to find the optimal gradient vectors such that a texture synthesized from Perlin noise can match a user-specified pattern and preserve the uniformity of the distribution of the gradient vectors. We develop a multi-level optimization scheme to reduce the computational time. The objective function consists of a goodness of fit test term  $E_S$  and a user control term  $E_C$ :

$$E(G) = w_1 E_S(G) + w_2 E_C(G), \quad (6)$$

where  $G$  is the set of gradient vectors.  $w_1$  and  $w_2$  are the weights of each term. We adopt the KS test to measure the uniformity of the distribution of the gradient vectors by setting

$$E_S(G) = KS(G_{ideal}, G) = \text{Max}|F_{ideal}(g) - F_{obs}(g)|, \quad (7)$$

where  $G_{ideal}$  is the ideal sample set constructed by uniformly sampling the gradient vectors in all directions.  $G$  is the set of gradient vectors to be optimized.

The user control term  $E_C$  is defined as

$$E_C(G) = \sum_{i=0}^I (V_i - D_i)^2, \quad (8)$$

where  $I$  is the total number of user-specified noise values;  $V_i = \text{Noise}(x_i, y_i; G)$  and  $D_i$  are the noise value and user-control noise value at point  $i$ . Note that we focus on controlling 2D texture synthesis in this paper.  $D_i$  is extracted from an input image of the control pattern specified by a user. Figure 6 shows some examples of control pattern, in which black color and red color denote the noise value of 0 and 1, respectively. We set  $w_1$  at 0.995 and  $w_2$  at 0.005 in all of our experiments and apply simulated annealing algorithm to solve the optimization problem.

The fractal sum used to express the natural-looking phenomena is composed of multiple noises at different frequency bands. Lower frequency bands control the global shape of the fractal noise while higher frequency bands affect the details of the noise. To speed up the optimization process, we divide the optimization problem into multiple stages. Specifically, we divide the gradient vectors in the gradient table  $G$  according to the frequency bands they affect and then compute optimal gradient vectors hierarchically from the lower frequency bands to higher frequency bands. The optimal gradient vectors obtained at a lower frequency band serve as the initial guess for the optimization at a higher frequency band. Assume that the frequency in the higher level is two times of that in the lower

level, the initial guess of gradient vectors at level  $l + 1$  is propagated from the optimal ones at level  $l$  using

$$G[P[P[[_{l+1} \cdot x]] + [_{l+1} \cdot y]]] = G[P[P[[_{2f_l} \cdot x]] + [_{2f_l} \cdot y]]], \quad (9)$$

where  $G$  is hashing indexed with a permutation table  $P$ .  $E_S$  and  $E_C$  terms in the objective function in each level are modified as follows:

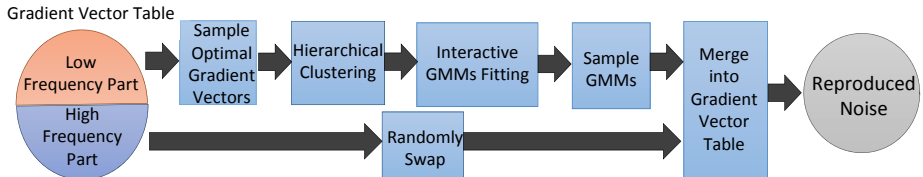
$$E_S^l(G^l) = KS(G_{ideal}, G^l), \quad (10)$$

$$E_C^l(G^l) = \sum_{i=0}^{I^l} (V_i^l - D_i^l)^2, \quad (11)$$

where  $G^l$  are the gradient vectors in the  $l$ th level and  $V_i^l$  is the  $i$ th noise value in the  $l$ th level.

### 3.4 Noise Reproduction

We propose a simple approach to make the noise reproducible in real time. We precompute multiple sets of gradient vectors by running the optimization process several times with different initial conditions and adopt a Gaussian Mixture Model (GMM) to represent the distribution of these optimal gradient vectors. By re-sampling from the constructed GMM, we can quickly reproduce a similar (but not the same) texture embedded with the user-specified control pattern. Figure 3 illustrates our framework for noise reproduction.



**Fig. 3.** The proposed framework for noise reproduction

At the first step, we separate the gradient vectors into the high frequency and low frequency parts. For the high frequency part, we simply swap some gradient vectors in the gradient table. In this way, the local details are disturbed while the distribution of the gradient vectors at the high frequency part does not change. For low frequency parts, the distribution of the gradient vectors would affect the global appearance significantly since the amplitudes for lower frequency bands are larger. To reduce the computational time, we focus on creating the varieties of the gradient vectors corresponding to low frequency noises.

We execute the multi-level optimization process several times with random initial conditions. At each execution, we collect  $N$  sets of gradient vectors. Let  $G_n$

be the set of gradient vectors collected at the  $n$ th iteration and  $E(G_n)$  the value of objective function when  $G_n$  is the set of gradient vectors. Moreover, let  $E^*$  be the minimum of the objective function and  $G^*$  is the corresponding optimal set of gradient vectors. We keep  $G_n$ 's whose objective function values are within a bound of the minimum during the optimization process, i.e.,  $E(G_n) - E^* < \varepsilon$ . After executing  $E$  times of the optimization, a solution pool containing  $N \cdot E$  sets of gradient vectors is build. We then apply the hierarchical clustering algorithm to divide these gradient vector sets into  $K$  disjoint subtrees. The advantage of hierarchical clustering is that the number of initial clusters is automatically determined. The algorithm merges clusters at the lower level to create larger clusters at the higher level until the Hausdorff distance between any two clusters exceeds a cutoff value set by the user. The algorithm then stops and outputs  $K$  disjoint subtrees (Figure 4(a)).

After initial clustering,  $K$  disjoint solution clusters are obtained. We then use Gaussian Mixture Models (GMMs) to represent the distribution of these solution clusters. Gaussian mixture models consist of multivariate normal density components. The parameters of the GMMs that best fit the distribution of a data set can be obtained using the expectation maximization (EM) algorithm. We use  $K$  as the initial guess of the number of components in GMMs fitting. In addition, Akaike information criterion [20] ( $AIC$ ) is used as a metric to determine the error between the fitted model and the input data.  $AIC$  takes into account both the goodness of fit of an estimated statistical model and the number of parameters in the statistical model. Lower  $AIC$  indicates that the parameters in GMMs better describes the input data. We run the GMM fitting iteratively and increase the number of components gradually until appropriate number of components is found. Figure 4(b) shows iterative GMM fitting. Once we get the parameters of GMMs, we can easily re-sample the gradient vectors of low frequency noise. By randomly swapping the gradient vectors of the high frequency noise, we can make some disturbance on the noise appearance while the uniformity is still preserved since the swap operation will not affect the distribution of gradient vectors. Combining the gradient vectors of high frequency and low frequency bands, we can easily reproduce noises.

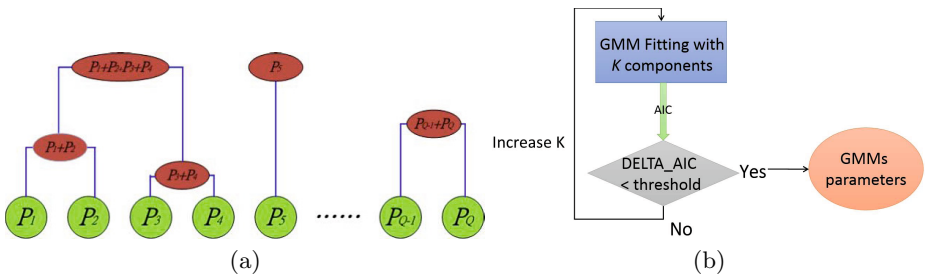
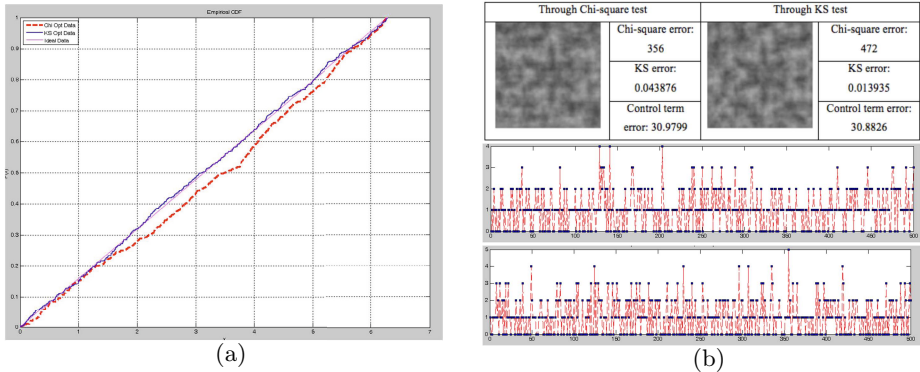


Fig. 4. (a) Hierarchical Clustering. (b) Iteratively GMM fitting.

## 4 Experimental Results

### 4.1 Statistical Analysis

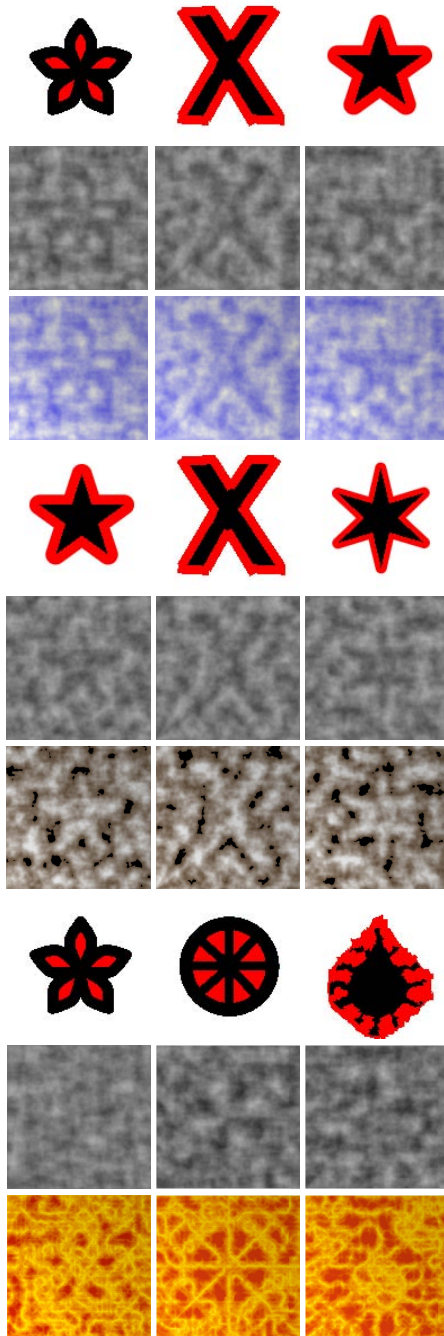
We first compare the results of using the KS test and chi-square test by plotting their probability distribution function and cumulative distribution function. Figure 5 shows an example. The pink line represents the CDF of the ideal distribution, the red dash line and blue line represents the CDF of the gradient vectors optimized with chi-square test and KS test, respectively. We also plot the histogram of gradient vectors using both tests. We separate the 512 gradient vectors into 500 bins and the expected number in each bin is set to 1. The control term errors in both tests are very close; however, one can observe that the CDF curve of KS test is very close to the ideal distribution while the CDF curve of chi-Square test obviously deviates from the ideal distribution. This example demonstrates that KS test is a better metric for measuring the uniformity.



**Fig. 5.** Comparison of chi-square test and KS test. (a) CDF of the gradient vectors. (b) Top: Synthesized result and errors; Middle: Histogram of the gradient vectors using chi-squared test; Bottom: Histogram of the gradient vectors using KS test.

### 4.2 Control of Texture and Terrain Synthesis

We demonstrate our approach on procedural texture synthesis, including cloud, erosion and fire textures. Figure 6 shows our results. The top, middle, and bottom row for each example shows the control pattern given by a user, the controlled noise obtained by our multi-level optimization process and the corresponding output textures with user control pattern. In the cloud texture, noise values are set as the alpha values to blend blue and white colors. The erosion texture is generated by taking the absolute value of the controlled noise, so  $E_C$  is modified as  $E_C = \sum_{i=0}^I (|V_i| - D_i)^2$ . Similarly, the fire texture uses the summation of the absolute noise value in all octaves, so  $E_C$  is defined as:



**Fig. 6.** Three textures generated with different user control patterns. In each example, the top, middle, and bottom rows correspond to user-specified control patterns, controlled noises, and output patterns, respectively.



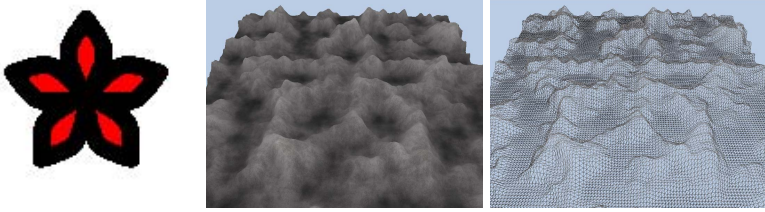


Fig. 7. Terrain generated by user-specified flower pattern in (a)

$$E_C = \sum_{i=0}^I \left( \sum_{l=1}^F \frac{|Noise(f_l x, f_l y)|}{a_l} \right)^2.$$

Our approach can flexibly control different types of textures by modifying the control term  $E_C$ . Besides, our noise control approach can be applied to terrain generation by taking the noise as a height map (Figure 7).

## 5 Conclusion and Future Work

We propose a method to generate noise whose global structure matches user-specified patterns while its local structure still preserves the original statistical properties. Our system achieves these goals by formulating the problem as a high-dimensional nonlinear optimization problem and solving it in a hierarchical optimization framework. Furthermore, we can reproduce similar controlled noise without re-running the time-consuming optimization process. Our approach can be applied to other applications such as computer animation or computer games. Currently, our system only implements the noise control in two-dimensional noise. We would like to apply our approach to control three- or multi-dimensional noises. Our framework is flexible. We may apply it to control other type of noise functions in the future. Finally, in the fractal sum representation, we only take the gradient vectors as the control mechanism. We may also optimize the amplitude of different frequency bands to achieve better user control.

**Acknowledgments** We thank the anonymous reviewers for their insightful comments. This work was supported in part by the Ministry of Science and Technology of Taiwan (NSC-101-2628-E-009-021-MY3 and NSC-102-2221-E-009-082-MY3) and the UST-UCSD International Center of Excellence in Advanced Bioengineering sponsored by the Ministry of Science and Technology I-RiCE Program under grant number: NSC-102-2911-I-009-101.

## References

1. Ebert, D.S., Musgrave, F.K., Peachey, D., Perlin, K., Worley, S.: Texturing and Modeling: A Procedural Approach. Morgan Kaufmann, San Francisco (2002)

2. Perlin, K.: An Image Synthesizer. In: 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 287–296. ACM, New York (1985)
3. Perlin, K.: Improving noise. In: 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 681–682. ACM, New York (2002)
4. Cook, R.L., De Rose, T.: Wavelet noise. *ACM Transaction on Graphics* 24, 803–811 (2005)
5. Lagae, A., Lefebvre, S., Drettakis, G., Dutré: Philip.:Procedural noise using sparse Gabor convolution. *ACM Transaction on Graphics* 28(54) (2009)
6. Perlin, K.: Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics* 1, 5–15 (1995)
7. Bridson, R., Houriham, J., Nordenstam, M.: Curl-noise for procedural fluid flow. *ACM Transaction on Graphics* 26(46) (2007)
8. Apodaca, G.L., Barzel, R.: *Advanced RenderMan: creating CGI for motion picture*. Morgan Kaufmann, San Francisco (2000)
9. Lewis, J.P.: Algorithms for solid noise synthesis. In: 16th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 263–270. ACM, New York (1989)
10. Perlin, K., Neyret, F.: Flow Noise. *ACM SIGGRAPH Technical Sketches and Applications* 187 (2001)
11. Hart, J.C.: Perlin noise pixel shaders. In: the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, pp. 87–94. ACM, New York (2001)
12. Yoon, J., Lee, I., Choi, J.: Editing noise: Research Articles. *Computer Animation and Virtual Worlds* 15, 277–287 (2004)
13. Lewis, J.P.: Generalized stochastic subdivision. *ACM Transaction on Graphics* 6, 167–190 (1987)
14. Yoon, J., Lee, I.: Stable and controllable noise. *Graphical Models* 70, 105–115 (2008)
15. Knuth, D.E.: *The art of computer programming*. Addison-Wesley, Redwood City (1998)
16. Lewis, D., Burke, C.J.: The use and misuse of the chi-square test. *Psychological Bulletin* 46, 433–489 (1949)
17. Lieberman, B.: *Contemporary problems in statistics; A book of readings for the behavioral sciences*. Oxford University Press, New York (1971)
18. Siegel, S.: *Nonparametric statistics for the behavioral sciences*. McGraw-Hill Humanities, New York (1956)
19. Mitchell, B.: A Comparison of Chi-Square and Kolmogorov-Smirnov Tests. *The Royal Geographical Society* 3, 237–241 (1971)
20. Akaike, H.: A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19, 716–723 (1974)

# Emotional Line: Showing Emotions through the Sharpness, Height, Width and Speed of a Series of Concatenated Digital Shifting Curves

Jesús Ibáñez<sup>1</sup> and Carlos Delgado-Mata<sup>2</sup>

<sup>1</sup> Madeira-ITI, University of Madeira, Funchal, Madeira, Portugal  
jesus.ibanez@m-iti.org

<sup>2</sup> Universidad Panamericana, Aguascalientes, Mexico  
cdelgado@up.edu.mx

**Abstract.** This paper proposes and explores an abstract approach to express emotions. Emotions are represented in terms of valence and arousal dimensions and they are visually expressed through the shape and speed of a series of concatenated digital shifting curves that together compose a curved line that resembles a non-periodic wave. In particular, the valence value is expressed through the sharpness of the curves (the more negative the valence, the sharper the curves), while the arousal value is expressed through their height, width and speed (the greater the arousal, the higher, thinner and quicker the curves). Furthermore, the paper describes a user experiment which investigated whether the valence and arousal expressed by our model are appropriately perceived by the users or not. The results suggest that combinations of sharpness, height, width and speed are perceived correctly as particular emotions; that sharpness is perceived as valence; that height, width and speed are jointly perceived as arousal; and that the perceptions of both valence and arousal are independent of each other.

## 1 Motivation

Nowadays, the most usual way to show emotions in digital environments is through the use of virtual characters. However, the use of human-like characters arises the problem of wrong expectations about the systems behaviour [6]. The human-like behaviour of the character in some aspects may lead the user to believe that the agent resembles human beings in other cognitive aspects as well. Thus, while the expression of emotions through virtual characters is useful in some contexts, there are other situations where virtual characters are neither necessary nor adequate. In these cases, more subtle, non-human-like approaches could be employed, in order to reduce the user's expectations.

This kind of non anthropomorphic models for showing emotions is particularly useful, for instance, for *social awareness systems*, which can be defined as systems whose purpose is to help connected individuals or groups to maintain a peripheral awareness of the activities and the situation of each other [21]. Interesting awareness systems have been proposed, in the last years, for both

workplaces [16] and social/family life [7]. In these systems, the expression of the emotional state is an open problem which has been explored by some works such as [1][15]. Another application area for this kind of visualization of emotions is digital art. Actually, recent works have explored the expression of emotions through several abstract digital features in digital art installations [5].

In this sense, we present in this paper an approach to show emotions that intends to minimize the user's expectations. Emotions are represented in terms of valence and arousal dimensions and they are visually expressed through the shape and speed of a series of concatenated digital shifting curves that together compose a curved line that resembles a non-periodic wave.

The idea of using a line to express emotions in the present work was initially inspired by the work of the comic theorist McCloud [22], who noted that a simple line has an enormous expressive potential. He subjectively argued, by showing visual examples, that by direction alone, a line may go from passive and timeless to proud and strong or to dynamic and changing. By its shape, it can be unwelcoming and severe, warm and gentle, or rational and conservative. By its stroke it may seem savage and deadly, weak and unstable, or honest and direct. While we do not explore the varied palette of possibilities McCloud suggests (more oriented towards a categorical model of emotions), we agree on the expressive potential of a simple line. The motivations for the particular line features selected for expressing valence and arousal in our model are detailed in section 2.

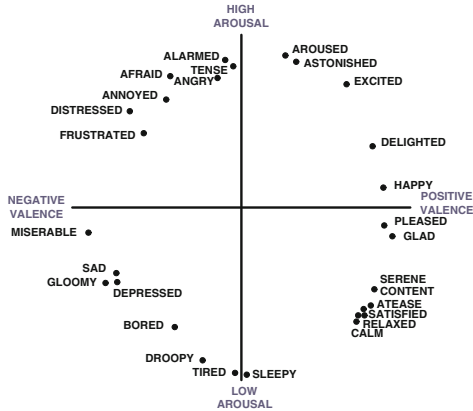
This paper also describes and discusses an experiment with users which investigated whether the arousal and valence expressed by our model are appropriately perceived by the users or not. The structure of the paper is as follows. First we review related work. Next we detail the proposed model. Then we describe the experiments and discuss its results. Finally we show the conclusions and point out future work.

## 2 Related Work

Two different approaches for modelling emotions are generally accepted: the categorical approach and the dimensional approach. We are interested in researching the expression of emotions through simple visual features that can be modelled as values in a continuous scale. This very well fits the *dimensional approach* which suggests the existence of major dimensions that are sufficient to describe and distinguish between different emotions. In contrast, the *categorical approach* [24] would not be directly applicable in our case, as it assumes the existence of a set of universal, so-called *basic emotions* that can clearly be distinguished from one another and form the basis for all other emotions we might experience.

The model proposed by Wundt [29] was probably the first dimensional model. Several alternatives have been proposed since then. Russell's circumplex model [26] is among the most accepted. According to his theory, emotions can be described in terms of two bipolar, continuous and orthogonal dimensions: the dimension of *valence* with the poles positive (or pleasure) and negative (or displeasure) and the

dimension of *arousal* with the poles calm and excited. Sometimes, an additional dimension of *dominance* with the poles strong and weak is suggested. Our model works with the bi-dimensional approach of emotions, where emotions are represented in terms of valence and arousal. In this sense, various authors have described several different emotions using these dimensions by locating the emotion words in the bi-dimensional space. For instance, Fig. 1 shows some emotion words according to Russell’s circumplex model of affect [26].



**Fig. 1.** Russell’s circumplex model of affect

In the following, we contextualize the motivations for the particular line features selected for expressing valence and arousal in our model. We start with valence, which is expressed through the line sharpness. The perception of curved versus angular lines and shapes has been studied in literature in terms of preference and several emotional aspects. For instance, Lundholm [20] reported an experiment where participants were asked to draw lines that characterized feelings. The lines were then categorized in terms of angularity. Angular lines were associated with feelings such as agitating, hard and furious. Curved lines were associated with feelings such as gentle, quiet and lazy. Similar results were found by Poffenberger and Barrows [25], who studied the experience of people viewing curved and angular lines. In an analogous study, Hevner [9] concluded that curves are found to be serene and graceful, while angles are robust and vigorous.

Monö [23] showed that circles, spirals and shapes with smooth curves were more pleasant than shapes with hard angles. Silvia and Barona [27] studied preference in terms of pleasantness as well. Results showed that people preferred round circles more than angular hexagons and curved polygons more than angular polygons. Leder and Carbon [19], researching on car interior design, found that curved designs are preferred to straight designs, and that curvature elicits increased positive emotions. Studies of human facial expression and interpretation suggest that sharp primitive elements transmit threat, while round ones

convey warmth [2]. Along the same line, Zebrowitz [30] found that rounder faces are more liked and generally perceived as more attractive than more angular faces. Halper et al. [8] found a relationship between line style and perception of safety. Objects rendered using jagged lines were perceived as more dangerous than objects rendered using smooth lines. Similarly, Bar and Neta [3] suggested that people prefer curved objects because angularity conveys a sense of threat [4].

Literature review suggests that sharpness of curvature could affect both arousal and valence perception. In terms of valence, curved objects seem to be perceived as more positive than angular objects. In terms of arousal, curved objects seem to be perceived as less activated than angular objects. Thus, in principle, the sharpness of curvature could be potentially used for expressing both valence and arousal. We think that the capability of sharpness for expressing valence could be emphasized (at the expense of its capability for expressing arousal) by combining it with other modalities that clearly express arousal (those other modalities could mask the arousal denoted by sharpness). The work presented in this paper follows that strategy. We combine sharpness of curvature (used here to express valence) with other modalities that according to our hypothesis clearly express arousal.

Regarding arousal, in previous works [14][11][10][12][13], we found that movement can be used to express arousal. In those works, we employed a flocking algorithm to control the movement of certain digital elements in a bi-dimensional space. Particularly useful was the effect of velocity. It seemed that the greater the velocity of the visual elements, the greater the arousal value the user perceived. That is the reason why, in the present work, we use the speed of the curves' movement as a parameter to show arousal. In the present model, the greater the arousal of the emotion to be shown, the quicker the movement of the curves. The lower the arousal, the slower the movement of the curves. Note that we do not use here any flocking algorithm, as the curves will only move along one dimension (from left to right).

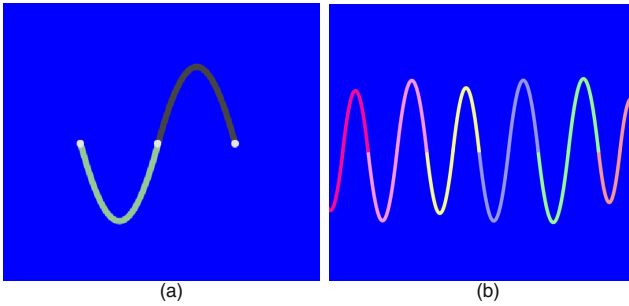
Apart from speed, we use both the height and width of the curves to express arousal. That decision is inspired by the expressiveness of human body postures. Wallbott [28] studied the bodily expression of emotion trying to elucidate whether body movements and body postures are indicative of specific emotions. He found that some emotion-specific characteristics seem to exist, but that differences between emotions can be partly explained by the dimension of activation. As Kok and Broekens [17] stated, a slumped posture denotes low arousal and an upright posture denotes high arousal.

Our intention is that the line abstractly mimics the expressiveness of the body postures. Thus, the greater the arousal, the higher and thinner the curves (mimicking an upright posture). The lower the arousal, the shorter and wider the curves (mimicking a slumped posture).

### 3 Our Model

In our model, the emotion is visually expressed through a curved line that scrolls horizontally from left to right. The line is built by concatenating variations of

a minimum block of construction which consists of a pair of Bézier curves. As shown in Fig. 2a, both curves of a block are the same height and width, although one of the curves is concave and the other one is convex. Note that both curves are horizontally concatenated (the last point of one of the curves is also the first point of the other curve). For clarity, both curves are shown in a different color in Fig. 2a (although in the actual model, both curves are shown in the same color).



**Fig. 2.** (a) The minimum block of construction consists of a pair of Bézier curves. (b) The resulting overall curved line is composed of pairs of curves that are similar but not identical to one another.

Each block is processed before it is added to the overall line. Certain characteristics of the curve's shape (i.e. its sharpness, width and height) depend on valence and arousal. Moreover, some randomness is introduced when determining the horizontal and vertical dimensions of each new couple of curves. Sections 3.1 and 3.2 provide more detail about the steps to construct a particular pair of curves.

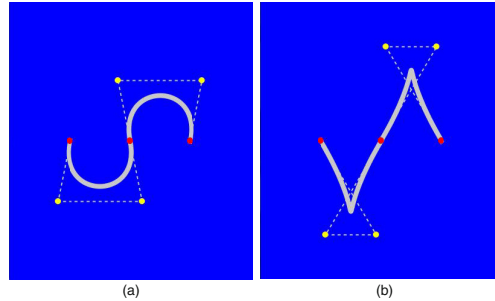
Thus, the resulting overall curved line, as shown in Fig. 2b, is composed of pairs of curves that are similar but not identical to one another. Note that the resulting line resembles a non-periodic wave. Again, for clarity, each couple of curves is shown in a different color in Fig. 2b. However, in the actual model, all the pairs are in the same color.

The curves (and therefore the line they constitute) are moved from left to right at each time step. The speed of that movement (in pixels per step) depends on the current arousal value (as described in section 3.2). When the last added pair of curves exceeds a minimum threshold (located at some distance from the left edge of the visible frame), a new pair of curves is added. When a pair of curves exceeds a maximum threshold (located at some distance from the right edge of the visible frame), it is removed.

### 3.1 Valence

The valence value, in our approach, is expressed through the sharpness of the curves added to the global line. In this regard, there are two different extreme

models of the basic block of construction: a pair of rounded Bézier curves (see Fig. 3a) and a pair of very sharp Bézier curves (see Fig. 3b). The first one is used for expressing the most positive valence. The second one is employed for the most negative valence. In Fig. 3, the end points of the Bézier curves are marked as red filled points, while the control points are marked as yellow filled points. Note that the end points of the rounded curves are located at the same coordinates as the corresponding end points of the sharp curves. Both models differ from each other only in the coordinates of the control points.



**Fig. 3.** The extreme models of the basic block of construction: (a) rounded curves corresponding to the most positive valence (b) sharp curves corresponding to the most negative valence

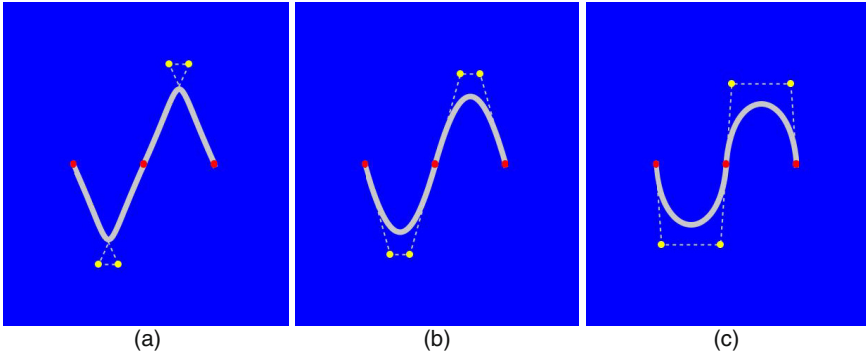
For values of valence between both extremes (the most positive valence and the most negative valence), a new pair of curves is generated as an interpolation of the two basic models (the rounded model and the sharp model). The end points of the new curves are identical to the end points of the original models. However, each control point of the new pair of curves is calculated as the linear interpolation of the corresponding control points of the extreme models. As a result, the more negative the valence value is, the sharper the new curves are. Moreover, the more positive the valence value is, the rounder the new curves are. Figure 4 shows three pairs of curves generated for intermediate valence values; particularly 2.5, 5.0 and 7.5, where the valence value is within the 0-10 range (0 for the most negative valence, 10 for the most positive valence).

The valence value is not the only factor that influences the shape of the new curves. As we will show in the next section, the new curves will still be modified, depending on the arousal value, before they are added to the global line.

### 3.2 Arousal

In our model, the arousal value is expressed through the height, width and speed of the curves. We first describe how arousal affects both the height and width of the curves. Specifically, the greater the arousal, the higher and thinner the curves. The lower the arousal, the shorter and wider the curves. This influence





**Fig. 4.** Three pairs of curves generated for intermediate valence values, particularly (a) 2.5, (b) 5.0 and (c) 7.5, where the valence value is within the 0-10 range (0 for the most negative valence, 10 for the most positive valence)

of the arousal on the curve's shape is achieved by simply scaling the curve height and width based on the arousal value. Moreover, some randomness is introduced when determining the horizontal and vertical dimensions of each new couple of curves, so that not all the curves (for a particular emotional state) are exactly equal to one another. Randomness is added in order to obtain a more attractive and dynamic line.

Thus, the final height and width of a new pair of curves depend on both the arousal value and a random factor which introduces small variations. Next we show the particular calculations we apply in the current model. Let *curveHeight* and *curveWidth* be, respectively, the height and width of the curves that were obtained as a result of the interpolation process described in the previous section. Then, their values are recalculated as follows:

$$randomFactor = random(minRF, maxRF)$$

$$curveHeight = curveHeight \cdot \frac{arousalValue}{CH} \cdot randomFactor$$

$$curveWidth = curveWidth \cdot \frac{CW}{arousalValue} \cdot randomFactor$$

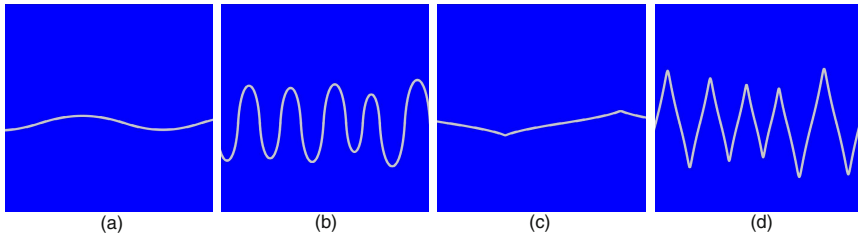
where  $random(x, y)$  is a function which generates a real number between  $x$  and  $y$ ; *arousalValue* is the current arousal value within the (0-10] range (10 for the highest arousal); and  $CH$  and  $CW$  are constants of the scaling factors of height and width respectively. In our current implementation, the value of both  $CH$  and  $CW$  is 5. Moreover, the values of  $minRF$  and  $maxRF$  are 0.8 and 1.2 respectively. That is, the random variation of the curve dimensions is between 80% and 120%.

Regarding the speed of movement, it is measured in number of pixels per time step. At each step, the curves are shifted (from left to right) a number of pixels calculated as:

$$speed = arousalValue \cdot speedFactor$$

In our current implementation, the value of *speedFactor* is 0.5. That is, at each step, the curves are shifted a number of pixels equal to a half the value of the arousal. Therefore, the quickest speed is 5 pixels per step, as the maximum arousal value is 10 (for the highest arousal).

In summary, valence influences the sharpness of the curves (the more negative the valence, the sharper the curves) and arousal influences their height, width and speed (the greater the arousal, the higher, thinner and quicker the curves). Figure 5 illustrates those influences by showing four different combinations of valence and arousal. Note that the final line is in white over a plain blue background.



**Fig. 5.** Four different combinations of valence and arousal: (a) positive valence and low arousal, (b) positive valence and high arousal, (c) negative valence and low arousal and (d) negative valence and high arousal

## 4 User Study

This section deploys an experiment which investigated whether the arousal and valence expressed by our model are appropriately perceived by the users or not.

### 4.1 Aims and Hypotheses

The study has two aims. The first aim is to investigate if particular combinations of valence and arousal expressed by our system are correctly perceived as particular emotions. Our hypothesis is: particular combinations of valence (expressed via sharpness in our system) and arousal (expressed via height, width and speed) will be correctly perceived as particular emotions. The second aim of the study is to investigate if both valence and arousal are correctly perceived when expressed by our system. Our hypotheses are: valence expressed via sharpness in our

system will be correctly perceived; arousal expressed via height, width and speed in our system will be correctly perceived; both the perception of valence and the perception of arousal (expressed by our system) will be independent of each other.

## 4.2 Method

Thirty two participants between the age of 24 and 44 (mean age = 34.32,  $SD = 5.63$ ) took part in the study. Twenty two were men and ten women. Thirty of them had a degree. Participants were asked to rate their experience with computers. Self-reported experience with computers ranged from moderate to very experienced, with a majority rating their experience as reasonable or better. Participants were not paid for their participation in the study.

For the experiment, we developed an application which implements the designed model. The application provides an administrator interface which allows the administrator to vary, in real time, the values of both valence and arousal. The main user interface displays the curved line in white color over a blue background in a 700 x 700 pixel frame. The line thickness is 8 pixels. The developed application was used to record the videos that were employed in the experiment.

The experiment was conducted in a quiet room free of distraction. Participants, seated in front of the computer, were shown a sequence of video that is 5 minutes long. The sequence comprised four windows, each containing a view of the curved line shifting in it from left to right. The order of the windows on the screen was randomised and distributed. Each window corresponded to 1 of the 4 conditions listed below.

- A. Negative valence and a low level of arousal.
- B. Positive valence and a low level of arousal.
- C. Negative valence and a high level of arousal.
- D. Positive valence and a high level of arousal.

Note that conditions A, B, C, and D correspond respectively to the lower left, lower right, upper left, and upper right quadrants of the bi-dimensional valence-arousal space shown in Fig. 1. The participants were allowed to spend as much time as they wanted watching the sequence of video. We annotated the time it took each user to watch the video. This time did not exceed the video duration (5 minutes) in any case.

Regarding the first aim of the study, participants were presented with four tags corresponding to various emotional states. In particular they were presented with these tags:

- T1. Angry, enraged.
- T2. Excited, joyful.
- T3. Bored, sad.
- T4. Calm, satisfied.

Note that tags T1, T2, T3, and T4 correspond respectively to the upper left, upper right, lower left, and lower right quadrants of the bi-dimensional valence-arousal space shown in Fig. 1. Participants were asked to assign each tag to the condition (concrete window in the video) where they thought that emotion was being expressed.

Regarding the second aim, participants were asked to rate each of the conditions according to valence and arousal scales. For this task we employed the paper and pencil version of the Self-Assessment Manikin (SAM), an affective rating system devised by Lang [18]. In this system, ratings of valence are indicated by graphic representations of facial expressions ranging from a broad smile (most positive) to a severe frown (most negative). For arousal, the manikin varies from a state of high to low agitation. In particular, we employed a 9 point Likert scale. In the valence dimension, 1 represents the extreme of pleasantness and 9 the extreme of unpleasantness. For the arousal dimension, 1 represents a high activation and 9 represents a low activation.

Finally, participants were asked to rate each of the four tags of emotional states (T1, T2, T3 and T4) according to the valence and arousal scales. For this task we used the same version of the SAM employed in the previous one. Note that, in this task, the sequence of video was not taken into account. In fact, this task was included in the study as a control mechanism, in order to detect anomalies. For example, participants who rate sad as more positive than joyful would be discarded from the study.

### 4.3 Statistical Analysis

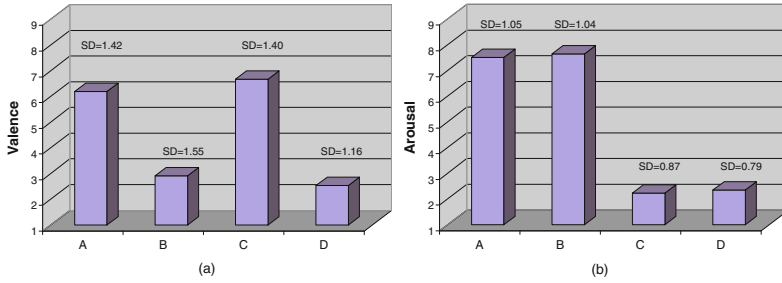
Regarding the first aim of the study, the frequencies of tags for each condition were analyzed by using the chi-square goodness of fit test. As to the second aim, for each studied aspect (valence and arousal) ordinal Likert scores were compared across conditions using Friedman's test for dependent samples. Where significant differences were observed, a Wilcoxon signed rank test was used to examine individual differences. All tests were 2-tailed. Friedman and Wilcoxon tests were preferred to ANOVA due to the nonparametric nature of the gathered data (the variables were not normally distributed).

### 4.4 Results

Regarding the first aim of the study, for each emotion 32 out of 32 participants perceived the intended emotion in the model. By using the chi-square goodness of fit test, a significant difference among the frequencies of tags was found in each of the four conditions ( $\chi^2(3, N = 32) = 96.0, p < 0.001$ ).

As to the second aim, Fig. 6a summarizes the subjective ratings for valence. Each column displays the mean value of the corresponding condition and the standard deviation is shown above it. Note that in the valence dimension, 1 represents the extreme of pleasantness and 9 the extreme of unpleasantness.

Comparison shows that participants found that condition A had a more negative valence than conditions B ( $z = 4.671, p < 0.001$ ) and D ( $z = 4.872,$



**Fig. 6.** Mean values and standard deviations for valence (a) and arousal (b)

$p < 0.001$ ). Comparison also shows that participants found that condition C had a more negative valence than conditions B ( $z = 4.897$ ,  $p < 0.001$ ) and D ( $z = 4.899$ ,  $p < 0.001$ ).

On the other hand, Fig. 6b summarizes the subjective ratings for arousal. Each column displays the mean value of the corresponding condition and the standard deviation is shown above it. Note that for the arousal dimension, 1 represents a high activation and 9 represents a low activation.

Comparison shows that participants found that condition A had a lower level of arousal than conditions C ( $z = 4.988$ ,  $p < 0.001$ ) and D ( $z = 4.978$ ,  $p < 0.001$ ). Comparison also shows that participants found that condition B had a lower level of arousal than conditions C ( $z = 4.971$ ,  $p < 0.001$ ) and D ( $z = 4.993$ ,  $p < 0.001$ ).

## 4.5 Discussion

The results show that participants perceived a more positive valence in any of conditions which had a positive valence than in any of the conditions which had a negative valence (whatever its value for arousal was). However, they did not perceive differences, in terms of valence, between the conditions which actually had the same value for valence. Thus, in principle these results seem to support the appropriateness of our approach to express valence through the sharpness of the curves.

The results also show that participants perceived a higher level of arousal in any of the conditions which had a high level of arousal than in any of the conditions which had a low level of arousal (whatever its value for valence was). However, they did not perceive differences, in terms of arousal, between the conditions which actually had the same value for arousal. Thus, these results seem to support the appropriateness of our approach to express arousal through the height, width and speed of the curves.

## 5 Conclusions

In this paper, we have proposed an abstract approach to visually express emotions (represented in terms of valence and arousal dimensions) through a curved

line that scrolls horizontally from left to right. The line is built by gradually concatenating Bézier curves. The valence value is expressed through the sharpness of the curves, while the arousal value is expressed through their height, width and speed. The proposed approach, which intends to minimize the user's expectations, can be used in two ways. On the one hand, it can be employed to express emotions on its own (for instance, in social awareness systems and artistic installations). On the other hand, it can be used in the background of digital scenes (in digital comics, cartoons and video-games) or real spaces (on a wall in a bar or on a concert stage to augment the environment with a representation of the emotion of the music being played).

Moreover, the paper has also described a study with users which investigated whether the valence and arousal expressed by our model are appropriately perceived by the users or not. The results of this study suggest that combinations of sharpness, height, width and speed are perceived correctly as particular emotions; that sharpness is perceived as valence; that height, width and speed are jointly perceived as arousal; and that the perceptions of both valence and arousal are independent of each other.

As we mentioned above, literature suggests that sharpness of curvature could affect the perception of both arousal and valence. In the presented work, we have explored the capability of sharpness for expressing valence by combining it with other modalities that clearly express arousal. As future work, we will explore other (more abstract) models that follow that approach. Moreover, we intend to study the capability of sharpness for expressing arousal (at the expense of its capability for expressing valence) by combining it with other modalities that clearly express valence (those other modalities could mask the valence denoted by sharpness). Our future research also includes further studies with other new non-anthropomorphic approaches for expressing emotions.

## References

1. André, P., Schraefel, M.C., Dix, A., White, R.W.: Experience in social affective applications: methodologies and case study. In: 2010: Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems, pp. 2755–2764. ACM, New York (2010)
2. Aronoff, J., Woike, B.A., Hyman, L.M.: Which are the stimuli in facial displays of anger and happiness? configurational bases of emotion recognition. *Journal of Personality and Social Psychology* 62(6), 1050–1066 (1992)
3. Bar, M., Neta, M.: Humans prefer curved visual objects. *Psychological Science* 17(8), 645–648 (2006)
4. Bar, M., Neta, M.: Visual elements of subjective preference modulate amygdala activation. *Neuropsychologia* 45(10), 2191–2200 (2007)
5. Bialoskorski, L.S.S., Westerink, J.H.D.M., Van Den Broek, E.L.: Mood swings: Design and evaluation of affective interactive art. *New Rev. Hypermedia Multimedia* 15(2), 173–191 (2009)
6. Dehn, D.M., van Mulken, S.: The impact of animated interface agents: A review of empirical research. *International Journal of Human-Computer Studies* 52(1), 1–22 (2000)

7. Dey, A.K., de Guzman, E.: From awareness to connectedness: The design and deployment of presence displays. In: CHI 2006: Proceedings of the SIGCHI Conference on Human Factors in computing Systems, pp. 899–908. ACM Press (2006)
8. Halper, N., Mellin, M., Herrmann, C.S., Linneweber, V., Strothotte, T.: Towards an understanding of the psychology of non-photorealistic rendering. In: Jochen Schneider, T.S., Marotzki, W. (eds.) Proc. Workshop Computational Visualistics, Media Informatics and Virtual Communities, pp. 67–78. Deutscher Universitäts-Verlag (April 2003)
9. Hevner, K.: Experimental studies of the affective value of colors and lines. *Journal of Applied Psychology* 19(4), 385–398 (1935)
10. Ibáñez, J.: Minimalist approach to show emotions via a flock of smileys. *Journal of Network and Computer Applications* 34, 1283–1291 (2011)
11. Ibáñez, J.: Showing emotions through movement and symmetry. *Computers in Human Behavior* 27, 561–567 (2011)
12. Ibáñez, J.: Emotional clouds: Showing arousal and valence through the movement and darkness of digital cartoonish clouds. *Int. J. Hum.-Comput. Stud.* 71(10), 967–977 (2013)
13. Ibáñez, J.: Emotional sea: Showing valence and arousal through the sharpness and movement of digital cartoonish sea waves. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 43(4), 901–910 (2013)
14. Ibáñez, J., Delgado-Mata, C., Gómez-Caballero, F.: A novel approach to express emotions through a flock of virtual beings. In: CW 2007: Proceedings of the 2007 International Conference on Cyberworlds, pp. 241–248. IEEE Computer Society (2007)
15. Ibáñez, J., Serrano, O., García, D.: Emotinet: A framework for the development of social awareness systems. In: Awareness Systems. Human-Computer Interaction Series, pp. 291–311. Springer, London (2009)
16. Jancke, G., Grudin, J., Gupta, A.: Presenting to local and remote audiences: Design and use of the telep system. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 384–391. ACM Press (2000)
17. Kok, R., Broekens, J.: Physical emotion induction and its use in entertainment: Lessons learned. In: Ciancarini, P., Nakatsu, R., Nakatsu, R., Rocetti, M. (eds.) *New Frontiers for Entertainment Computing*. IFIP, vol. 279, pp. 33–48. Springer, Boston (2008)
18. Lang, P.J.: Behavioral treatment and bio-behavioral assessment: Computer applications. In: Sidowski, J.B., Johnson, J.H., Williams, T.A. (eds.) *Technology in Mental Health Care Delivery Systems*, pp. 119–137. Ablex Pub, Norwood (1980)
19. Leder, H., Carbon, C.-C.: Dimensions in appreciation of car interior design. *Applied Cognitive Psychology* 19(5), 603–618 (2005)
20. Lundholm, H.: The affective tone of lines: Experimental researches. *Psychological Review* 28(1), 43–60 (1921)
21. Markopoulos, P., de Ruyter, B., Mackay, W.E.: Awareness systems: known results, theory, concepts and future challenges. In: CHI 2005: Extended Abstracts on Human Factors in Computing Systems, pp. 2128–2129. ACM Press (2005)
22. McCloud, S.: *Understanding comics: The invisible art*. Harper Paperbacks (1994)
23. Monö, R.: *Design for product understanding: the aesthetics of design from a semiotic approach*. Liber, Stockholm (1997)
24. Oatley, K., Johnson-Laird, P.N.: Towards a cognitive theory of emotions. *Cognition and Emotion* 1(1), 29–50 (1987)
25. Poffenberger, A.T., Barrows, B.E.: The feeling value of lines. *Journal of Applied Psychology* 8(2), 187–205 (1924)

26. Russell, J.A.: A circumplex model of affect. *Journal of Personality and Social Psychology* 39(6), 1161–1178 (1980)
27. Silvia, P.J., Barona, C.M.: Do people prefer curved objects? angularity, expertise, and aesthetic preference. *Empirical Studies of the Arts* 27(1), 25–42 (2009)
28. Wallbott, H.G.: Bodily expression of emotion. *European Journal of Social Psychology* 28(6), 879–896 (1998)
29. Wundt, W.: *Grundriss der psychologie*. Engelmann, Leipzig (1896)
30. Zebrowitz, L.: *Reading faces: Window to the soul?*. Westview Press (1997)



# Application Friendly Voxelization on GPU by Geometry Splitting

Zhuopeng Zhang<sup>1</sup> and Shigeo Morishima<sup>2</sup>

<sup>1</sup> Waseda University, Japan  
zhangzp@asagi.waseda.jp

<sup>2</sup> Waseda Research Institute for Science and Engineering, Japan  
shigeo@waseda.jp

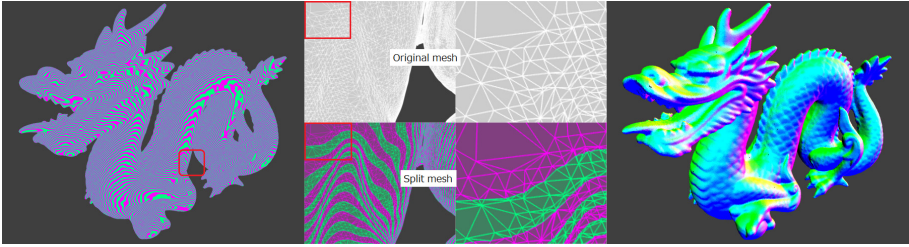
**Abstract.** In this paper, we present a novel approach that utilizes the geometry shader to dynamically voxelize 3D models in real-time. In the geometry shader, the primitives are split by their Z-order, and then rendered to tiles which compose a single 2D texture. This method is completely based on graphic pipeline, rather than computational methods like CUDA/OpenCL implementation. So it can be easily integrated into a rendering or simulation system. Another advantage of our algorithm is that while doing voxelization, it can simultaneously record the additional mesh information like normal, material properties and even speed of vertex displacement. Our method achieves conservative voxelization by only two passes of rendering without any preprocessing and it fully runs on GPU. As a result, our algorithm is very useful for dynamic application.

**Keywords:** voxelization, geometry shader, rasterization, applications.

## 1 Introduction

Voxelization is a process to convert surface geometry to a regular volumetric representation. For 3D objects can be very complex and usually composed by millions of triangles, it is very expensive to do collision or intersection test against each triangle. So a space management technique like octree or voxelization is always necessary to simplify the querying of the spatial distribution of the mesh. The ability of fast testing on the existence of geometry makes voxelization very useful on the solution of collision detection, global illumination and many other graphic applications. For instance, in the rendering of hair under environment light [1], voxelization of the hair is performance-critical. Reinbothe et al.[2] utilize voxelization to speed up ray tracing when applying ambient occlusion. Fluid simulation as [3] uses depth peeling [4], a fast voxelization method to handle deformable boundary. Lattice Shape matching [5] improves the Shape Matching method by voxelization which simplifies the simulation domain as regular lattices.

In some situation, e.g. the rendering of semi-transparent object, voxelization with the interior filled is usually required so as to trace light inside the mesh. This kind of voxelization is called solid voxelization. Otherwise, only the mesh, i.e., surface polygons are voxelized, referred to as surface voxelization.



**Fig. 1.** Stanford Dragon has been voxelized interactively using our geometry splitting method at a resolution of  $512^3$ . Left: The mesh is split by Z-order to different layers. Center: Close view, a triangle crossing different layers are cut by the Z-split-planes. Right: Our method can achieve conservative voxelization straightforwardly, and voxelize mesh with normal information of full precision.

Technically, there are two types of voxelization on GPU: rasterization-based approaches and computational approaches. Rasterization-based approaches rely on rendering pipeline [6]. With the programmable stage, i.e. shaders, we can create and design some rendering strategy as particular voxelizer [7]. Computational methods, mostly basing on GPGPU api like CUDA or OpenCL, usually run in parallel and achieve high performance [8,9].

An ideal voxelization process should be efficient enough, and be flexible as well to support the application. This paper presents a rasterization based algorithm. Its totally based on standard graphic pipeline, so we can avoid handling context switch which is necessary in computational methods (CUDA or OpenCL implementation). Our method, outlined by Fig 1, is easy to implement by just enforcing a geometry shader program, which is responsible for the geometry splitting. Comparing to previous rasterization based methods, our conservative voxelization strategy is simple and straightforward. First, we need not render the mesh for each coordinate plane. Second, our method avoids a lot of operations like modifying the triangle boundary and discarding superfluous fragments. Furthermore, our method has some advantages that are application-friendly. By using our method, a single texture is enough to store the result at full precision. We can encode the residence(i.e. if a voxel is occupied) and face normal into the same texture. With Multiple Render Targets (MRT), it's even possible to encode tangent, color, and (or) motion speed to other textures.

**Other Related Work.** An early hardware voxelization approach is the slicing method [6], which sets clip planes and renders object multi-pass to many slices. When the number of slice increase, this method becomes inefficient for it has to render the target mesh as many times as the slices. Another problem is when certain triangle has a grazing angle in view direction, the voxelization may fail for multiple voxels fall on the same fragments. To address this problem, Dong et al.[7] render mesh in 3 view directions with a final composition step. A conservative rasterization method proposed in [10] can be used in the miss-rasterization

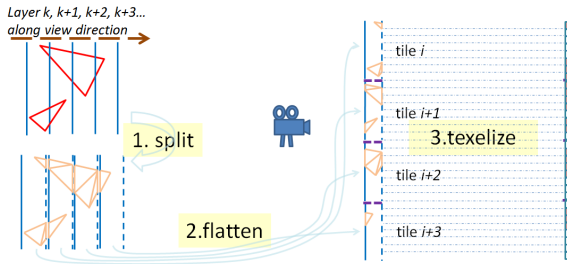
situation. By enlarging the triangle and setting bounding box clipping, it can be used to cover surrounding pixels on triangle edges. Zhang et al.[11] present a conservative voxelization solution that computes the actual depth range of each fragment.

An efficient rasterization based method is the slicemap [12]. This method presents the scene as binary grids, and encodes the bit values into textures using logical operation, each bit can be 0 or 1 for an empty grid or occupied one. This method was improved in [13], resulting solid voxelization within single pass. The main advantage of the slicemap is that it's a single pass method of remarkable performance. The draw-back is that the geometry information is restricted to binary presentation. Value of density or normal should be estimated in additional pass and will be in a very limited range. And a lot of processes as what introduced in [10] and [11] should be performed for conservative voxelization.

Our method evolves from slicing method. However, for arbitrary model, only two passes will be performed.

## 2 The Geometry Splitting Method

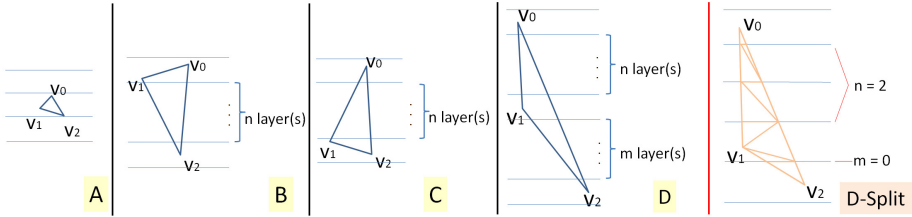
At the time of slicing method [6], triangle can't be split inside graphic hardware. Our method exploits the ability of modern GPU which can create multiple triangles by utilizing geometry shader. Thus triangles can be split to thinner pieces.



**Fig. 2.** The process of the Geometry Splitting Method

As shown in Fig 2, our core algorithm can be summarized as 3 steps: 1. Splitting triangles along view direction; 2. Distributing the split triangle to tiles; and 3. Rasterization. (texelization).

**Step 1. Splitting.** Shown in Fig 3. according to the layout of triangle vertices, we define 4 different patterns of triangle splitting. Pattern A: the whole triangle stays inside the same layer. Pattern B: two vertices of the triangle are inside one layer, and another vertex is inside a farther layer which at a distance of  $n$  ( $n \geq 0$ ) layer(s) from the other two. Pattern C: this case is similar to pattern B, but with the two vertices located at a farther layer instead of being at a nearer one. In the case of pattern D, the first vertex is  $n$ -layer far from the second one and the second one is  $m$ -layer far from the third one.



**Fig. 3.** Patterns of triangles' layout, and an example splitting the pattern D

In the geometry shader, we get the three input vertices of the triangle simultaneously. Firstly, we have to calculate the layer index of each vertex. Then we sort the vertices by the layer indices (make sure  $layer\_index(v_0) \leq layer\_index(v_1) \leq layer\_index(v_2)$ ). After the sorting, we can figure out which pattern current triangle falling into.

For the pattern A, we can simply output the triangle unmodified. For pattern B, C, and D, we do the splitting.

**Step 2. Flattening (transforming to tiles).** In order to voxelize the mesh into a tiled texture, we flatten the pieces of mesh after the splitting.

The voxelization will be completed by the rendering with an orthogonal projection. As we want to create a volume  $P$  of size  $vx \times vy \times vz$ . We setup an off-screen render target, tiled by  $tx$  and  $ty$ . This implies that the number of layers is the same as the tiles ( $vz = tx * ty$ ). And the resolution of the off-screen render buffer is of width as  $tx * vx$  and of height as  $ty * vy$ . And then we scale the pieces of mesh and put each piece to different tile of the render target.

**Step 3. Texelization (rasterization).** Most works have been done in geometry shader including specifying the color which will be recorded in texture. According to certain requirement, geometry shader can transmit normal, texcoord or other essential information to fragment shader. So the fragment shader is quite simple, only responsible for color output. For density estimation, we can use ADD blend mode to sum up the density inside a local voxel grid. For surface normal, we can output it to RGB channels, remaining Alpha channel keeping the occupying flag. And if MRT used, more information can be written in a single execution of voxelization.

A side benefit of our algorithm is the conservative voxelization. We repeat the rendering twice, once by fill mode, and once more using line mode. The splitting of triangle produces edges parallel in view-direction (Z-direction). The minimum width of rasterization of any line is 1, which implies it can always be rasterized. When these edges are drawn by line mode, we can ensure the continuity of voxelization on Z-direction.

### 3 Implementation Details

Our algorithm is implemented using OpenGL. This algorithm also can be implemented by Direct3D and compatible on video cards of feature level Direct3D 10.

OpenGL provides a feature called Transform-feedback, which is used to record primitives generated from vertex shader or geometry shader into buffer objects. If the Transform-feedback feature is available, we can save the new output mesh geometry to a large buffer. Therefore, we can avoid generating new mesh twice in geometry shader, with the cost of a large block of video memory and the operations for the Transform-feedback. But still we have to submit the output primitives twice, once using polygon-mode `GL_FILL`, another using `GL_LINE`.

The geometry shader can be same even resolutions are different; we can put the resolution as the parameters of the shader. Modern GPU usually has a render buffer size limitation equal to or large than 4096. A render buffer of size 4096 is large enough for a voxelization of size  $256^3$ . Our algorithm uses tiled 2D texture; each tile presents one slice; and usually be R8G8B8A8 pixel format. How to calculate the max voxelization size was talked in section 2.

We implemented our algorithm on an AMD Radeon HD 7770. For the case of hair, we only need single pass because the input type is line. Table 1. gives the statistics of our method, the same cases resulted with Transform-feedback are in brackets. For comparison, we also list two cases run by slicemap written in red.

**Table 1.** Geometry splitting performance without/with Transform-feedback. (Number of primitives of each model: Bunny 17,072 tris; Dragon 871,414 tris; Hair 1,509,823 lines.)

	$64^3$	$128^3$	$256^3$	$512^3$
Bunny	4.1ms(3.7ms),-	5.2ms(4.4ms),-	9.8ms(7.1ms),-	-(-),-
Dragon	78.7ms(28.2ms),-	94.5ms(32.5ms),-	118.7ms(62.5ms), <b>22.5ms</b>	154.7ms(68.0ms),-
Hair	18.2ms(-), <b>4.2ms</b>	-(-),-	104.1ms(-),-	-(-),-

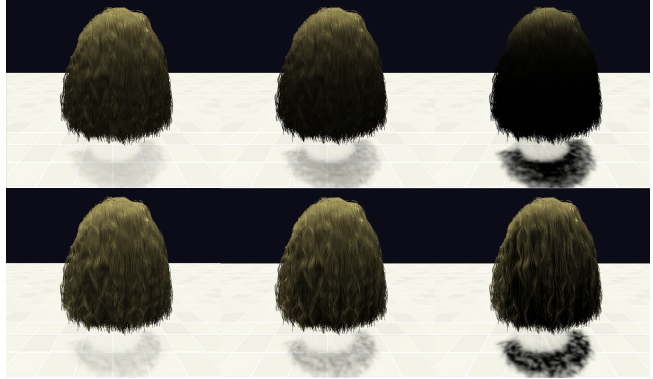
When we compile a geometry shader, we must specify the max output vertex count. For example, if the max output vertex is 64, we can split a triangle to 21 triangles, i.e. to a maximum about 10 layers. Why we omit the data of bunny for large resolutions is that, when the layers increased, some triangles can't be divided completely, leading to some holes on the output mesh. This may be the main limitation of our method.

In table 1, we also get comparisons between our method and slicemap. When using slicemap, for density or normal evaluation, we have to downsample, so the cases of slicemap are run at doubled resolution. Here we know our method consumes three or four timers of slicemap does. It's a tradeoff; later we will show the advantage of our method from the view of usage.

## 4 Applications

### 4.1 Hair Transparency

**Fig. 4.** Rendering hair using voxelization at resolution  $64^3$ . Slicemap voxelization is downsampled from  $128^3$ . The absorb factor  $\kappa$  increases from left to right. The light direction is set downward. These images are all screen-shots of program run at real-time.



Hair models are usually presented as massive line segments. Fortunately, our method is also suitable for voxelizing line primitives. We split lines instead of triangles in geometry shader. And with transform and rasterization using blend mode ADD, we get the density field of hair.

We render hair by follow equation:

$$I = T(x, \omega_i)S(\omega_i, \omega_o) \quad (1)$$

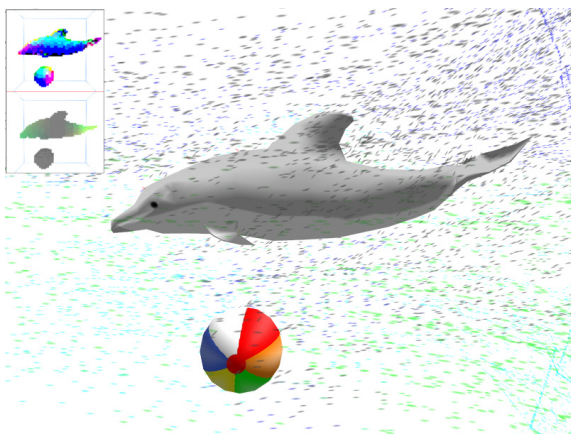
$$T(x, \omega_i) = \prod_{k=1}^n \exp(-\kappa \tau(x_k, \omega_i)) \quad (2)$$

where  $I$  is the final color of hair,  $S(\omega_i, \omega_o)$  is the scattering function described in [14], with lighting direction  $\omega_i$  and eye direction  $\omega_o$ . And the  $T(x, \omega_i)$  is the transmittance[15] at position  $x$ , can be calculated by equation (2). In equation(2),  $\tau(x_k, \omega_i)$  is the density in a voxel  $x_k$  along path  $\omega_i$ , and  $\kappa$  is a user control factor.

In Fig 4, images of up row are the rendering results using slicemap, the ones at bottom are the results using our method. By contrast, the slicemap based shading looks too smooth due to the excessive shadow, while our method provides more details on the variance of density. This is because our method stores the density information in 8-bit float-point numbers, while slicemap only stores 0 or 1. With downsample, slicemap provides 9 possible integer values in  $[0, 8]$ , in comparison, our method has obvious advantage by providing 256 possible values.

### 4.2 Fluid with Dynamic Obstacles

We implemented the fluid simulation by GLSL, totally run on GPU. The obstacle information should be marked for boundary handling of fluid evolution. As



**Fig. 5.** Fluid simulation with voxelization of a deformable object. Leftup image shows the voxelization of the surface normal and the lower one shows the speed of vertices (the vertices are moving up). The fluid solver is on a  $64^3$  grid. The resolution of voxelization is also  $64^3$ . This fluid simulation runs over 100Hz on the same PC previously mentioned.

shown in Fig 5, the residence and normal of mesh are voxelized, we also encode the soft body morphing speed into texture by using MRT. When we perform boundary handling, we enquiry the textures, to know if a grid cell is occupied by object; and if so, we apply reaction to fluid basing on normal and moving speed information. For the obstacle dolphin is morphing between two poses, voxelization is executed every frame. Similarly, our voxelization method can also be combined with skinned animation.

## 5 Conclusion and Future Work

In this paper, we presented a method for surface voxelization. It makes a large improvement on the slicing method by the exploitation of geometry shader. Our method is totally run on GPU and can be applied to arbitrary dynamic scene without any pre-computation. By using our method, the conservative voxelization is so easy to implement just by repeating the rendering in line mode. Our method is a rasterization based method, with simple process and convenient accessing of geometry information. We showed this application-friendly property by two case studies. One case is the hair rendering, illustrated the advantage of our method on the estimation of density information. Another is the fluid simulation with dynamic obstacles, with the ability of our method of outputting normal and displacement simultaneously. Comparing to slicemap, shaders and textures are easy to organize by our method. We can achieve high depth range voxelization using a single texture, while slicemap has to break the layers to several textures. And our method avoids logic operation, performing better on platform independence. Our voxelization also provides full precision of normal, density or speed information directly and efficiently.

The main limitation of our method is the geometry shader output restriction. An input triangle cannot cross too many layers; otherwise the exhausted geometry shader will give up further splitting. This leads holes on output mesh, messing up the voxelization. This could be fixed by a pre-procedure that

detects and tessellates this kind of triangles. Another limitation is that the maximum voxel resolution is limited by render target size supported by hardware. And our method does not provide a hierarchical representation like [8] does, which relatively consumes less video card memory.

We tried solid voxelization using a post-process as [7] mentioned. But the result was inefficient. For better performance, some alternative process should be tested like utilizing the compute shader [16]. And as far as we know, creating vertices is not very efficient in geometry shader.

**Acknowledgments.** Thank for the hair model file generated by Cem Yuksel. Thank for the Stanford 3D scanning repository for the bunny and dragon models.

## References

1. Ren, Z., Zhou, K., Li, T., Hua, W., Guo, B.: Interactive hair rendering under environment lighting. *ACM Transaction on Graphics* 29(4), 55:1–55:8 (2010)
2. Reinbothe, C.K., Boubekeur, T., Alexa, M.: Hybrid ambient occlusion. In: *Eurographics 2009 Annex (Areas Papers)*, pp. 51–57 (2009)
3. Li, W., Fan, Z., Wei, X., Kaufman, A.: Gpu-based flow simulation with complex boundaries. In: Pharr, M. (ed.) *GPU Gems 2*, ch. 47, pp. 747–764. Addison Wesley, Boston (2005)
4. Everitt, C.: *Interactive order-independent transparency* (2001)
5. Rivers, A.R., James, D.L.: FastLSM: Fast lattice shape matching for robust real-time deformation. *ACM Transactions on Graphics* 26(3), 82:1–82:6 (2007)
6. Fang, S., Chen, H.: Hardware accelerated voxelization. *Computers & Graphics* 24(3), 433–442 (2000)
7. Dong, Z., Chen, W., Bao, H., Zhang, H., Peng, Q.: Real-time voxelization for complex models. In: *Proceedings of Pacific Graphics*, pp. 43–50 (2004)
8. Schwarz, M., Seidel, H.-P.: Fast parallel surface and solid voxelization on gpus. *ACM Transaction on Graphics* 29(6), 179:1179:9 (2010)
9. Pantaleoni, J.: VoxelPipe: A Programmable Pipeline for 3D Voxelization. In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, pp. 99–106. ACM, New York (2011)
10. Hasselgren, J., Akenine-Müller, T., Ohlsson, L.: Conservative Rasterization. In: Pharr, M. (ed.) *GPU Gems 2*, ch. 42, pp. 677–690. Addison Wesley, Boston (2005)
11. Zhang, L., Chen, W., Ebert, D.S., Peng, Q.: Conservative voxelization. *The Visual Computer* 23(9–11), 783–792 (2007)
12. Eisemann, E., Décoret, X.: Fast Scene Voxelization and Applications. In: *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, I3D*, vol. 17, pp. 71–78. ACM, New York (2006)
13. Eisemann, E., Décoret, X., Single-Pass, G.P.U.: Solid Voxelization for Real-Time Applications. In: *Proceedings of Graphics Interface 2008, Canadian Information*, vol. 17, pp. 73–80. Processing Society, Toronto (2008)
14. Marschner, S.R., Jensen, H.W., Cammarano, M., Worley, S., Hanrahan, P.: Light scattering from human hair fibers. In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, vol. 22(3), pp. 780–791 (2003)



15. Zinke, A., Yuksel, C., Weber, A., Keyser, J.: Dual Scattering Approximation for Fast Multiple Scattering in Hair. *ACM Transactions on Graphics* 27(3), 32:1–32:10 (2008)
16. Rauwendaal, R., Bailey, M.: Hybrid Computational Voxelization Using the Graphics Pipeline. *Journal of Computer Graphics Techniques* 2(1) (2013)

# A Pattern-Based Tool for Creating Virtual Cinematography in Interactive Storytelling

Pei-Chun Lai<sup>1</sup>, Hui-Yin Wu<sup>1</sup>, Cunka Sanokho<sup>2</sup>,  
Marc Christie<sup>2</sup>, and Tsai-Yen Li<sup>1</sup>

<sup>1</sup> National Chengchi University, Taiwan  
{100753017,li}@nccu.edu.tw

<sup>2</sup> IRISA/INRIA Rennes, France  
{marc.christie,hui-yin.wu}@inria.fr, cunka.sanokho@irisa.fr

**Abstract.** In this work we design a tool for creators of interactive stories to explore the effect of applying camera patterns to achieve high level communicative goals in a 3D animated scenario. We design a pattern language to specify high level communicative goals that are translated into simple or complex camera techniques and transitions, and then flexibly applied over a sequence of character actions. These patterns are linked to a real-movie shot specification database through elements of context such as characters, objects, actions, and emotions. The use of real movies provides rich context information of the film, and allows the users of our tool to replicate the feel and emotion of existing film segments. The story context, pattern language and database are linked through a decision process that we call the virtual director, who reasons on a given story context and communicative goals, translates them into the camera patterns and techniques, and selects suitable shots from the database. Our real-time 3D environment gives the user the freedom to observe and explore the effect of applying communicative goals without worrying about the details of actor positions, scene, or camera placement.

**Keywords:** context-aware camera control, interactive storytelling, virtual cinematography, computer animation.

## 1 Introduction

In film theory, Jakob Lothe [1] describes the mechanisms of discourse as how “people are presented through characterization, and the transmitted content filtered through narrative voices and perspective.” Discourse, such as that in film, is how the director guides and awes the audience to understand the multiple emotions, feelings, and other hidden messages in each story scenario.

With the advances of multimedia and virtual storytelling comes the benefit of quick and realistic cinematography in 3D gaming environments. Existing machinima tools can automate camera placement to capture action in well-defined story scenarios based on certain elements of context such as number of characters, action, and scene type. However, the power of cinematography lies in its capacity to communicate the story-actions, emotions, and morals—which we



**Fig. 1.** Example of an intensification sequence from the film *White Ribbon*. Our objective is to encode both the compositions of the shots (hence the eyes and facial annotations) and their context, and reproduce the discourse pattern corresponding to intensification.

term as communicative goals. From film literature, we observe a number of well-established camera techniques—which we define as “patterns” of shots in our system—used to convey certain messages. An example of this is the “intensification” technique that consists in placing the camera closer and closer to the actors at each cut in order to communicate a building of emotion within the dialogue (see Fig. 1). As in the making of film, creators of interactive stories and games for 3D environments would need cinematographic tools that can achieve multiple communicative goals in each scene, but also respond to dynamically changing story lines. Thus, a simple but expressive method of defining and applying camera techniques to construct elaborate levels of discourse is still highly desired.

Thus our objective it to develop smart tools for creators of interactive stories to search and experiment with various camera techniques that can fulfil multiple communicative goals in a given scenario, find shots that fit camera patterns for a sequence of actions, and observe their effects in a real-time 3D environment.

This objective is achieved in three steps: (1) we propose a means to create a database of shot specifications by annotating shots from real movies in order to reapply them, (2) we design a pattern language to express communicative goals as a construction of cinematographic techniques, and to express techniques as patterns defined over sequences of shots, and (3) we propose a decision-making process (a “virtual director”) for smart cinematography that finds suitable shots and transitions based on the current story context and specified communicative goals, and performs the according shots and edits in a real-time 3D environment.

## 2 Related Work

Related work in computer animation and interactive storytelling reveals the growing importance of automated camera planning techniques in virtual environments for storytelling and gaming scenarios. He [2] was the first to propose an automated cinematography system in character-based interactive scenarios, and established a state machine mechanism to describe the rules for camera control. Other constraint-based camera systems include [3,4] for game scenarios. Such methods view cinematographic rules as constraints to camera planning problems, which works effectively for simple scenarios such as dialogue and basic movements such as walking, but do not account for contextual story elements

such as emotions, genre, or style. Work on interactive storytelling have also proposed the use of virtual cinematography, demonstrating the importance of defining camera idiom languages [5], communicate emotions in a shot [6,7], or ways to interpret perspective in terms of the story [8].

The growing applications of virtual cinematography in interactive storytelling points to the development of tools that can create story-driven and context-aware virtual cinematography.

The machinima production tool, Cambot, developed by [9] implements a library of “blockings” which are stage plans (the scene in the 3D environment) of where characters are placed and how they move around the stage. Then cameras are placed by looking for shots that are suitable for a given stage and blocking. This method has the strength to bind the blockings, staging, and camera motions, ensuring the tight coordination between the actors, scene, and camera before a scene is shot. In our work, using the toric manifold method proposed by [11] we open the capacity for computing camera positions dynamically for scenarios with unknown stage and actor configurations, and dynamic gaming scenarios. We also provide a method to structure links between shots that maintain a style over a series of story actions.

The Darshak system [10] uses the bi-partite model of story to generate cinematic discourse for a given story plan. The system has the strength of reasoning over the current story scenarios by designing its discourse planner on the level of scenes, instead of individual actions. The system’s camera configurations must be defined in advance for each context that will be shown (such as Show-Robbery). In our work we emphasise the reusability of existing patterns across different story scenarios that have similar context or emotions, and also provide multiple stylistic camera options for desired communication goals.

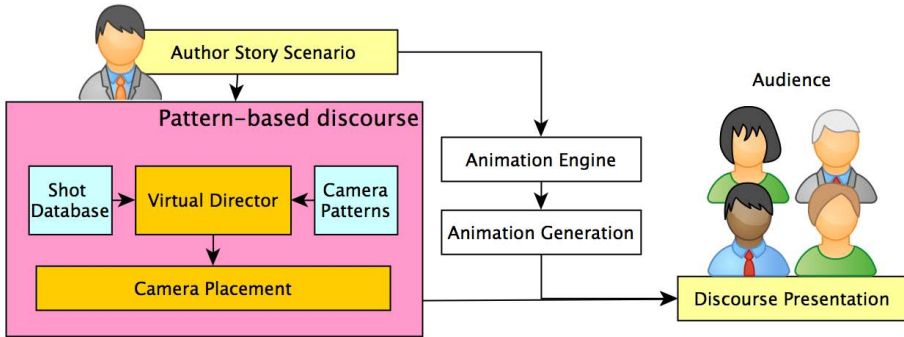
From the literature we observe the necessity for camera tools that can identify story context, and provide familiar camera techniques for creators of 3D interactive stories to apply and experiment with combinations of techniques and thus achieve their communicative goals.

### 3 Overview

The main objective is to develop a tool for creators of interactive storytelling to explore means of achieving communicative goals in 3D environments. The tool should allow easy specification of camera patterns with well-established cinematographic techniques.

This objective can be described in three parts: a language for users to specify style and communicative goals, a database of shot specifications annotated from real movies, and a decision-making process to identify story context and communicative goals and translates them to shot specifications from the database.

In order to achieve the above objectives, we design a pattern-based representation of discourse to encode communicative goals and camera techniques, the shot database, and a virtual director that reasons on the story context, goals, and performs the lookup in the database for shot specifications (see Fig. 3).



**Fig. 2.** Role of the discourse based camera in the framework

The shot database is established by annotating shots from real movies for the purpose of establishing a link between the story context and shot specification in existing film literature. The virtual director receives context information on the upcoming scene on three aspects: (a) the current actions being carried out, (b) the characters involved and (c) the communicative goals for that scenario, and translates communicative goals into camera patterns which are comprised of specific shot types or transitions. A lookup in the database is performed based on the shot type and story context, and produces the most suitable shot specifications and transitions for the given scene. The shots selected by the director are filtered by occlusion detection and the output of our tool is the camera position of the best selected best shot for the scenario. The camera is then placed in the virtual environment using the placement algorithm and the scene is shot.

### 3.1 Context and Communicative Goals

The first input of the tool is the 3D animated scenario comprised of a scene, virtual actors, communicative goals for each scene (such as *build emotion*, *compassion*, or *horror*), and high level actions that are carried out (such as *escape danger*, *tell secret*, or *discuss murder*). This gives the virtual director an idea of what actions are being carried out, which characters are involved, and the physical locations in the 3D environment.

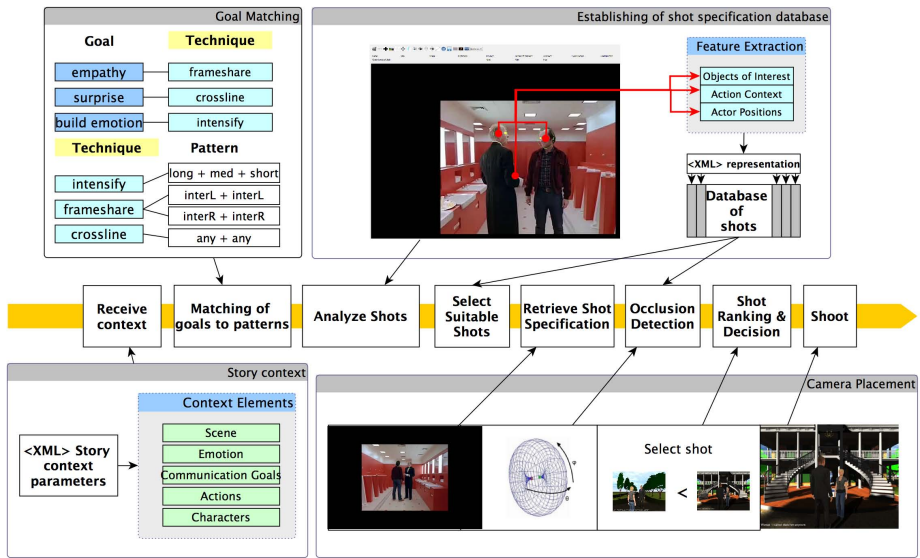
Our story is represented by an XML scenario which specifies the scene, character position and actions, and story context (such as emotion parameters) in high level commands. The specification of the story context also defines the point at which the camera can make a cut. A sample of the context description is:

```

<AnimContext Scene="discontent" Action="tellSecret"
Goal="buildEmotion" Actor1="millionaire" Actor2="woman" />

```

Where **Scene** represents the name of the scene. **Action** represents the action that is carried out by the characters, such as walking, talking, running, etc. **Goal**



**Fig. 3.** Workflow of the context aware camera control which involves four main components: the given story context, a pattern library, shot database, and camera placement

is the communicative goal that the author wishes to convey, such as emotional intensify, horror, affinity, isolation, etc. Actor + Number represents the IDs of the character involved in the specific action. In the above example, the XML represents the statement “millionaire is telling a secret to woman with communicative goal of buildEmotion.” Each scene may have any number of actions and communicative goals, and the virtual director parses and extracts all the necessary elements.

A communicative goal can then be expressed with a collection of techniques. Each technique can then be described in terms of a pattern. A pattern defines constraints on the types of shots and types of transitions between shots that are needed to encode the technique. In the following example, the technique “intensify” is expressed by first selecting the longest possible shot (longest in terms of shot size, i.e. farthest from the characters). Then, transitions should be made to shots closer than the initial shot (i.e. getting closer and closer to the characters).

```
<technique id="intensify">
    <initial operation="longest"/>
    <transition operation="closer"/>
</technique>
```

Another way to define intensify could be more strict, by directly constraining the type of shots in the sequence, starting with a long shot, followed by a medium shot, and then a close shot.

```

<technique id="intensify3">
  <initial shot="long"/>
  <next shot="medium"/>
  <next shot="close"/>
</technique>

```

Our system provides a number of parameters that can be used to define the techniques. These patterns include operations such as *any* (any shot), *opposite* (opposite angle of a shot), *sameside* (check the characters are framed on the same side of the shots), *longest*, *closer*, and *continuity* (respect of classical continuity editing rules between shots), or specific shot types such as *internal*, *external*, *parallel*, *long*, *medium*, and *close*, etc. Since directors may want to convey many levels of discourse in a single scene, a benefit of our tool comes with the capacity to combine a number of techniques using multiple communicative goals (e.g. buildEmotion with crossline). Communicative goal can be achieved with different techniques (one of them will be selected at runtime according to the quality of the first shot in the sequence).

## 4 Shot Database

We annotate and output shot specifications from real movies in to establish the shot database. The use of real movies provides the system with information on what decisions directors make in real filmmaking. Each shot specification in the database provides information that links elements of visual composition in cinematography (pertaining to where characters positions, shot angle, size, distance...etc.) to elements of context (what is happening in the story).

Both geometric and story information is annotated in the shot specification. As in the example of Fig. 1, each shot is annotated with eye positions, head position, shot distance, and body orientation (the first two through a point-and-click, and the other two through estimation). Moreover, story context is annotated in the form of characteristics, where the actor name, the action being carried out, and the object on which the action is performed is recorded. A sample output is as below. Note that the object can be other actors or inanimate objects:

```

<Shot ShotName="ribbon-b-6" NbActors="2">
  <Actor Name="Man" Action="talk to" Object="Woman"
    EyesPosition="0.681 0.166 0.754 0.145"
    Distance="5" ShotType="OVSR"
    BodyDirection="0.875"/>
  <Actor Name="Woman" Action="listen to" Object="Man"
    ..."/>
</Shot>

```

The database holds 77 shots extracted from real movies using our cinematographic tool for scene annotations. A sample XML entry would be as the following:

ShotName is an ID given to the shot, and NbActors is the number of actors/points of interest in the shot. Each actor has a number of characteristics. The eye position is relative to the frame, supposing the width and height are both 1. This allows cinematographers to reproduce the same shot in different frame sizes. The distance parameter has a scale from 1 to 10, where 1 is an extreme close-up, and 10 is an extreme long shot or establishing shot, and 6 would be a medium-long shot similar to an American shot.

## 5 Discourse-Based Camera Control

Like a screenplay is to the film director, our story script and shot database is the reference for our tool to realise the 3D scenario. In our tool, the virtual cinematographic director and camera placement algorithm play respectively the film director—in deciding the framing composition, the timing, pacing, organization of the scene and character movements—and the cameraman that shoots a scene.

### 5.1 Virtual Cinematographic Director

We have defined communicative goals such as *Horror*, *BuildEmotion*, and *Compassion* to be realized by a combination of cinematographic techniques such as *Crossline*, *Intensify*, and *Frameshare* (more can be defined by specifying different patterns of shots). *Crossline* is a situation in which the camera begins with any shot and crosses lines (often creating surprise or a sense of uneasiness). *Intensify* happens when the camera moves closer and closer to the ongoing action location, intensifying the context, action, emotion or relationship. *Frameshare* is when the camera begins with an internal shot, and switches between actors while keeping them on the same side of the frame (generally creating a bond between the characters).

The communicative goals and shot specifications in the shot database provide the virtual director with the necessary information to decide on the suitable camera techniques over a sequence of story actions.

Since there is no direct mapping between each context to shot specification, the virtual director engages in a decision process that involves searching for camera techniques to a sequence of actions in the database of shots and the library of camera techniques.

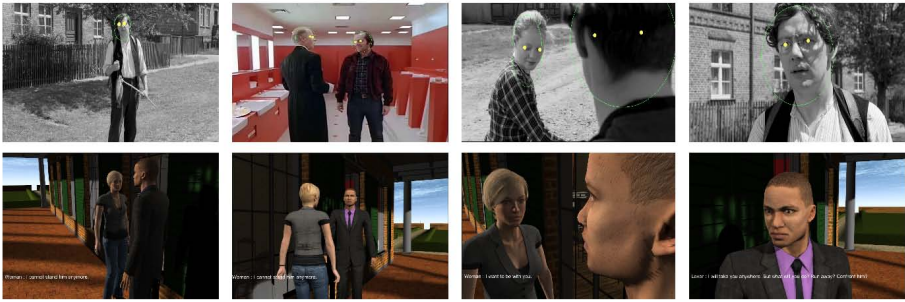
The workflow of the virtual director is as follows: First, the context of the scene is extracted. The director receives the number and names of actors involved, the action, and the communicative goal. Communicative goals are translated into camera techniques expressed in terms of patterns, which are again translated into a number of shot specifications from the shot database. From the number of characters and action, the cinematographic director looks in the database for shots that fit the context description as well as the pattern. One thing to note is that when a given scene has 2 people, the camera configuration can be either two people in the same frame or one person (cutting shots between two characters such as in dialogue scenarios).



All the shot specifications that are suitable to the context are selected and extracted for the camera placement procedure.

We provide two examples of how this process takes place:

**Example 1: Intensify Pattern.** In our tool, “intensify” is defined as one of the techniques that can achieve emotional buildup (such as in our previous example in Fig. 1). Suppose the user has a goal to build emotion, the virtual director first looks up the goal from the mapping. The director finds that the technique “intensify” can achieve “BuildEmotion” using the pattern *long + medium + close* or the pattern *longest + closer*. The former directly defines the types of shots to be used while the latter is more flexible in defining a pattern of shots with a transition criteria (i.e. to move closer for each consecutive cut in the sequence). Fig. 4 shows how shots from the database can be recombined to generate an intensification sequence based on the above patterns.



**Fig. 4.** How shots in the database are reused and recombined to generate the intensification sequence

**Example 2: Crossline Pattern.** In filmmaking, the 180 degree rule describes the consistency of the spatial composition in a frame concerning the relationship between two characters or objects in a scene: the axis is the line that goes through the two characters, and the camera should remain on one side of the axis for each shot in the same scene, ensuring that the relative positions of the two characters is fixed. Sometimes the line can be crossed (i.e. the “crossline” technique), switching the composition of the two characters in the frame, thus creating a confused sense of space.

When two characters are present in a scene, we use their positions in the 3D scene to establish a line called the axis. This can be used to maintain the 180 degree rule. The placement of the camera provides a basis in determining which side of the line of continuity the camera should be situated. Using the axis as a basis, we can then define the patterns “continuity” as *any + sameSide* or “crossline” as *any + (-sameSide)*.

## 5.2 Camera Placement

In the shot database, there may be many shots that fit the context of the sequence of actions in the story. However, the shot context does not account for camera style nor for 3D scene components, which may result in problems such as occlusion of characters, or the inconsistency of camera styles such as the line of continuity (otherwise known as the 180 degrees rule). Therefore, it is necessary to check for shots to ensure critical parts of the scene, such as the character's face, critical scene components, or objects of interest are present in the frame.

After the director has selected a number of shots, this data is sent to the virtual cinematographer. The role of the cinematographer is to calculate the camera position for a specific composition and make the shot. The amount of occlusion and camera distance to subjects is calculated and returned to the director. Finally, the shots are ranked according to the occlusion, and the best shot is selected and executed.

We use the computing method proposed by [11] to find the exact on-screen positioning of two or three subjects, where the solution space for the exact composition of two subjects is represented as a 2D manifold surface. The manifold is defined by two parameters  $\varphi$  (the vertical angle of the camera) and  $\theta$  (the horizontal angle). The search algorithm finds a 3D camera position  $P$  such that the two key subjects  $A$  and  $B$  ( $A \neq B$ ) project respectively at normalized screen positions  $p_A(x_A; y_A), p_B(x_B; y_B) \in [-1; +1]^2$ , and the camera orientation is algebraically determined from the camera position.

With the camera position calculated for each shot, the camera can be placed in the environment to evaluate the continuity of camera style and occlusion of a scene.

**Occlusion.** In order to detect occlusion, we first establish bounding boxes roughly representing the volume of actors, then by initiating a ray cast in the system, we can detect the presence of objects between the camera and the target points (the points on the character bounding box). We rank the shots according to the number of occlusions detected. The shot with the fewest number of occlusions is selected.

In our implementation, 8 lines are casted on the bounding box of each character in the context. Shots are then ranked according to the number of lines that reach the target points. When more points of interest are added the ranking can really take into better account of objects of interest, and find shots that would be more informative in terms of visible interest points.

## 6 Demonstration

The simulation of our interactive story is performed by a 3D animation engine featuring a smart control of characters and scenarios, which we refer to as *The Theater*. Each story fragment (i.e. piece of story) is linked to an authored XML script, which describes the high-level actions and character behaviours

occurring in the fragment, together with abstract locations. The author specifies behaviours at a semantic level (e.g. meet someone in a given room, find a specific object or exchange utterances with a character). He also specifies the communicative goal to achieve in the story fragment. Our 3D character animation engine then computes the exact locations, paths and 3D animations for the characters, given the geometry of our 3D environment and the behaviours to simulate.

## 6.1 Example Scenario

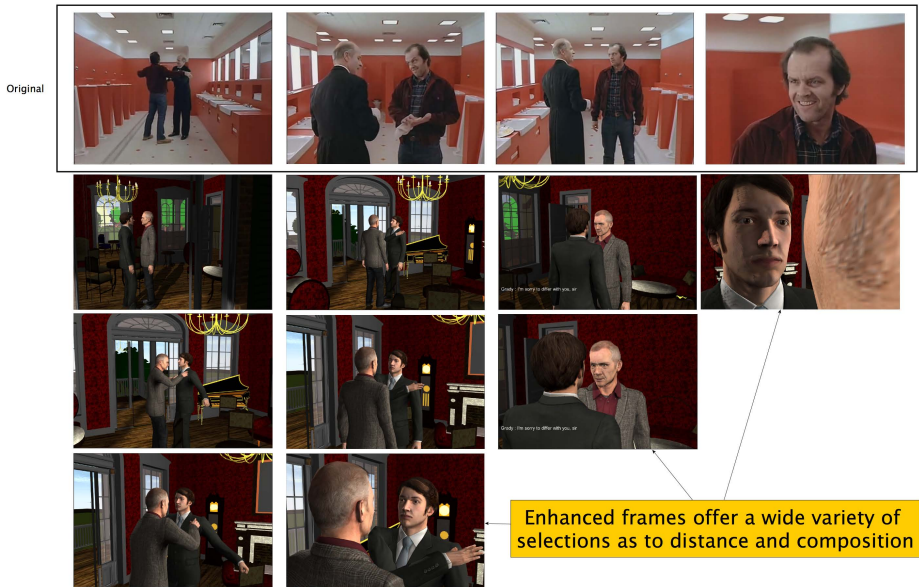
In our sample scenario we use a dialogue scene from the movie *The Shining* where the antagonist Jack Torrence is talking to a ghost Delbert Grady about past events in the Overlook Hotel, where Jack is the winter caretaker. While they chat, Jack interrogates Grady about Grady’s horrifying past (Jack: “You eh... chopped your wife and daughters up into little bits...and you blew your brains out”). The director of the original film chose a crossline technique to create a feeling of split personality in Jack, giving the viewer a sense of confusion and horror.

In our example, we replicate the technique by using the crossline technique, which places the camera on opposite sides of the establishing line for the purpose of creating the feeling of horror. Suppose apart from horror, one would also like to build emotion as the dialogue progresses. The virtual director receives the communicative goal of “Horror” and “BuildEmotion” for the dialogue. The two goals are looked up in the pattern library and are translated into two patterns: crossline and intensify. The virtual director then selects shots that fulfill both the distance requirement (i.e. starting from a long shot and moving closer) and the crossline requirement (i.e. switching between the two sides of the axis). The result comparison of the second example is shown in Figure 5. A video of the scenario can be found at: <https://vimeo.com/93258366>

From the demonstration we can see that the tool we developed can provide a number of possibilities to interpret and shoot communicative goals, and also how multiple communicative goals can be specified to experiment multiple levels of discourse in a single scene. The context—including the story, location, character positions, and actions taking place—are provided by the system in real time while the virtual director makes decisions on the shot specification to use based on the communicative goals, translated into the camera techniques that are described in terms of shot patterns.

## 7 Future Work

Our work in this paper provides creators of interactive stories a simple and efficient tool for defining and experimenting with cinematographic styles in 3D environments. As a first step to creating the link between story context, communicative goals, to shot specifications and camera placement in the 3D environment, though we believe a more complex model of linking story context could



**Fig. 5.** Enhancing of an existing scene with variety of choices of shots that fulfill certain criteria

greatly improve the final results. For example, though we currently use occlusion as the primary factor for ranking shot specifications, selecting shots with least occlusion may not always be the most desirable when occlusion is used as a framing technique for creating emotions such as mystery or suspense. By allowing users to specify elements of story context, style, and importance of objects in the frame that should be taken into account for the ranking, the decision process of the virtual director can be further enriched.

When we view the tool as a potential for assisting creativity, our next step is to evaluate the use of the tool in real scenarios: game design, film production, and machinima. From understanding the needs of application scenarios, we also envision easy-to-use interfaces and authoring tools for storytellers and cinematographers to quickly create story scenarios, and select, edit, and apply camera shots to scenarios, while maintaining conformity to story logic and camera style.

## 8 Conclusion

In this paper we have introduced a tool to assist creators of interactive stories in enriching machinima discourse and achieving their communicative goals through a simple and expressive pattern language developed from well-established camera techniques in film. We enforce the link between discourse and story context through the building of a shot database that provides rich context information

from existing shots in film literature. Our animation system further allows users of the system to immediately adjust and observe the effects of the chosen communicative goals. Through a demonstration of replicating and enhancing a dialogue scene from the movie *The Shining*, we show how multiple effects can be combined to allow the user to experiment various cinematographic representations and build elaborate levels of discourse.

## References

1. Lothe, J.: *Narrative in Fiction and Film: An Introduction*. Oxford University Press (2000)
2. He, L.W., Cohen, M.F., Salesin, D.H.: The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In: *Proceedings of SIGGRAPH 1996*, pp. 217–224. ACM Press (1996)
3. Drucker, S.M., Zeltzer, D.: Intelligent camera control in a virtual environment. In: *Proceedings of Graphics Interface*, vol. 94, pp. 190–199 (1994)
4. Bares, W.H., Thainimit, S., Mcdermott, S.: A Model for Constraint-Based Camera Planning. In: *Smart Graphics* (2000)
5. Amerson, D., Kime, S.: Real-time cinematic camera control for interactive narratives. In: *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pp. 369–369. ACM Press (2005)
6. Kneafsey, J., McCabe, H.: Camerabots: Cinematography for games with non-player characters as camera operators. In: *DIGRA Conf.* (2005)
7. Lino, C., Christie, M., Lamarche, F., Schofield, G., Olivier, P.: A Real-time Cinematography System for Interactive 3D Environments. In: *2010 ACM SIGGRAPH Eurographics Symposium on Computer Animation*, vol. 1, pp. 139–148 (2010)
8. Porteous, J., Cavazza, M., Charles, F.: Narrative generation through characters' point of view. In: *Proceedings of the 9th AAMAS Conference, International Foundation for Autonomous Agents and Multiagent Systems*, pp. 1297–1304 (2010)
9. Elson, D.K., Riedl, M.O.: A Lightweight Intelligent Virtual Cinematography System for Machinima Production. In: *Proceedings of the 3rd Conference on Artificial Intelligence and Interactive Digital Entertainment, Palo Alto, California, USA* (2007)
10. Jhala, A., Young, R.M.: Cinematic Visual Discourse: Representation, Generation, and Evaluation. *IEEE Transactions on Computational Intelligence and AI in Games* 2(2), 69–81 (2010)
11. Lino, C., Christie, M.: Efficient Composition for Virtual Camera Control. In: *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2012)

# Semantically Consistent Hierarchical Decomposition of Virtual Urban Environments

Carl-Johan Jorgensen and Fabrice Lamarche

IRISA/University of Rennes 1, France  
{cjorgens,flamarch}@irisa.fr

**Abstract.** When planning a path in their environment, humans reason on a hierarchical representation of this environment. They first plan a path through coarse zones, then refine this path during navigation, as relevant information is perceived. In this article, we propose a method that automatically generates a semantically consistent hierarchical decomposition of an urban environment. We also present a path planning process that takes advantage of this representation to delay some decisions and propose path options that enable a smart path adaptation when unexpected events occur.

**Keywords:** Hierarchical, environment partition, path planning.

## 1 Introduction

When planning a path in their environment, pedestrians do not consider every detail at once. Instead, people first plan a coarse path, choosing streets to travel to reach their goal. Local decisions such as where to cross a street or on which side to pass by a pole are taken during navigation. In computer science, hierarchical representations of an environment are often used to reduce the computation cost of the planning algorithm [1][2]. Such a representation also enables smarter navigation behaviours. Indeed, it offers the opportunity to delay the local planning until relevant information is available. It also enables a quick recovery from unexpected events, as the high-level path might stay valid even if unexpected events alter the lower-level path. In order for these smart processes to generate credible results, the environment must be partitioned into semantically coherent zones like streets, crossroads or crossings. However, in most current works, the hierarchical decomposition of the environment is computed using purely geometrical approaches.

**Contribution.** In this paper, we propose a method that automatically generates a three-level hierarchical representation of an informed urban environment. In this hierarchy, each level is a semantically coherent partition of the navigation areas and can be used to plan paths at different levels of abstraction. The first level decomposes the environment into street segments, crossroads and pedestrian areas. The second level decomposes these streets and crossroads into

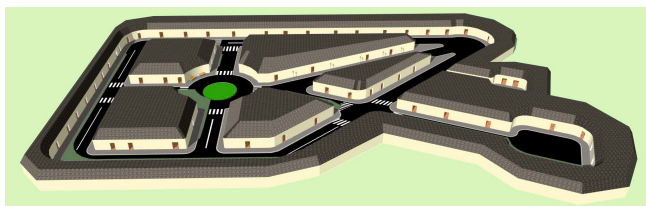
sidewalks and road segments. The third level is composed of atomic navigation zones like crosswalks, building access or sections of sidewalks.

In section 2, we give an overview of existing environment representations and hierarchical path planning methods. In section 3, we explain how to compute the hierarchical representation of the environment. In section 4, we depict our hierarchical path planning algorithm and its relation to reactive navigation. Finally, in section 5, we demonstrate the effectiveness of our partitioning method for different kind of environments and the good properties of our path planning algorithm.

## 2 Related Work

Virtual agents' navigation behaviour relies on the ability to plan a path inside a complex environment. This ability depends on the adequacy of the environment representation. In robotics, where navigation is compulsory, many methods were proposed to represent the environment typology [3]. Two main techniques can be distinguished: the roadmap and the cell decomposition approaches. The roadmap approach captures the connectivity of the free space thanks to sets of standardized paths [4][5][6]. This concept has been extended to corridor maps that add information on the distance to obstacles [7]. However, this kind of approach focuses on creating a data structure enabling path planning but does not explicitly model the borders of obstacles, which is a prerequisite of our approach. The cell decomposition approach represents the free space with a set of cells. Those decomposition approaches can be either approximate or exact. Approximate cell decompositions use predefined cells shapes (uniform grids, quad trees, circles) whose union is strictly included in the free space [8][9]. Exact cell decompositions exactly cover the free space. Among other techniques [3], Constrained Delaunay Triangulations have been used in the last ten years to compute exact cell decompositions of virtual environments [10][11] and have been slightly modified to identify bottlenecks (most constrained part of the environment) [12]. Their good properties in terms of navigation queries [13] and path finding [14] have been demonstrated. To handle more complex cases and increase the realism of simulations, semantic information on the nature of the environment must be taken into account. The notion of informed environment has been introduced. It associates a data structure to regions of the environment. This data structure stores information related to the behaviour of agents [8]. This approach has been used to model populated cities [15] and to propose semantic abstractions of a city structure [16]. This kind of semantic information has also been used to produce terrain-dependent paths that reflect the agents' preferences [17].

The idea of hierarchical representation of an environment has been proposed to reduce path planning complexity while avoiding path degradation [1][2]. However, these methods usually rely on purely geometric partitions of the environment, without taking semantic information into account. Based on constrained Delaunay Triangulations, several techniques have been proposed to produce a hierarchical abstraction of the environment based on its topology. The main



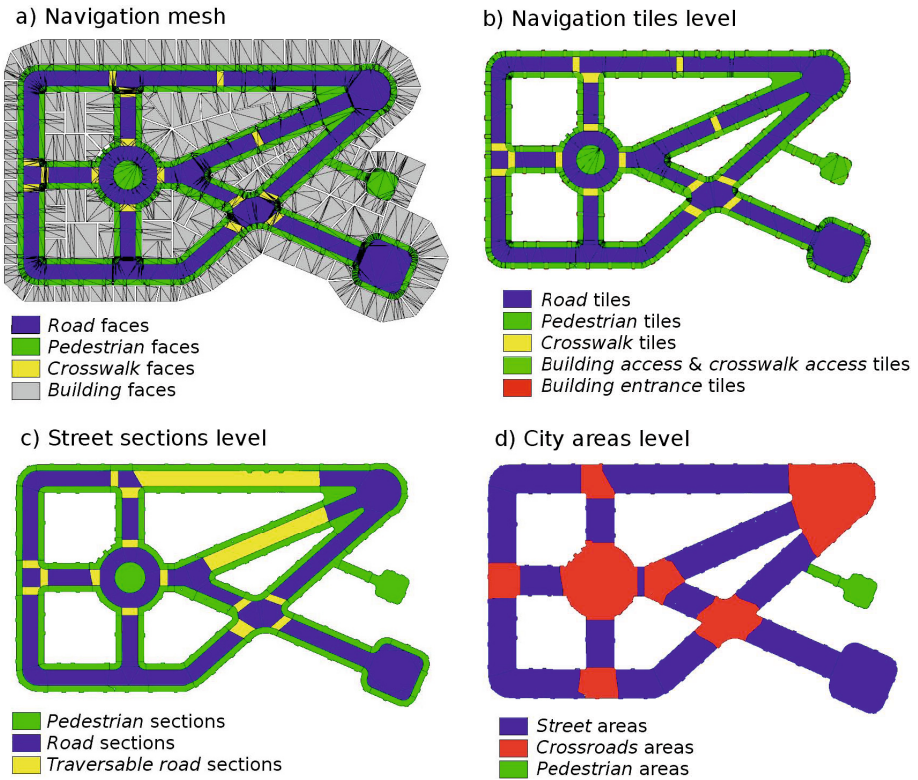
**Fig. 1.** A simple urban environment example

idea is to qualify cells or sets of cells (regions) given the number of accesses (dead ends, corridors, crossings) [12][18]. On this basis and on the basis of the geometric shapes of the identified regions, a topological abstraction is automatically computed. Once again, this technique was mainly used to increase path planning performances. More recently, informed environments were used to produce more semantically coherent hierarchical representations. In [19], Jiang et al. proposed a hierarchical decomposition of multilayered environments taking into account notions of floors and stairs. In the field of geographic information systems, semantic information is used to provide an informed and hierarchical representation of virtual geographic environments, in order to speed up and enhance the quality of the computed paths [11]. Such kinds of informed hierarchical environments were automatically generated for indoor environments and used to generate smarter agents' navigation behaviours [20]. However, none of these methods offer to automatically extract a semantically coherent hierarchical decomposition of virtual urban environments.

### 3 Hierarchical Partition Process

When navigating in a city, pedestrians take different kinds of decisions. At first, a global path planning is achieved by considering the city as a network of streets. Then, while following the path, more local decisions are taken considering the sidewalks and crossing possibilities offered by the crosswalks. Finally, even more locally, pedestrians can choose, depending on their perception of their surroundings and their current situation, to locally adapt their path by crossing the street or choosing a side to bypass a static obstacle like a bench or a pole. In this section, we present a hierarchical representation of the environment that helps to model such kind of decision making. This representation relies on a navigation mesh automatically extracted from an informed environment [15] [16]. Information included in the environment identifies zones as being buildings, pedestrian zones, roads or crosswalks. The first level, named "navigation tiles", subdivides the environment into small zones, named tiles, that identify roads, sidewalks, crosswalks, crosswalk access and building access. At this level, a precise path can be computed. The second level, named "street sections" identifies pedestrian sections and traversable or non-traversable roads. At this level, a pedestrian can choose a sidewalk, where to cross a street or anticipate the lack of crosswalks,





**Fig. 2.** The navigation mesh and three levels of decomposition of our example environment

for example. Finally, the third level, named "city areas", subdivides the environment into streets, crossroads and pedestrian areas. Those areas are used to plan a coarse path through the city. In the following, we detail how those levels are automatically generated and discuss their good properties in terms of decision making.

### 3.1 Almost Convex Zones

While pedestrians plan their global path through the city prior to travel to their destination, local decisions are delayed to the moment these decisions are needed. Many of these decisions rely on dynamic information like traffic density or traffic lights. These decisions require that the pedestrian perceives the area before making his choice. Partitioning the environment into convex zones would ensure an agent in a zone to perceive this entire zone. Nevertheless, in real situations, small obstacles like poles or benches, as well as the exact shapes of walls do not affect much of the pedestrian perception. In order to determine semantically consistent local zones, we use the concept of "almost convex zones".

A zone is considered almost convex if the ratio of its surface divided by the surface of its convex hull is higher than a given threshold. We also define the fact of "improving a zone convexity" if merging another zone to this zone decreases the difference between the zone surface and the surface of its convex hull.

### 3.2 Navigation Mesh

The first processing step extracts an informed navigation mesh from an informed environment geometry (Cf. figure 2.a). This navigation mesh is a Delaunay triangulation, augmented with bottlenecks [12]. Extracted triangular cells are informed with the nature of the navigation zone provided by the informed environment geometry: *pedestrian*, *road*, *crosswalk* or building cells. Edges of the triangulation model a zone nature change (frontier between road and sidewalk for example), an obstacle border or a free edge that can possibly be a bottleneck.

### 3.3 Navigation Tiles Level Generation

During navigation, a pedestrian chooses where to cross a street or on which side to bypass a pole for example. Such a decision is made considering small semantically homogeneous zones like crosswalks or portions of sidewalks. The navigation tiles level is designed to provide a representation adapted to such low level decisions (Cf. figure 2.b). The environment is partitioned into tiles which are composed of adjacent triangular cells of the same type. In a tile, a unique simple path links a border to another. At this level, six different kinds of tiles are identified: *road*, *crosswalk*, *pedestrian*, *building entrance*, *crosswalk access* and *building access* tiles. Partitioning is achieved by the following process:

1. As we focus solely on exterior environments, only building entrances are kept in the decomposition. *Building entrance* tiles are defined as convex unions of building cells, adjacent to at least one *pedestrian* cell.
2. *Crosswalk* tiles are identified by merging adjacent *crosswalk* cells into almost convex zones.
3. All *Pedestrian* cells adjacent to a *crosswalk* tile or a building tile are respectively merged into *crosswalk access* tiles or *building access* tiles. To obtain more consistent *crosswalk access* tiles and *building access* tiles, adjacent *pedestrian* cells are added to these tiles if it improves their convexity.
4. *pedestrian* tiles are identified by merging adjacent *pedestrian* cells that do not belong to a *crosswalk access* or to a *building access* tile. These cells are merged if they are not separated by a bottleneck and if the resulting zone is almost convex. Using the bottleneck information to separate tiles ensures that these tiles do not contain any punctual obstacle and thus guaranties to find a simple path through any of them.
5. In order to have information on the ways to cross the road, segments of roads that link facing *pedestrian* tiles are identified. First, vertices shared by at least two *pedestrian* tiles and a *road* cell are selected. Then, for each of these vertices, the shortest edge linking to such a vertex on the other

side of the road is identified as "separating edge". If no such edge exists, the shortest edge linking to any vertex on the other side of the road is identified as "separating edge". Finally, all adjacent *road* cells are merged if they are not separated by a "separating edge" and if the resulting tile is almost convex.

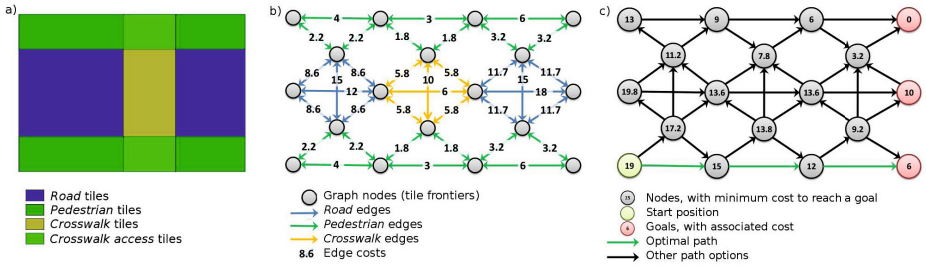
### 3.4 City Area Level Generation

High-level path planning decisions in urban environment mostly rely on the representation of the environment as a network of macroscopic areas such as streets separated by crossroads (Cf. figure 2.d). The city area level aims at providing such a high level representation and decomposes the environment into street, crossroads and pedestrian areas. Those areas are composed of adjacent tiles and are identified by the following process:

1. An area is created for each *road* and *crosswalk* tile. Each area is labelled "crossroad" if contiguous to strictly more than two *road* or *crosswalk* tiles, "path" if exactly contiguous to two *road* or *crosswalk* tiles or "dead end" if contiguous to only one *road* or *crosswalk* tile.
2. Due to the irregularities in the shapes of roads, some small dead-end areas may exist on road borders, generating improperly identified crossroads. To avoid this, each small dead-end area is merged with its contiguous area, transforming improperly identified crossroads into paths.
3. Areas labelled crossroads are merged with their adjacent areas containing a *crosswalk* tile and the resulting area is labelled *crossroad*. Moreover, *pedestrian* and *crosswalk access* tiles adjacent to this area are added if their addition improves the convexity of the area.
4. *pedestrian* and *building access* tiles that do not belong to a *crossroad* area are added to their adjacent area if the area is labelled path or dead end. As those tiles can be shared by several areas, all areas sharing at least one *pedestrian*, *building access* or *road* tile are merged and labelled path or dead end.
5. Path or dead end areas are merged with their adjacent *crossroad* area if this operation improves the *crossroad* area convexity. This generates the final *crossroad* areas.
6. Adjacent path and dead end areas are merged into almost convex zones and become the final *street* areas.
7. The remaining *pedestrian* and *building access* tiles are merge into almost convex zones that become the *pedestrian* areas.

### 3.5 Street Sections Level Generation

An intermediate level of path planning decisions features questions like "Is it better to cross the road now or will I have a better opportunity in the next street?". To answer such a question, a rough representation of *street* sections is provided by the *street* sections level (Cf. figure 2.c). This level partitions the



**Fig. 3.** (a) A simple zone example. (b) The path planning graph extracted from this zone. (c) The computed path options graph.

environment into *pedestrian* sections, *road* sections and *traversable road* sections (which include at least one crosswalk). For each area identified at the city area level, its compounding tiles are regrouped as follow:

1. All *pedestrian*, *building access* and *crosswalk access* tiles are merged into almost convex zones which are labelled *pedestrian* sections.
2. All *road* and *crosswalk* tiles adjacent to the same *pedestrian* sections are merged into a *traversable road* section if a *crosswalk* tile is included or into a *road* section otherwise.

The proposed method generates three semantically consistent hierarchical levels of decomposition of the outdoor navigable zones. The first level subdivides the environment into navigation tiles. At this level, a precise path can be computed. The second level regroupes the tiles into *street* sections, considering pedestrian navigation zones and the crossing opportunities between them. At this level, a coarse path can be computed. On a sidewalk or more generally in a pedestrian zone, the choice of the side used to bypass a small obstacle like a pole is not made. This gives more freedom during navigation. However, crossing opportunities are identified. Thus the pedestrian can choose on which sidewalk he should navigate and where he could cross a street. He can also anticipate the lack of crosswalk and choose to use one earlier along its path or not. The third level regroupes the *street* sections into city areas (streets, crossroads, pedestrian areas...). At this level, a path identifies the *street* sections and crossing along which the pedestrian should travel. The choice of where to cross a street or the side of the road on which the pedestrian should walk is not made. This enables to globally know where the pedestrian should go while delaying more precise choices. Those choices can be made during navigation when relevant information can be perceived.

## 4 Hierarchical Path Planning Process

In the previous section, we presented the generation of three semantically consistent hierarchical levels of decomposition of an urban environment. Each of these levels corresponds to a level of decision making. This enables the identification of a coarse path that can be refined when needed. In this section, we

describe how to take advantage of this decomposition to generate smarter navigation behaviour. First, we explain how to extract a hierarchical path planning graph from the environment partition. Then, we propose an algorithm that computes a set of path options through the environment. Those options can be used in real time to adapt the planned path to the perceived situation and make a detour if needed or necessary. Finally, we explain our hierarchical path planning process that takes advantage of the environment decomposition to generate smarter navigation behaviours by delaying some local decisions in order to adapt to unexpected events.

#### 4.1 Hierarchical Path Planning Graph

The hierarchical decomposition of the environment ensures that any zone belonging to a decomposition level is a union of zones identified in the lower level. It enables the creation of three hierarchically organized path planning graphs: the navigation tile graph, the street sections graph and the city areas graph. In these graphs, nodes represent borders between two zones and edges symbolize paths linking borders (Cf. figure 3.b). We define a border between two zones as the set of adjacent segments shared by two zones. The edges are labelled with the length of the path linking the borders and a typology of crossed zone. Edges in the navigation tiles and the street sections graphs are labelled as *pedestrian*, *road* or *crosswalk* depending on the nature of the crossed zone. In the city areas level, areas are composed of mixed cells typologies. We define high-levels zone natures: *streets*, *crossroads* and *pedestrian zones* to be associated with the corresponding edges. Each agent is given a preference factor over each typology of edge. The estimated cost of travelling an edge is computed by multiplying its length by the agents' preference factor associated with the typology of this edge. As every zone of a level is the exact union of zones from the lower level, the border between two zones is also the exact union of borders between zones of the lower level. This property is used to hierarchically organise the three planning graphs by linking a node in a given graph to the corresponding nodes in the lower level graph.

#### 4.2 Path Options Planning

When navigating in a city, people delay some decisions. For example, a pedestrian will decide on which side to bypass a city light only when reaching it, depending on other pedestrians and obstacles. Classical path planning techniques tend to select a unique path that should be followed by the pedestrian. This can lead, to situations in which multiple pedestrians struggle to pass to the same side of a pole while none pass to the other side. To avoid such issues and allow the agent to take a detour, we use an algorithm that identifies a set of path options linking a start position to one or multiple goals. This algorithm is based on the path planning graph. It computes an oriented graph (which is a subset of the original graph) with no cycle or dead end (Cf. figure 3.c). In this graph, the entry node is the start node and the exit nodes are the goals. This graph is computed on

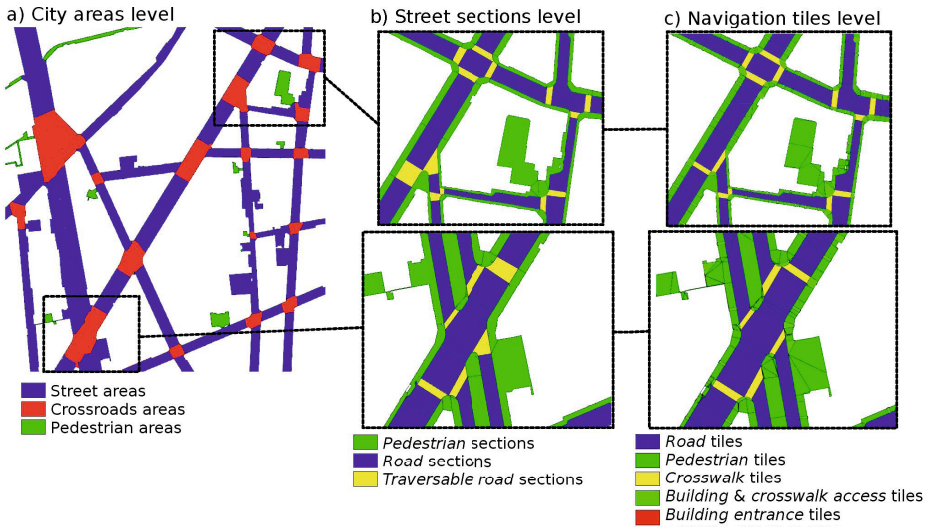
the basis of the Dijkstra shortest path tree. In this tree, edges are oriented from each node to the goals, thus identifying the shortest paths from every node to the goals. Undirected edges that link two branches of the shortest path tree are oriented in order to maximize the number of proposed detours while avoiding the creation of cycles. The resulting graph identifies the optimal path as well as possible detours. It also enables to compute the cost of a detour. Finally, for any node of the computed graph, using any of the exiting edges ensures to follow a path that reaches a goal. This process produces a path options graph in which all nodes are labelled with the cost associated with the minimum cost path reaching one of the goals.

### 4.3 Hierarchical Path Planning

Based on the hierarchical path planning graph, we propose a two steps path planning process. The first step identifies a high level path in terms of city areas. This path is then refined to identify paths options in terms of street sections. The second step occurs during navigation. The path previously identified is refined in terms of navigation tiles. Local path options are computed when the agent perceives the city area he is about to reach. The produced path options graph is then updated to reflect the impact of perceived dynamic events.

**High-Level Path Planning.** The high-level path planning process aims at choosing a global path in the environment. It is performed before the agent leaves his initial position. First, an optimal path is computed in the city area graph. This path is computed with an A\* algorithm [21] using the Euclidean distance to the goal as a heuristic. The computation of this path selects a list of city areas that should be crossed to reach the goal. Once this path is computed, it is refined by computing associated path options in the street sections graph. It is important to note that path options are computed only within the street sections that compose the selected city areas. This enhances the algorithm performance and only identifies path options that can be used along the high level path. Those path options are used during navigation, by the low level path planning, to evaluate the mid-term impact of a detour or to better choose a location where a street could be crossed for example.

**Low Level Path Planning.** Previously computed path options depict coarse paths that do not take dynamic information into account. Yet, some navigation decisions need to be taken reactively while being too complex to be treated by a reactive navigation process. Such decisions can be deciding on which side a pole should be bypassed or to cross a street to avoid a group of people for example. Every time the agent is about to reach the border of a new city area, local path options are computed within this area. First, all the navigation tiles borders that belongs to the goal area border are identified and stored as "local goals". Each local goal is given a cost equal to the one computed in the high-level path options graph. Then, a path options are computed in the navigation tiles graph associated with the reached street area. The obtained path options

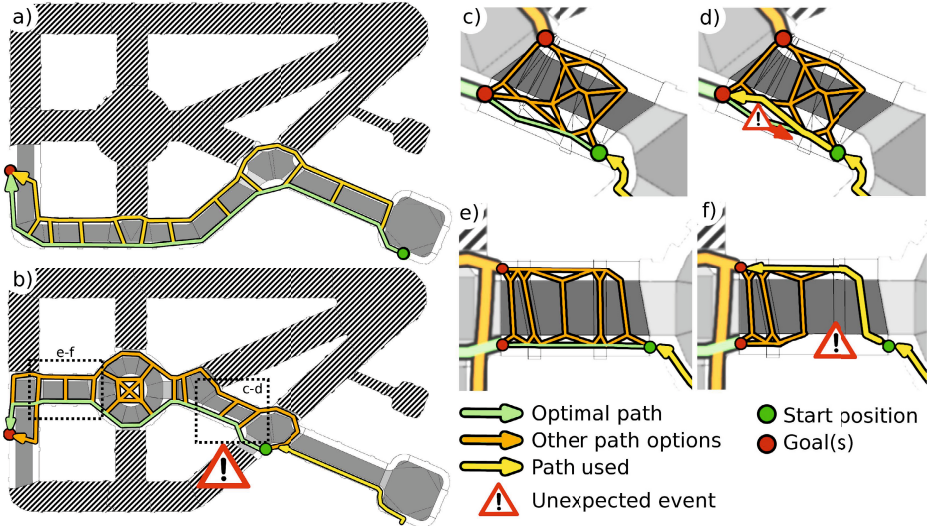


**Fig. 4.** Hierarchical decomposition of a real city map

graph describes a set of representative path options, which are labelled with their estimated impact on the final cost of the path. This way, local decisions can be made and their impact on the final path can be estimated. To reflect the impact of dynamic events happening in the city, costs of the path options graph are updated dynamically. These unexpected events are of three kinds:

1. A navigation tile is blocked by an obstacle. In this case, blocked nodes and edges of the path options graph are invalidated and costs associated with nodes of the graph are updated. If all path options are blocked, a new high-level path planning is performed from the actual position of the agent, after invalidating this area from the city areas graph.
2. An event (a high density of pedestrians or muddy ground, for example) can increase the cost associated with some path options. In this case, the costs associated with the nodes of the path options graph are updated.
3. An incoming interference with a dynamic isolated obstacle (a pedestrian for example) is detected. In this case, the path option leading to this interference is penalized. This can lead to favour another path option reflecting the adaptation to the situation.

The proposed hierarchical path planning process takes advantage of the hierarchical representation of the environment in order to delay some decisions until relevant information is available. Dynamic events can be taken into account while limiting the need for path re-planning. This process generates smart navigation behaviors for virtual agents as it proposes many options that can be used to better adapt the agent navigation to perceived situations and unexpected events.



**Fig. 5.** Demonstration of our path planning ability to react to unexpected events (for clarity reasons, only a representative sample of the computed path option is displayed)

## 5 Results

**Hierarchical Partition Process.** In order to test the effectiveness of our hierarchical partitioning method, we used a district of the city of Paris. We can see in figure 4.a that the obtained city areas efficiently identify crossroads in a large range of configurations. Figures 4.b and 4.c show that consistent street sections and navigation tiles are identified, even in complex configurations involving traffic islands and complex sidewalk shapes including punctual obstacles. Using such a complex real-life urban environment demonstrates the robustness of our method.

**Path Planning.** To demonstrate the impact of delaying local decision making and using path options to adapt to unexpected situations, an agent has to navigate in our example environment depicted figure 1. The figure 5.a shows the initially selected city areas and a subset of the computed high-level path options. When reaching the first crossroad, the agent perceives that the next street is obstructed. In this case, all the path options being invalidated, a new high level path is computed (Cf. figure 5.b). In the next street, we can see on figure 5.c that the optimal local path passes the city light to the left side. However, our pedestrian perceives another agent using the same path. This increases the cost of choosing this path and the second best option, passing the city light to the right is selected (Cf. figure 5.d). In the last street, the optimal local path reaches the lower cost local goal by staying on the same sidewalk. However, when a car parks on this sidewalk this path option is invalidated (Cf. figure 5.e). This results in the selection of a new optimal path that crosses the street in order to



reach the other sidewalk. With this path adaptation, the agent now navigates toward a local goal with a higher cost. The agent could have chosen to cross the street for a second time to reach the lower cost local goal. However, the crosswalk located in the next street area implies that after the previous detour, the current path is now optimal. These examples show how our algorithm takes advantage of the hierarchical decomposition of the environment to delay local decision making and compute multiple path options. The agent is able to react to unexpected events without always replanning a whole path but only when really necessary.

## 6 Conclusion

In this paper, we proposed a process which generates a semantically consistent hierarchical decomposition of an urban environment. It relies on identifying usual urban entities like streets, crossroads or sidewalks. Each of the three generated environment partitions is adapted to a different level of decision making. We also proposed a hierarchical path planning process that takes advantage of the good properties of the hierarchical decomposition. Unlike most of existing hierarchical path planning methods that focus on reducing path planning complexity, our method focuses on generating smarter navigation behaviours. Local decisions are delayed until relevant information is available and path options are planned through the environment. This allows to efficiently adapt to most unexpected events without needing a full path replanning. Such properties make this method a good solution for populating virtual cities with credibly behaving pedestrians. However, it is adapted to conventional city planning and could generate incoherent results if used on unexpected configurations. Future work on this subject should include some study on the interaction between multiple agents using this path planning process. It should also study how this method impacts on path planning complexity and how it can scale to very large environments and high numbers of pedestrians.

**Acknowledgments.** This work has been supported by the French National Research Agency through CONTINT program (iSpace&Time project, ANR-10-CORD-023).

## References

1. Botea, A., Müller, M., Schaeffer, J.: Near optimal hierarchical path-finding. *Journal of Game Development* 1(1), 7–28 (2004)
2. Brand, S., Bidarra, R.: Parallel ripple search—scalable and efficient pathfinding for multi-core architectures. In: Allbeck, J.M., Faloutsos, P. (eds.) *MIG 2011. LNCS*, vol. 7060, pp. 290–303. Springer, Heidelberg (2011)
3. Latombe, J.: *Robot motion planning*. Kluwer Academic Publishers, Boston (1991)
4. Arikan, O., Chenney, S., Forsyth, D.A.: Efficient multi-agent path planning. In: *Computer Animation and Simulation 2001*, pp. 151–162 (2001)

5. Bayazit, O.B., Lien, J.M., Amato, N.M.: Roadmap-based flocking for complex environments. In: *Computer Graphics and Applications*, pp. 104–113 (2002)
6. Geraerts, R., Overmars, M.H.: Creating high-quality roadmaps for motion planning in virtual environments. In: *IROS*, pp. 4355–4361 (2006)
7. Geraerts, R., Overmars, M.: The corridor map method: Real-time high-quality path planning. In: *ICRA*, pp. 1023–1028 (2007)
8. Shao, W., Terzopoulos, D.: Environmental modeling for autonomous virtual pedestrians. In: *Digital Human Modeling for Design and Engineering Symposium*, pp. 735–742 (2005)
9. Pettre, J., Laumond, J.P., Thalmann, D.: A navigation graph for real-time crowd animation on multilayered and uneven terrain. In: *V-Crowds*, pp. 81–89 (2006)
10. Kallmann, M., Bieri, H., Thalmann, D.: Fully dynamic constrained delaunay triangulations. In: *Geometric Modelling for Scientific Visualization*, pp. 241–257 (2003)
11. Mekni, M.: Hierarchical path planning for situated agents in informed virtual geographic environments. In: *SIMUTools*, pp. 66–91 (2010)
12. Lamarche, F., Donikian, S.: Crowd of virtual humans: A new approach for real time navigation in complex and structured environments. *CGF* 23(3), 509–518 (2004)
13. Kallmann, M.: Navigation queries from triangular meshes. In: Boulic, R., Chrysanthou, Y., Komura, T. (eds.) *MIG 2010*. LNCS, vol. 6459, pp. 230–241. Springer, Heidelberg (2010)
14. Demyen, D., Buro, M.: Efficient triangulation-based pathfinding. In: *AAAI*, pp. 942–947 (2006)
15. Thomas, G., Donikian, S.: Modeling virtual cities dedicated to behavioral animation. *CGF* 19, 71–80 (2000)
16. Farenc, N., Boulic, R., Thalmann, D.: An informed environment dedicated to the simulation of virtual humans in urban context. *Computer Graphics Forum (Proc. of Eurographics)* 18, 309–318 (1999)
17. Jaklin, N., Cook, A., Geraerts, R.: Real-time path planning in heterogeneous environments. *CAVW* 24(3-4), 285–295 (2013)
18. Paris, S., Mekni, M., Moulin, B.: Informed virtual geographic environments: An accurate topological approach. In: *Int. Conf. on Advanced Geographic Information Systems & Web Services*, pp. 1–6 (2009)
19. Jiang, H., Xu, W., Mao, T., Li, C., Xia, S., Wang, Z.: A semantic environment model for crowd simulation in multilayered complex environment. In: *VRST*, pp. 191–198 (2009)
20. Jorgensen, C.-J., Lamarche, F.: From geometry to spatial reasoning: Automatic structuring of 3d virtual environments. In: Allbeck, J.M., Faloutsos, P. (eds.) *MIG 2011*. LNCS, vol. 7060, pp. 353–364. Springer, Heidelberg (2011)
21. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2), 100–107 (1968)

# A New Way to Model Physics-Based Topology Transformations: Splitting MAT

Saman Kalantari, Annie Luciani, and Nicolas Castagné

ICA Laboratory, Grenoble INP, Grenoble, France

{Saman.Kalantari, Annie.Luciani, Nicolas.Castagne}@imag.fr

**Abstract.** This work focuses modelling and simulation of physics-based topological discontinuities in deformable objects, as they appear in fracturing, tearing or cracking phenomena. It introduces a new methodology, called “Splitting MAT”, which integrates into masses-interactions modelling. This methodology enables modelling topological discontinuities not on an interaction element, but directly on a mass element. The principles of the Splitting MAT method are presented and then illustrated through various models featuring topological transformations due to large physics based deformations. The properties of the method are analyzed: optimization of the modelling process of topological transformations, and fully stable memory and computational costs.

**Keywords:** Computer Animation, Simulation, Topological Discontinuities, Deformation, Physics-based Modelling, Masses-Interactions, Fracture.

## 1 Related Works and Positioning

After being used for a long time to simulate the behaviour of rigid and deformable objects, physical modelling started being used to deal with spatial discontinuities, such as those exhibited by fractures, tearing, cracking, and so on.

Several physics-based works targeting topological changes have been proposed in the contexts of various methods: finite elements method (FEM) [1][2][3][4], diffuse elements method (DEM) [5][6], masses-springs methods with mesh [7][8][9] and particular methods or more generally masses-interactions modelling [10][11][12][13][14]. In all these methods except the last one, the main difficulty is the fact that when fracture occurs, complex geometric and physical rearrangements have to be computed.

Fractures, tearing, cracking, fragmentations or even fusion and sticking, in both masses-springs methods with mesh and masses-interactions modelling have been modelled in two general ways. The first way, “structural modifications”, consists in changing the structure of the physical model [8][9] by adding or removing masses and springs or more generally interactions whenever a topological transformation happens. The second way, “parametric modifications” rest on obtaining topological transformations by just modifying the interactions’ physical parameters, such as elasticity and stiffness, without any change in the structure of the physical model [13].

However, in this case, the topological changes happen between the masses, onto the physical interactions.

The use of “Parametric modification” to target topological modification was indeed introduced at the very beginning of this type of models in Computer Animation. Exemplary works are: thermal particles in [11], fractures modelled by a Lenard-Jones interaction function types [13], etc. In all these cases, the topological changes occur between the masses, onto the physical interactions. Thus, depending on the associated geometrical representations, although there is not formally addition nor suppression of springs nor dampers, the geometrical representation may create a visual artefact of matter removal at the time of the rupture.

We propose here, in the context of the masses-interactions approaches, a methodology to support topological discontinuities, not on a physical constraint, but rather on a mass element (whatever its type is). We have called this process “Splitting MAT”, the term MAT being the name of material elements in the modelling and simulation formalism we employed for experimental works, CORDIS-ANIMA [12]. However, the proposed principles can easily integrate into any other approach based on masses and interactions. Five interests of the proposed method are (1) to avoid the effect of matter removal when topological changes occur; (2) to avoid post-duplication of masses and creation/suppression of interactions at the time of the topological modification; (3) to avoid physical parameters values recalculations all along the reconfigurations; (4) to provide a fully stable memory and computational cost, no matter the emerging effects are; and (5) to support a new modelling methodology of topological modifications, that starts by considering the fully-split state, instead of the fully-tied state.

In the following, we first present the concept of Splitting MAT, then we describe models of highly deformable surfaces, that physically tear in various non-predetermined ways, including effects such as propagations of tears within the material, or apparition of ravelling. Finally, we analyze the proposed Splitting MAT method in terms of design optimization.

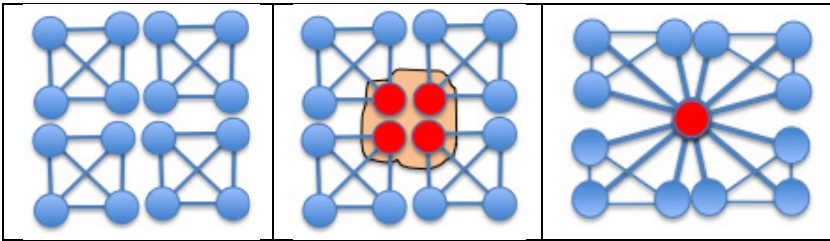
## 2 The “Splitting MAT” Method

### 2.1 Mass-Interaction Modelling

The Splitting MAT system integrates into masses-interactions modelling approaches. To support our experimental work, we employed the CORDIS-ANIMA formalism [12]. In masses-interactions modelling, building a model consists in assembling material elements, and interaction functions. Material elements are named MAT in CORDIS-ANIMA, giving its name to the Splitting MAT method. Material elements can of course be simple masses, but also exhibit more behaviours than inertial behaviour. Interactions can be chosen among a large library of interaction profiles: linear or non-linear elasticity, linear or non-linear viscosity, cohesive Van der Waals interactions, plastic interactions, user-defined or state-graph-defined interaction profiles, etc. The masses and interactions assembly constitutes a network, on which masses are freely interconnected by interactions. The networked structure is not meshed-based, and its building is not based on a discretization of any spatial continuum.

## 2.2 The Concept of Splitting MAT

Usual modelling processes of cracking and fracturing start from the fully-tied state, which is about to be split, then consider the necessary steps to manage the breaking. The splitting MAT corresponds to a new philosophy of topological rupture modelling. Indeed, modelling consists in considering first the fully-split state, the smaller possible grains of matter, from which masses are then united to constitute larger pieces of material. From the smaller possible grains, subsets of masses are selected to constitute what we call “*Mass Unions*” (Figure 1). Before any splitting occurs, the system will guarantee that, for each Mass Union, all the masses within each Union have the same exact movement, meaning that they are superimposed at each instant, and that they act as a single mass.



**Fig. 1.** A Mass Union with four masses. Left: four grains, each made of 4 masses and 6 interactions. Middle: four masses of each grain have been gathered into a Mass Union. Right: the gathered masses will indeed remain superimposed at each instant, before any splitting occurs.

In order for each mass of a Mass Union to obey to the same behaviour, the necessary and sufficient conditions are: (1) they be set with the same initial conditions (position and speed), (2) they have the same values of inertia, and (3) they receive identical forces. Provided these three conditions are met, and before any split, the system effectively behaves as if all the masses of a Union were only one single mass, to which all the interactions tied to any mass of the Union would be connected. Figure 1 illustrates this notion: from left to central figure, the formal union process, and on the right, its spatial mapping.

During the simulation, a splitting will occur when certain conditions are gathered on all or a part of the system state variables. The considered state variables can be all the positions and speeds of the masses, as well as the forces applied onto them, and the forces produced by each interaction function.

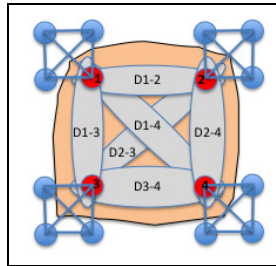
As a result of splitting, a subset of masses of the considered Union leaves the Union to constitute new Unions, each with fewer masses than the original, and with lower inertia values, to preserve the principle of mass conservation. During the simulation, as masses gradually leave a Union, the Unions are modified and new, smaller, Unions progressively appear. The “fully-split” state of the material, in which all the Unions are composed of a single mass, may or may not occur, and intermediary states are possible. This results in the non-predetermined production of holes, tears,

fractures, material pieces (e.g. subset of connected masses-interactions networks) of various sizes, etc.

### 2.3 The Notion of Duplet, as a Mean to Build Mass Unions

The State Change conditions and the resulting changes of the model constitute a state automaton. The design and control of a general state machine to manage the evolution of the Mass Unions would quickly become very complex. We propose the notion of “*Duplet*” to simplify the process while not reducing the possibilities of structural modification too drastically.

A duplet is an entity  $D_{ij} \{M_i, M_j, T_{ij}\}$  composed of two masses  $M_i$  and  $M_j$ , and a transition condition  $T_{ij}$ . The transition condition defines the state of the duplet, which is either *active*, i.e. the two masses are tied by the duplet, or *inactive*, i.e. the masses are not tied to each other. Once inactivated, a duplet will remain inactive till the end of the simulation.



**Fig. 2.** A Splitting MAT with 4 masses and 6 duplets. If 6 duplets are active, the 4 masses constitute a single Mass Union.

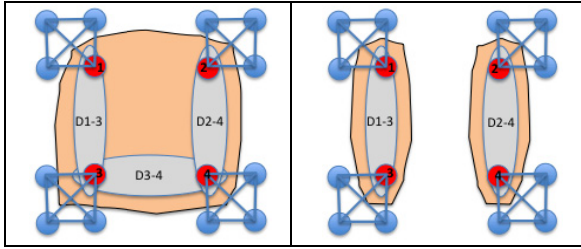
Ensuing from the notion of duplet, a Mass Union is defined as the set of masses tied by active duplets. Hence, all the masses of a Union are those that can be reached by going from one mass to another along an active duplet. Considering the undirected graph in which vertices are masses, and edges are active duplets, this means that a Union is formally equivalent to a connected sub-graphs. As an example, a Union of four masses can be built using 6 duplets (Figure 2).

### 2.4 Management of the State Automaton of the Splitting MAT

The state of a Splitting MAT (that is: the set of its Unions) is entirely determined by the state of all the duplets, active or inactive. For a Splitting MAT with  $n$  masses, one can create a maximum of  $Q = n * (n-1)/2$  duplets. Similarly, for a Splitting MAT containing  $Q$  duplets, we can consider  $2^Q$  intermediary duplets’ states.

All the possible  $2^Q$  states of the  $Q$  duplets of a Splitting MAT are represented by a minimal data structure made of an array of  $Q$  bits, where the bit  $i$  is set if the duplet  $i$  is active. Such a bit array is a simple integer with a value between 0 and  $2^Q-1$ . For instance, for a Splitting MAT composed of  $N$  masses and  $Q$  duplets:

- If the integer representing the state of the duplets is equal to  $2^Q - 1$ , then all the duplets are active, and there is a single Mass Union. In other terms, all the masses of the Union are grouped, meaning that no splitting have occurred yet.
- If the integer representing the state of the duplets is equal to  $0$ , then all the duplets are inactive, and there are  $N$  Unions, each composed of a single mass. In other terms, all the masses are free (fully-split state of the Splitting MAT).



**Fig. 3.** Two possible successive states of the unions of a Splitting MAT made of 4 masses and 6 duplets. On the left, the structure presented in Figure 3 in which the duplets  $\{D_{1-2}, D_{2-3}$  and  $D_{1-4}\}$  are supposed to be inactive and are not drawn on the figure. However, the Union remains the same. Right: if  $D_{3-4}$  duplet now deactivates, two Unions appear. Note that other breaking could occur, depending on which duplets are inactivated first.

Let us go back to the example of a Splitting MAT composed of four masses and six duplets (Figure 3). The duplets therefore able to take  $2^6$  (64) different states, and there can be 15 states for the Unions, with 1, 2, 3 or 4 masses. In the beginning, in the non-split state (number 63), all the duplets of the Splitting MAT are active. Deactivating 3 duplets, whatever they are, does not change the Unions (Figure 3 - left), since, even after, all the masses are still reachable by the three remaining active duplets. Conversely, the deactivation of a fourth duplet, e.g.  $D_{3-4}$  on Figure 4, results in the apparition of two Unions (Figure 3 - right).

We now consider how to manage the state automaton and the transitions. Computing dynamically the Mass Unions whenever a duplets is inactivated, by running through the graph made of masses and active duplets would be computationally costly. The following mechanism faces this issue.

Considering that Unions are fully determined by the state of the duplets, we pre-calculate a table, named “States Table of the Mass Unions”, that stores the correspondences between each possible state of the duplets and the corresponding Unions. In this table, lines are indexed by the possible states of the duplets, from  $0$  to  $2^Q - 1$ , and each line indicates which mass belongs to which Union, and the inertia value shared by the masses in each Union. With such a pre-computed States Table, when a splitting condition occurs for a duplet, there is no need to run through the graph made of masses and active duplets, nor to recomputed parameter values: change from one state to another is immediate.

Noticeably, the State Table is identical for all the Splitting MAT sharing the same duplets organization. Consequently, the memory cost of the pre-computed State Table is negligible, even for large models with many Splitting MAT.

## 2.5 Properties of the Duplets-Based Automaton for the Modelling Process

The Duplet concept presents a number of interests to manage the splitting of Splitting MAT, as compared to an explicit final state automaton. Among these, importantly, the notion of Duplet is a mean, for the modelling user, to *build* the process of splitting. Using duplets, a user is able to freely build his (her) own organization in order to obtain the wished frailness of the material. He (she) has a mean to manipulate the ease with which the splitting occurs, as he (she) wants.

Indeed, the splitting will be less difficult when the number of duplets is low. If  $n$  is the number of masses of a Union, the minimal number of duplets to obtain a fully-tied state is  $(n-1)$ . With this minimal number, whenever the condition of a single duplet is reached, the Mass Union will split into two other Unions. When the number of duplets is greater than this minimum, then the splitting of the Union into several other Unions requires more than one duplet's transition to occur. Hence, choosing the number of duplets implicitly manage the difficulty for the material to break.

## 3 Modelling Experiments

The Splitting MAT method allows the user to model fractures and tearing as he (she) sees fit, through three main modelling phases:

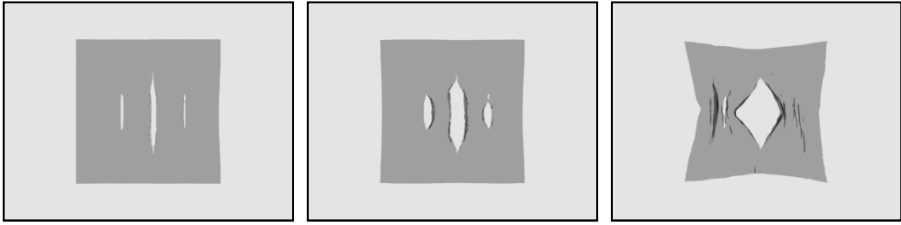
1. Creation of basic grains of matter. This corresponds to the fully-split state of the model – which late, during simulation, will or will not be reached during simulation, depending of the occurring phenomena.
2. Uniting the grains by creating duplets on pairs of masses from different grains. Duplets definition results in an implicit specification of splitting state automaton. Once all duplets have been created, the system can automatically determine the needed pre-computed States Tables.

The three models described hereafter have been modelled with the Splitting MAT concept and system. They model surfaces of variable deformability that exhibit various types of topological changes (multiple fractures, holes, apparition and propagation of multiple tears, ravelling on the edges of the slits, and so on), and their evolution during the simulation.

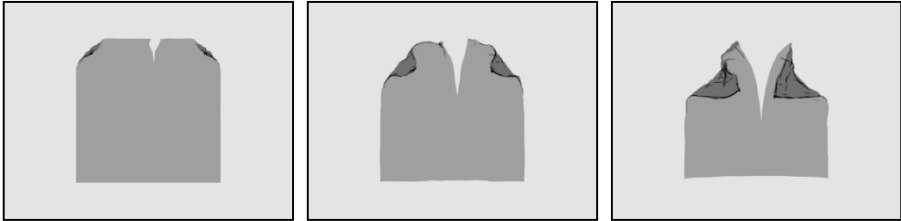


Fig. 4. Model of complete tearing of a visco-elastic surface





**Fig. 5.** Modelling cracks and holes in a visco-elastic surface



**Fig. 6.** Modelling the propagation of tears in a highly deformable visco-elastic surface

These models are high-resolution surfaces composed of 96100 masses and 144150 visco-elastic interactions. They were assembled, according to the schema given in Figure 2 and 3, by first building grains made of 4 masses, then gathering the masses of these grains into Splitting MAT, by instantiating duplets. Hence, all the inner Splitting MAT are made of 4 masses and 6 duplets. In all the models, each sensor calculates a distance between two masses, and inactivates its associated duplet if this distance is higher than a chosen threshold. The first model (Figure 4) results in a complete tearing of a very deformable surface. Two points, at the bottom and top left corners, attach the surface. It is then stretched from the right-side corners. Two tears appear and propagate, finally tearing completely the surface into two deformable pieces that evolve independently. The second model (Figure 5) is a deformable surface in which several cracks and holes appear and then propagate. In the third model (Figure 6), the surface is attached by two points, on bottom left and bottom right. The two other corners are the stretched. A tear appears and propagates over time.

## 4 Conclusions

In this article, we have studied the problems regarding modelling of topological discontinuities in deformable objects, and phenomena such as fractures, breaks, tears and cracks, that are naturally very present in our everyday surroundings. Many research have shown interest in modelling and simulating fracture-type phenomena. Several methods have been proposed, such as: mass-spring networks, finite element methods, meshless methods and masses-interactions methods. A common question in these methods, discussed in the beginning of this article, is where the topological discontinuities can appear and propagate. We have studied this problem through several approaches of physical modelling of topological discontinuities. We have then proposed

our approach, based on the masses-interactions formalism, which allows modelling a range of deformable models presenting topological discontinuities. Our approach, called Splitting MAT, enables modelling topological discontinuities on physical mass elements rather than on physical constraint element. This avoids the visual artefact of matter removal when the discontinuities occur between masses or points, and suppresses the need for refinement along the borders of the discontinuity. Our method has been developed under the form of a fully modular modelling device, intended for users of modelling tools.

**Acknowledgements.** This research was supported by the French National Research Agency through the cooperative project DYNAMÉ – ANR-2009-CORD-007, the French Ministry of Culture and Grenoble Institute of Technology.

## References

1. O'Brien, J.F., Bargeil, A.W., Hodgins, J.K.: Graphical Modelling and Animation of Ductile Fracture. *ACM Transactions Graphics* 21(3), 291–294 (2002)
2. Molino, N., Bao, Z., Fedkiw, R.: A Virtual Node Algorithm for Changing Mesh Topology During Simulation. In: *Proc. of SIGGRAPH 2004, ACM TOG*, vol. 23, pp. 385–392 (2004)
3. Bao, Z., Hong, J.M., Teran, J., Fedkiw, R.: Fracturing Rigid Materials. *IEEE Transactions on Visualization and Computer Graphics* 13, 370–378 (2007)
4. Glondu, L., Marchal, M., Dumont, G.: Real-Time Simulation of Brittle Fracture using Modal Analysis. *IEEE Transactions on Visualization and Computer Graphics* 19(2), 201–209 (2013)
5. Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., Alexa, M.: Point Based Animation of Elastic Plastic and Melting objects. In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2004*, pp. 141–151 (2004)
6. Pauly, M., Keiser, R., Adams, B., Dutré, P., Gross, M., Guibas, L.J.: Meshless animation of fracturing solids. *ACM Transactions Graphics* 24(3), 957–964 (2005)
7. Ganovelli, F., Cignoni, P., Montani, C., Scopigno, R.: A multiresolution model for soft objects supporting interactive cuts and lacerations. *Computer Graphics Forum* 19(3), 271–281 (2000)
8. Meseure, P., Darles, E., Skapin, X.: Topology-based Physical Simulation. In: *Proc. of VRIPHYS 2010, Copenhagen, DK*, pp. 1–10 (November 2010)
9. Fléchon, E., Zara, F., Damiand, G., Jaillet, F.: A generic topological framework for physical simulation. In: *21st International Conference on Computer Graphics, Visualization and Computer Vision*, pp. 104–113 (2013)
10. Miller, G., Pearce, A.: Globular Dynamics: a Connected Particle System for Animating Viscous Fluids. *Computers & Graphics* 23(3), 169–178 (1989)
11. Tonnesen, D.L.: Modelling Liquids and Solids using Thermal Particles. In: *Graphics Interface 1991*, pp. 255–262 (1991)
12. Luciani, A., Jimenez, S., Florens, J.L., Cadoz, C., Raoult, O.: Computational physics: A modeler simulator for animated physical objects. In: *Proc. of the Eurographics 1991* (1991)
13. Luciani, A., Godard, A.: Simulation of Physical Object Construction Featuring Irreversible State Changes. In: *Proceedings of WSCG*, pp. 321–330 (1997)
14. Greenspan, D.: *Particle Modeling*. Birkhauser (1997)

# 3D Artistic Face Transformation with Identity Preservation

Tanasai Sucontphunt

Graduate School of Applied Statistics  
National Institute of Development Administration  
Thailand

**Abstract.** This paper describes a 3D face modeling technique that automatically transforms a normal human face to an artistic face. Starting from any 3D human face, the selected 3D artistic-style is gradually added to the face to transform it toward the plausible artistic face. In the transformation process, the human facial identity will remain recognizable while the artistic-style is slowly visible. The main contribution of this work is in the identity preservation algorithm that can prevent the human identity to be altered by the added artistic-style. Our numerical evaluations illustrate that our technique can significantly maintain the facial identity in the transformation process better than other transformation techniques.

**Keywords:** 3D Face Modeling, 3D Face Transformation, Artistic Faces, Cartoon Faces, Geometric Deformation.

## 1 Introduction

Transforming a 3D face to a target artistic face requires highly trained skills from a 3D artist. Facial transformation special effect is used in many movies and games such as werewolf, vampire, and monster transformations. However, there is no automatic tool developed to handle this time-consuming job. For example, in the AVATAR movie, each 3D face model of the Na'vi character is crafted manually with a 3D modeling tool by highly skilled 3D artists. Starting from a 3D face model of an actor/actress, the artists manually craft the face surface toward the target artistic-style bit-by-bit so that the main identity of the actor/actress is not completely changed. Current works on face modeling algorithms that focus on manipulating, deforming, or exaggerating an initial face model can be potentially used for this purpose. Nonetheless, these works typically require users to manually fine-tune some parameters or require a collection of artist-drawn examples to generate statistical models. Many 3D face avatar modeling tools are typically asking users to manually compose facial components from a provided templates (e.g., Second Life) or to adjust a morphing of a human face and a monster face together (e.g., [www.evolver.com](http://www.evolver.com)). However, to the best of our knowledge, all these techniques cannot retain a human identity while transforming the face. Furthermore, they do not provide a meaningful control to tailor the level of artistic-style over the human face. This paper presents a technique for 3D face modeling through the use of facial recognition and 3D deformation techniques. The facial recognition is used to capture the human identity as well as to protect it from mixing with other identities.

The 3D deformation is employed to add the proper artistic-style to the human face on the specific areas that have minimal intervention in the facial identity.

Figure 1 illustrates the four main steps in our technique: (i) Starting from a 3D human face (representing *human identity* or further abbreviated as the **HI-face**) and a 3D artistic-style face (representing *artistic-style* or further abbreviated as the **AS-face**), both the HI-face and AS-face are projected to reduced Principal Component Analysis (PCA) representations in order to identify their identities. (ii) The artistic-style is then extracted from the AS-face with our artistic-style decomposition algorithm. At the same time, the HI-face is used as a guideline to preserve the face identity. (iii) The extracted artistic-style is added to the HI-face to gradually transform the HI-face to its artistic face by a single controlled parameter. To prevent change of identity, the artistic-style is only added to the parts of the HI-face where they are not preserved (not in the red blocks) as the main human identity. (iv) A resultant 3D artistic face model (representing *artistic-style over human identity* or further abbreviated as the **AH-face**) is automatically constructed by our 3D surface and texture synthesis algorithms while the face identity is preserved.

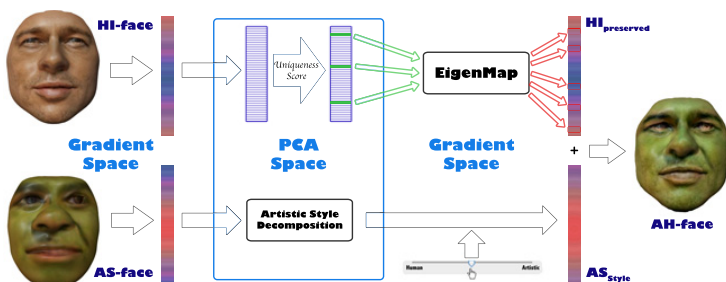


Fig. 1. The schematic overview of our approach

## 2 Related Work

In the past, researchers developed several 3D face modeling techniques in order to automatically create natural facial shapes according to controlled parameters especially for facial animation purpose. This section only focuses mainly on the works of the 3D human face and artistic face modeling.

**3D Human Face Modeling:** Traditional blendshape or geometric deformation approaches allow users to generate individual-specific models by moving weight control sliders. Essentially, these approaches simultaneously move and edit a group of relevant vertices, which save significant manual efforts for users. Data-driven 3D face modeling and editing techniques exploit the intrinsic correlations among different facial regions by learning statistical models from a pre-collected face dataset. For example, Blanz and Vetter [1] build a 3D morphable face model by constructing PCA spaces from the 3D geometry and texture from a scanned 3D face dataset. Some tools attempt to improve the efficiency of producing muscle actuation based blend shape animations [13]. However, these approaches generally require considerable manual efforts to create a set of blendshape targets, even for skilled animators. A number of statistic

learning techniques [4,6] were proposed to address this issue. Recently, many intuitive tools [10,16,12] have offered interfaces that can automatically adjust the co-related facial regions when posing a face. However, these methods are exclusively focused on modeling normal human faces, and none of them can be used to transform a human face to a 3D artistic face while preserving the human identity.

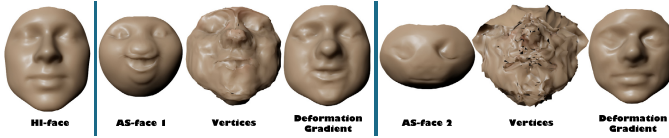
**Artistic Face Modeling:** For an artistic shape modeling, several works focus on developing a deformation algorithm to morph the 3D meshes to a target shape. For example, FiberMesh system [9] employs Laplacian deformation technique to inflate 2D silhouette strokes to become a 3D object. Recent example-based caricature generation techniques statistically learn the drawing styles from artist-drawn caricature examples or extract crucial information from human face images. For example, Liu *et al.* [7] use examples from artists to generate 2D caricatures from photographs using eigen-space mapping. Hsu and Jain [3] use interactive snakes algorithm to detect facial components and then generate a 2D caricature by scaling their difference from the average face. The variations of the caricatures produced by these approaches are often limited to the provided examples. Recently, Sucontphunt [14] has developed an artistic character creation tool. However, the work only focuses on creating a variation of a character while maintaining its artistic-style, and it cannot preserve a human identity or create a natural facial transformation. Mathematically, the work cannot preserve a human identity better than a traditional linear blending approach in order to keep its artistic-style intact.

### 3 Data Processing Step

The 3D human faces are first processed in an offline manner to construct proper representation to be used in the real-time application. This work relies on a pre-collected 3D human face dataset [17] consisting of 100 subjects (56 females and 44 males) with diversified ethnicities. Each entry in the dataset consists of a 3D face mesh and its corresponding texture. We select the topology of 6,130 vertices and 11,934 triangles as a prototype face. Then, we register all the faces with the prototype face using the deformation transferring algorithm [15]. As a result, all the registered faces have the same topology as the prototype face.

In a PCA-based face recognition, the similarities between two faces are measured by their Euclidean distance of the faces in a concise and meaningful PCA subspace. Thus, to statistically exploit knowledge of human identities from the dataset, a PCA representation (referred by  $\phi$ ), which is a column matrix of eigen-vectors, is constructed in this offline step. Since there are three major identity spaces used in this work, each PCA representation is constructed separately as follows:

**Facial Geometric Representation.** The geometric identity of each 3D face is encoded as deformation gradients [15] which are  $3 \times 3$  affine transformation matrices between the face and an average human face. The deformation gradients are used rather than raw vertex positions because they produce smoother surfaces in compositing the faces together in grain level as shown in Figure 2. The deformation gradient can produce smoother surfaces because they balance the transformations of the shared vertices among meshes. Then, the PCA representation of the geometry,  $\phi_{Geo}$ , is constructed using a Singular



**Fig. 2.** Comparing the transformation results of using vertex positions and deformation gradients as feature vectors. Each result is created from transforming the HI-face on the left-most to the AS-face on their left.

Value Decomposition (SVD) from the covariance matrix of feature vectors forming by these affine transformation matrices. Each feature vector is constructed by putting all matrix entries into a column vector as shown by each column in Eq. 1 where  $A_{1-3}$  represent each column of the affine transformation matrix ( $A$ ),  $t$  is the total number of triangles, and  $f$  is the total number of human faces in the dataset. After SVD,  $\phi_{Geo}$  is the eigen-vectors, where  $v$  contains eigenvalues. The  $\phi_{Geo}$  is the size of  $(9t) \times (f - 1)$ .

$$G = \begin{bmatrix} A_{1_{11}} & \dots & \dots & \dots & A_{1_{1f}} \\ \dots & \dots & \dots & \dots & \dots \\ A_{3_{t1}} & \dots & \dots & \dots & A_{3_{tf}} \end{bmatrix}; G^T G = \phi_{Geo} \cdot v \cdot \phi_{Geo}^* \quad (1)$$

**Facial Texture Representation.** Similarly, we also construct a face texture PCA representation,  $\phi_{Tex}$ , based on the textures of all the faces in the dataset using the same technique as in the geometry part. Again, to prevent the obvious color artifacts from the grain-level blending process similar to the geometry problem, the texture is represented by an image gradient from an average human face instead of raw pixel values. The image gradient is calculated by applying  $3 \times 3$  window of Laplacian operators on each pixel (for each color channel (R,G,B) separately). In this process, a feature vector is created by concatenating a gradient of each color channel of each pixel from the face texture of the size of  $w \times h$  together, i.e.  $(r_{11}, g_{11}, b_{11}, r_{12}, g_{12}, b_{12}, \dots, r_{wh}, g_{wh}, b_{wh})^T$ . After SVD, the size of  $\phi_{Tex}$  is  $(3 \cdot w \cdot h) \times (f - 1)$ . In this work, each face texture size is  $256 \times 256$  square pixels.

**Facial 2D-Image Representation.** In this work, all 3D human faces in the dataset are rendered with their textures by Lambert shading from their frontal view to create 2D images of  $500 \times 500$  square pixels of the faces. Then, the PCA representation for these 2D images,  $\phi_{Image}$ , are constructed the same way as the texture representation. A feature vector in this representation is created from R, G, B colors of each pixel from each 2D image. This  $\phi_{Image}$  will only be used for the evaluation in Section 6.

## 4 Identity Preservation Algorithm

Our goal is to transform a 3D human face by gradually adding an artistic-style without changing its identity. The main issue of this goal is in developing a technique to identify which parts of the face are the human identity to be maintained and which parts are the artistic-style to be seamlessly added. Thus, our Identity Preservation (IP) algorithm consists of two main functions which are 1) *IdentitySelection*: to select which parts of

the face are human identity to be preserved (Section 4.1), and 2) *Transformation*: to transform the face to its artistic-style by skipping its preserved identity (Section 4.2) as shown in Algorithm 1.. Since a human face in this work is composed of its geometry and texture, to make an algorithm concise, they are both called “a face” because they use the same algorithm to achieve the goal but in different spaces. The face identity in this work also means its gradients (in both spaces) rather than its primitive vertex positions and image pixels as mentioned in Section 3.

---

**Algorithm 1.** Identity Preservation Algorithm
 

---

**Input:** HI-face, AS-face,  $\phi$ ,  $v$ .

**Output:** AH-face.

- 1:  $HI_{preserved} = \text{IdentitySelection}(\text{HI-face}, \phi, v)$ ;
  - 2:  $\text{AH-face} = \text{Transformation}(\text{HI-face}, \text{AS-face}, HI_{preserved})$ ;
- 

#### 4.1 Identity Selection

The simplest way to identify which parts of the face represent its identity is to compare it against an average face. The more discrepancies between the parts on original face and those of the average face, the more unique (implying more prominent facial identity) of those parts are [2]. Thus, for geometric part, we can compare the vertex positions of the original face to those of an average face. However, this local comparison yields unnatural facial surface because it does not consider any relationships between nearby vertices. This problem can be alleviated by comparing its transformation matrices using deformation gradient framework, as mentioned in Section 3. In fact, the deformation gradients help indicate exactly which parts of the original face have high discrepancies from an average face by judging from the values in the transformation matrices. However, the deformation gradient alone cannot tell which parts are more unique among human face shapes in general. The reason is that it does not consider the variations of other human face shapes. Some parts of the human face, though considered as main contributing identity of the face, are transformed very little comparing to other parts of the face. For example, the eye parts contain marginal transformations but human uses these parts to distinguish between different persons. Also, some parts of the face are always required large transformations, e.g, facial outline. The texture counterpart also has the similar issues but in a color space. If we mainly preserved high gradient values of a face, the artistic-style will only be added to those small transformations without considering the real facial uniqueness.

To take a statistic of human faces into consideration, our Identity Selection algorithm makes use of the way an eigenface identifying unique parts of a face. The eigenface recognizes faces by measuring them in a concise and meaningful PCA space. The face in PCA space is represented by a linear combination of eigen-vectors (which are a concise form of face shapes), i.e.  $Face_{PCA} = w_1.EV_1 + w_2.EV_2 + \dots + w_n.EV_n$  where  $w$  is a weight (a.k.a. a PCA coefficient) and  $EV$  is an eigen-vector in  $\phi$ . Having similar weights means the faces are alike. The weights basically imply which eigen-vectors are important in forming this particular face. The eigen-value of each eigen-vector also defines its variance which explains a variation of this shape. The weights and their

eigen-values can then be used to determine which eigen-vectors create unique shapes for this face. The uniqueness of an eigen-vector can be inferred by a face likelihood which is  $\frac{1}{(2\pi)^{\frac{M}{2}}} \exp\{-\frac{1}{2} \sum_{i=1}^n \frac{w_i^2}{v_i}\}$  where  $i$  is the number of eigen-vector and  $v$  contains eigen-values. To simplify a calculation, a uniqueness score of  $EV_i$  can be calculated by  $\frac{w_i}{v_i}$ . Thus, we use this uniqueness score as a criterion to select which eigen-vectors are unique as shown at lines 1-7 in Algorithm 2.. Here, a threshold  $T$  is obtained from an automatic threshold selection algorithm. In this work, an isodata clustering algorithm [8] is used to rapidly classify which eigen-vectors cover necessary unique patterns. In this algorithm, the uniqueness scores are clustered so that the scores stand out to be an outlier forming their cluster centroid. This centroid is then used to calculate a threshold to separate typical scores from the outstanding high scores. Thus, the eigen-vectors that have uniqueness scores higher than  $T$  are then added to a selected eigen-vector list, *SelectedEVList*. The selected eigen-vectors from *SelectedEVList* are in PCA space but the transformation process is performed in a gradient space. Thus, each eigen-vector will be reconstructed to its original space. Specifically, the original spaces of  $\phi_{Geo}$  and  $\phi_{Tex}$  are the deformation gradient and the image gradient respectively. As shown in Figure 1, each reconstructed eigen-vector is called *EigenMap* in this work and calculated by  $EigenMap = EV + \mu$  where  $\mu$  is its mean gradient. These *EigenMaps* represent transformation patterns of the face statistically rather than the real transformations of the face. In this *EigenMap*, the high values mean they are significant factors in creating this probabilistically unique pattern. Thus, we can now consider these values in identifying which gradients are worth to be preserved. To achieve this, the values in each *EigenMap* are checked against a threshold obtained from the isodata clustering algorithm [8]. If the values are above the threshold, their indices will be added to the set of  $HI_{preserved}$  as shown at lines 9-16 in Algorithm 2..

---

### Algorithm 2. Identity Selection Algorithm

---

**Input:** HI-face,  $\phi$ ,  $v$ .

**Output:**  $HI_{preserved}$ .

```

1: HI-facePCA = HI-face *  $\phi$ ;
2: HI-faceScore = UniquenessScore(HI-facePCA,  $v$ );
3: for  $i = 1$  to  $sizeofvector$ (HI-facePCA) do
4:   if HI-faceScore( $i$ )  $\geq$  T(HI-faceScore) then
5:     Add  $i$  to SelectedEVList;
6:   end if
7: end for
8:
9: for all EV in SelectedEVList do
10:  EigenMap = Reconstruction(EV)
11:  for  $j = 1$  to  $sizeofvector$ (EigenMap) do
12:    if EigenMap( $j$ )  $\geq$  T(EigenMap) then
13:      Add  $j$  to  $HI_{preserved}$ ;
14:    end if
15:  end for
16: end for

```

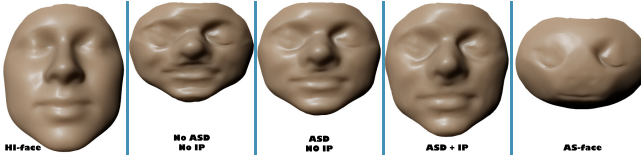
---



## 4.2 Transformation

Now, we have a list of features considered as preserved identity stored in  $HI_{preserved}$  as illustrated in the red blocks in Figure 1. The next step is to add gradients of AS-face to HI-face by avoiding any interference with the identified features in the list. However, the identity of the HI-face will still be disturbed by AS-face from its hidden human identity as illustrated in Figure 3. This is because there is a new identity emerging from the combination of HI-face and added AS-face that outshines the main identity. To remove the possible racing identity, the AS-face is first disassembled to search for its hidden human identity. The main assumption for this disassembling is that each AS-face is composed of an artistic-style ( $AS_{Style}$ ) and its base human identity ( $AS_{Ident}$ ), i.e.  $AS\text{-face} = AS_{Style} + AS_{Ident}$ . The artistic-style adds the artistic aspects to the face while the base human identity helps differentiate the faces with same artistic-style. Thus, removing its base identity from AS-face can prevent having its base identity mixed with the main identity.

**Artistic-Style Decomposition.** The PCA representation of  $\phi$  is used again to identify its base identity. The trick behind this technique is that the PCA representation is constructed from the dataset of human faces which gives a normal range of variations of human faces. Therefore, by performing the PCA projection of the AS-face, its base identity is captured by the base components. Then, the artistic-style can be calculated from the residue after the reconstruction of the projection by Eq. 2. To capture only major human face shapes, only the first 10 base components of  $\phi$  are used for the PCA projection. Figure 4 shows the  $AS_{Style}$  and the  $AS_{Ident}$  of the AS-face example. The  $AS_{Style}$  is now identity-free and safe to be added to the HI-face with minimum racing identity. Note that the  $AS_{Style}$  is the residue in gradient space after PCA reconstruction, it cannot be represented or manipulated in PCA space.



**Fig. 3.** Left-most: the HI-face. Second: HI-face blended with AS-face half-half in PCA space. Third: using Artistic-Style Decomposition (ASD) on AS-face in the blending. Fourth: using ASD on AS-face and Identity Preservation (IP) on HI-face in the blending. Right-most: the AS-face.

$$\begin{aligned} AS_{Style} &= AS\text{-face} - AS_{Ident}, \\ AS_{Ident} &= AS\text{-face} * \phi * \phi^T \end{aligned} \quad (2)$$

**Artistic-Style Transformation.** Finally, to transform a face, we carefully compose  $HI_{preserved}$  and  $AS_{Style}$  together with Algorithm 3.. The  $HI_{preserved}$  is preserved for the parts of the face containing only HI-face. For the rest of the face, the pure artistic-style,  $AS_{Style}$  is added directly to the HI-face. To adjust the level of the  $AS_{Style}$ , the scaling factor  $u$  with the range of 0 to 1 is multiplied to the  $AS_{Style}$  in order to control the transformation, e.g., from a slide-bar control, as shown in Figure 7.

**Algorithm 3.** Transformation Algorithm**Input:** HI-face, AS-face,  $HI_{preserved}$ ,  $u$ .**Output:** AH-face.

---

```

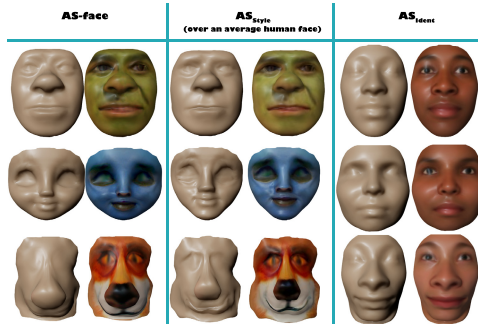
1:  $AS_{Style} = \text{ArtisticStyleDecomposition}(AS\text{-face});$ 
2: for  $i = 1$  to  $sizeofvector(AH\text{-face})$  do
3:   if  $i \in HI_{preserved}$  then
4:      $AH\text{-face}(i) = HI\text{-face}(i);$ 
5:   else
6:      $AH\text{-face}(i) = HI\text{-face}(i) + u * AS_{Style}(i);$ 
7:   end if
8: end for

```

---

## 5 3D Face Construction

The 3D deformation gradient ( $AH\text{-face}_{Geo}$ ) and texture gradient ( $AH\text{-face}_{Tex}$ ) created from the IP algorithm are neither a 3D model nor its texture image but they are simply transformation operators. Thus, the next step is to construct the 3D face model and its texture image from these operators.



**Fig. 4.** AS-face,  $AS_{Style}$  (applied over a face structure for illustration purpose), and  $AS_{Ident}$ . If the  $AS_{Ident}$  is not removed from the AS-face, it will be mixed to HI-face in the transformation process making the face less recognizable.

**3D Geometric Construction.** To construct a 3D facial geometry, the  $AH\text{-face}_{Geo}$ , which is transformation matrices, are applied to average face's meshes by using a deformation gradient framework [15]. The main constraint of this framework is the shared vertices of each triangle mesh. The least-square is used to balance the affine transformations while maintaining the positions of the shared vertices. This transformation can be solved by a sparse matrix optimization by Eq. 3 where  $T$  is the linear operator of the average face surface and  $x$  is the vector containing the vertex positions of the resultant artistic face. It can be performed in real-time with Cholesky factorization [15]. Since the  $AH\text{-face}_{Geo}$  is already identity preserved, the constructed 3D face geometry maintains its identity with only necessary artistic-style added on.

$$\arg_x \min \|(Tx - AH\text{-face}_{Geo})\| \quad (3)$$

**Texture Construction.** For the texture synthesis, AH-face<sub>Tex</sub> is used to construct the artistic facial texture by using Poisson image editing technique [11]. In this technique, the image gradients are applied to an average face texture (in RGB colorspace) in a least-square sense constraining by the pixels nearby as shown in Eq. 4 where  $F$  is the resultant artistic face texture and  $Avg$  is an average face texture. This operation is performed in each color channel separately. Again, since the AH-face<sub>Tex</sub> is already identity preserved, the constructed face texture maintains its identity with only necessary artistic-style added on.

$$\arg_F \min \sum_{all\ pixels} \|(F - Avg) - AH\text{-face}_{Tex}\| \quad (4)$$

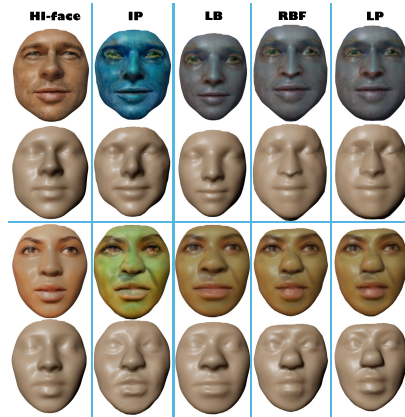


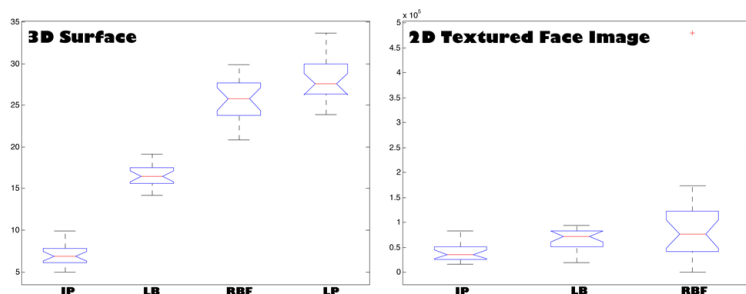
Fig. 5. The results generated by different techniques

## 6 Evaluations

The goal of this numerical evaluation is to assess the similarity of the artistic faces (AH-faces) synthesized by our technique to their starting human faces (HI-faces) in comparison with other face synthesis techniques. Since the facial similarity perception is highly subjective and varies from person-to-person, especially, when a person has to compare more than two unfamiliar faces, we choose to make use of a facial recognition technique as a standard measurement instead of using a human-based user study. Moreover, in a closed environment (calibrated parameters), this technique yields fairly accurate recognition. In our study, the forty-five AH-faces, constructed from five HI-faces and nine AS-faces, are used as the main subjects to be evaluated. Figure 7 shows some transformation results generated from our IP algorithm.

**3D Surface Evaluation:** To measure the 3D surface similarity, the AH-faces' 3D models are projected to eigen-vectors of 3D position constructed similar to the construction of  $\phi_{Geo}$  in Section 3. The face synthesis techniques to be compared in this evaluation are 1) our Identity Preservation (IP), 2) Linear Blending (LB), 3) Radial Basis Function (RBF) [10], and 4) Laplacian (LP) [9] techniques. To be able to objectively compare the outcomes of the aforementioned techniques, all synthesized faces are generated with the highest level of artistic-style possible for each particular technique. For our IP

technique, the  $AS_{Style}$  is added to be 100 percent to the HI-face (the second right-most column in Figure 7). For LB technique, the artistic-style is added only 50 percent because adding 100 percent means having no human identity on the face at all. For RBF and LP techniques, the AS-face is used as a based model and 83 feature points on facial contours of the HI-face are used as control points to morph the AS-face to reflect the human identity which implies 100 percent artistic-style is added. Figure 5 demonstrates some examples generated from each technique. The 3D similarity between two faces, the resultant face AH-face and the starting HI-face, are then measured by their Euclidean distances in this PCA space. Figure 6 (left) shows ANOVA of the surface-based distance of each technique. From the ANOVA, it shows that our IP technique is significantly superior than other techniques in synthesizing 3D surface surface which is most resemblance to the original starting HI-face. Mathematically, the LB is supposed to synthesize the surface that is very close to the HI-face since it directly represents half of the HI-face. Our IP technique, however, takes only small fragments (approximately 10% of the HI-face surfaces) which are considered most crucial to preserve the human identity, to generate the resultant AH-face. For RBF and LP, the synthesized surfaces are created from only a spare set of major facial landmarks. The majority of the surfaces are still closer to the AS-face than the HI-face. The exception is when the AS-face is also similar to the HI-face; RBF and LP tends to generate better results.



**Fig. 6.** The comparison of our Identity Preservation (IP), Linear Blending (LB), Radial Basis Function (RBF), and Laplacian (LP) technique by using ANOVA. Left: the 3D surface evaluation. Right: the 2D textured face image evaluation.

**2D Textured Face Image Evaluation:** This evaluation is designed to assess the facial similarity in terms of shape and texture together. To measure the similarity, the AH-faces' images are projected to  $\phi_{Image}$  (constructed from Section 3) for measuring their distances in PCA space. To achieve this, all AH-faces are rendered with their textures to images by using Lambert shading from their frontal view. The similarity between two faces are then measured by their Euclidean distances in this PCA space. The face synthesis techniques to be compared in this evaluation are 1) our IP, 2) LB, and 3) RBF techniques. Again, all synthesized faces are generated with the highest level of artistic-style possible for each particular technique. Since RBF technique is used only for surface synthesis, the facial texture for RBF is synthesized using LB technique. The LP is excluded in this evaluation because it behaves very similar to the RBF technique. Figure 5 demonstrates some examples generated from each technique. Figure 6 (right) shows ANOVA of the image-based distance of each technique. From this evaluation, our

IP technique generates results generally closer to the HI-faces than other techniques. The results are not obviously significant as in the 3D surface evaluation due to color factors. This image-based evaluation mainly considers colors (RGB) over 2D shape. Since the LB blends half of the HI-face’s colors to the result, it favors this evaluation in some degrees (Figure 5, the second example (the green face)). In contrast, our IP technique also tries to keep its original artistic-style.



**Fig. 7.** Transformation results created from our IP algorithm. The left most column shows the input HI-faces. The right most column shows the input AS-faces.

## 7 Discussion and Conclusion

This paper presents a 3D face transformation with a facial identity preservation algorithm. The artistic face is crafted starting from an input 3D human face model. The crafting process is designed to carefully add the artistic-style from the input artistic face to the human face model. The identity preservation algorithm prevents the improper transformation from altering the main identity of the human face. The numerical

evaluations illustrate that our technique achieves its goal in preserving the human identity significantly better than other techniques measured by a 3D and an image-based facial recognition systems. In the near future, we plan to conduct a comparative user study [5] from a human perspective which will be benefited to the research community in this direction. This work can be used to craft a variety of 3D faces for many purposes such as for movies, games, virtual realities, and plastic surgeries. The main limitation of this work is that every 3D face model to be used in the process requires to be registered first. This is because our PCA representations are constructed from a certain facial topology. This problem can be alleviated by using a mapping technique suggested in the deformation gradient framework [15] to transform any 3D models to a certain topology.

## References

1. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3D faces. In: *Proceeding of ACM SIGGRAPH*, pp. 187–194 (1999)
2. Gooch, B., Reinhard, E., Gooch, A.: Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph.* 23(1), 27–44 (2004)
3. Hsu, R.-L., Jain, A.K.: Generating discriminating cartoon faces using interacting snakes. *IEEE Trans. Pattern Anal. Mach. Intell.* 25(11), 1388–1398 (2003)
4. Joshi, P., Tien, W.C., Desbrun, M., Pighin, F.: Learning controls for blend shape based realistic facial animation. In: *Proc. of SCA*, pp. 35–42 (2003)
5. Ledda, P., Chalmers, A., Troscianko, T., Seetzen, H.: Evaluation of tone mapping operators using a high dynamic range display. In: *ACM SIGGRAPH Papers*, pp. 640–648 (2005)
6. Lewis, J.P., Mooser, J., Deng, Z., Neumann, U.: Reducing blendshape interference by selected motion attenuation. In: *Proceeding of I3D*, pp. 25–29 (2005)
7. Liu, J., Chen, Y., Gao, W.: Mapping learning in eigenspace for harmonious caricature generation. In: *Proc. of ACM MM*, pp. 683–686 (2006)
8. Memarsadeghi, N., Mount, D.M., Netanyahu, N.S., Le Moigne, J.: A fast implementation of the isodata clustering algorithm. *Int. J. Comput. Geometry Appl.* 17(1), 71–103 (2007)
9. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Fibermesh: Designing freeform surfaces with 3D curves. *ACM Trans. Graph.* 26(3) (2007)
10. Noh, J.Y., Neumann, U.: Expression cloning. In: *Proc. of SIGGRAPH 2001*, pp. 277–288 (2001)
11. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. *ACM Trans. Graph.* 22(3), 313–318 (2003)
12. Seol, Y., Lewis, J.P., Seo, J., Choi, B., Anjyo, K., Noh, J.: Spacetime expression cloning for blendshapes. *ACM Trans. Graph.* 31(2), 14:1–14:12 (2012)
13. Sifakis, E., Neverov, I., Fedkiw, R.: Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph.* 24(3), 417–425 (2005)
14. Sucontphunt, T.: A practical approach for identity-embodied 3d artistic face modeling. *Int. J. of Computer Games Technology*, Article ID 781950, 9 (2014)
15. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. *ACM Trans. Graph.* 23(3), 399–405 (2004)
16. Tena, J.R., De la Torre, F., Matthews, I.: Interactive region-based linear 3d face models. *Proc. of ACM SIGGRAPH*, pp. 76:1–76:10 (2011)
17. Yin, L., Wei, X., Sun, Y., Wang, J., Rosato, M.J.: A 3D facial expression database for facial behavior research. In: *FGR 2006*, pp. 211–216 (2006)

# Real Time Visualization of Large Data Using Clustered Projections

Pranav D. Bagur

National Institute of Technology, Karnataka, India  
prnvbagur@gmail.com

**Abstract.** Modern graphics systems rely on many techniques to achieve interactive frame-rates for rendering large and complex scenes. Impostoring is a popular technique where rendering of repetitive geometry is optimized by leveraging pre-calculated information for each type of repeated geometry. This paper discusses a view independent impostor-algorithm that stores the sampled projections of a given surface in a clustered format. The implications of such a format on the rendering performance and storage are discussed along with some of its advantages like selective adjustment of level of detail and shadowing.

**Keywords:** image based rendering, impostor rendering, real-time rendering.

## 1 Introduction

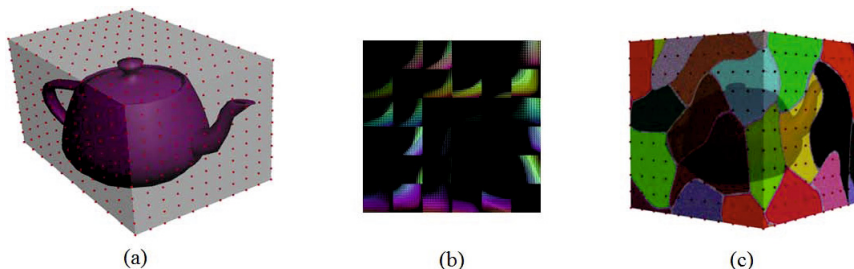
Impostors, unlike triangulated meshes, are fill-rate dependent using minimal amount of geometry yet rendering with per-pixel level of detail. The number of polygons used to render any impostor is fixed and is extremely low compared to other methods. This and the fact that only visible fragments of the impostor reference surface are processed each frame in the pixel shader makes impostoring one of the most performance efficient techniques. The technique proposed in this paper builds upon view-dependent displacement mapping[1] and proposes an alternate representation to achieve accurate rendering from any viewing angle. The representation uses indexing at a cluster level to allow retrieving data using only a binary search on the GPU.

## 2 Background

The technique of view dependent displacement mapping[1], involves pre-computing a texture for every potential view from which the surface can be looked at. The visual quality obtained using this technique is extremely good, although large amount of memory is utilized. Relief mapping uses the technique of ray marching[2], which involves a linear search followed by a binary search to determine the exact point of intersection with the surface. Although it produces good visual quality, it suffers from the performance point of view, since all the processors, processing a set of pixels must wait until the last pixel finds its point of intersection.

### 3 Methodology

Any surface in 3D-space has infinitely large number of view-rays passing through it. To quantize the number of such rays, the surface is bound by a finitely sampled cuboid which encloses it from all sides as shown in fig.1(a). Any point on the surface can now be mapped to a ray originating at one sample point and ending at another. Such a sampling produces a large square matrix having every sample point against another, storing a null if the corresponding ray doesn't intersect the surface and the attributes of the surface like its position, depth and normal direction for others.



**Fig. 1.** (a) Teapot surface enclosed by a finitely sampled cuboid (b) Shows the texture obtained by sampling the teapot using 486 sample points. Each RGBA corresponds to an intersection with 3 bytes storing the normal direction and 1 byte storing the distance from the reference surface (c) The cuboid reference surface in fig1.(a) with sample points put into different logical clusters.

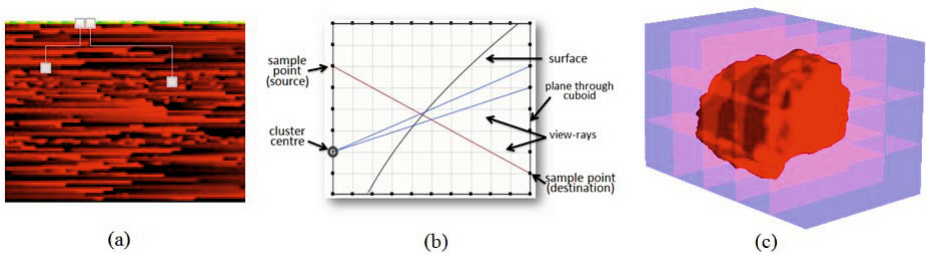
Several large textures are needed to hold this data from the matrix on the GPU[fig.1(b)]. At runtime, the view-rays originating from the camera are dynamically mapped to a pair of sample points on the cuboid to obtain the offset into these textures. Any ray not passing through one of these sample points directly, is given the weighted average of the four sample points surrounding it. The surface attributes obtained at the offset can be used for further computation. Even though this naïve implementation is extremely efficient in terms of performance, it occupies very large amount of storage space.

As the number of sample points on the cuboid increase, so does the amount of redundant sampling information. To reduce the amount of redundancy, the sample points on the cuboid are put into clusters[fig.1(c)] using the k-means algorithm and data is stored only for view rays originating at cluster centers and ending at one of the sample points. For other sample points, only the destination point numbers of view rays from the sample point which make an intersection with the surface while the view ray from corresponding cluster center to the destination point doesn't or vice-versa are stored[Fig.2(a)]. These other points are put with cluster centers such that the amount of data to be stored for them is minimum. The reason for choosing such a clustering criteria is to maintain the correct shape of the surface, while its properties are left for approximation.



These clusters tend to localize spatially as fig.1(c) depicts, although it depends fairly on the surface under consideration.

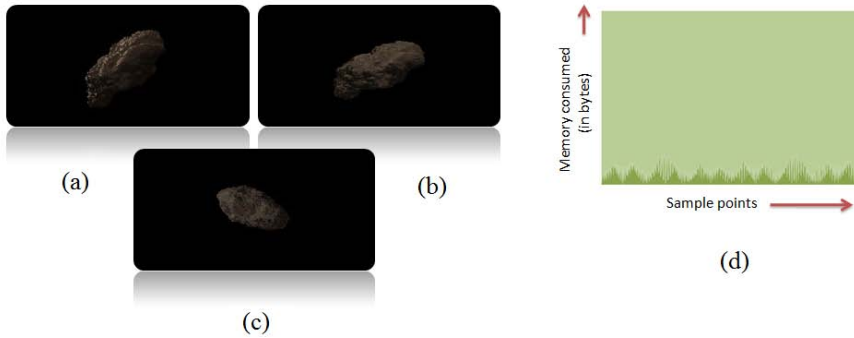
At runtime, the two sample points through which the view ray passes is first determined and the range within which the data for this source-destination pair can be found is obtained[fig.2(a)]. Binary search is used to check if there is an entry for the pair within this range. Then, the data for cluster center-destination pair is obtained from the other texture and compared against the entry to discard pixels. To obtain the surface properties at the point of intersection, the cluster center-destination view ray is offset as shown in fig.2(b). Since the data is stored for every ray coming out of every cluster center, the surface properties can be approximated fairly well. Given fairly large number of sample points, the offset by which the destination sample point has to be moved can be calculated directly.



**Fig. 2.** (a) Shows a slice of the texture storing values corresponding to non cluster-centers. The first row stores offsets corresponding to position at which the data for that sample point starts (b) Shows a plane passing through the cluster centre, the source and the destination points. The dots represent the sample points on the cuboid. The surface is assumed to be planar and an offset is calculated. An additional amount is added or subtracted depending on the curvature and requires an additional texture read. (c) An asteroid surface sampled using subdivisions of the bounding cuboid. Each subdivision has its own set of sample points.

Using a cuboid instead of a billboard eliminates the need to move the reference surface around since its projection on the viewport is always a super-set of the projection of the surface. Also, the attributes of the surface can be statically bound to the cuboid, mapping the surface transformations directly onto the cuboid. By subdividing this bounding cuboid further as shown in fig.2(c), we can reduce unnecessary sampling and efficiently manage bulky textures. This also helps in controlling the level of detail since these component cuboids can be designed to have varying sample points depending on the end of the surface being visualized by the cuboid and the context in which the surface is used. Filter textures can also be used to disable rendering samples from certain areas and from certain component cuboids to effectively control the level of detail at run-time.

To determine if a point on the surface is in a shadow region, the point of intersection on the surface of the ray from light source to the point is calculated. If this is in a different cuboid, the point can directly be marked as being in



**Fig. 3.** (a) Shows the rendering of an asteroid sampled at  $90 \times 54 \times 54$  obtained using un-clustered format (b,c) Shows the same asteroid surface rendered using clustered format with normal and depth approximation (d) Shows the amount of compression achieved after clustering for the asteroid model. The sample points are picked at random and correspond to non cluster-centers.

shadow. Otherwise, the ray length is compared against the distance calculated from the original source point on the cuboid to determine the same.

## 4 Results and Discussion

Rendering surfaces stored in an un-clustered format is extremely efficient as the offset into these textures can be calculated directly on the fragment shader. A frame rate of 620fps was achieved while rendering a rectangle that completely covered a  $640 \times 480$  window. Clustered data rendering incurs the overhead of binary search, but since the data stored in this case is comparatively much lesser [ $<10\%$ ], the search step is restricted to less than 5-6 steps. A frame rate of 119fps was achieved while rendering a similar rectangle as above that was stored in clustered format. Conceptually, the algorithm should perform much better since only a binary search is performed but the frame-rate is reduced due to normal approximation. A frame rate of 57fps was achieved while rendering fifty thousand objects that occupied more than 70% of the viewport. The same saturated at around 73fps when less than 5% of the screen was occupied, hinting of the real cost of processing the pixels in fragment shader. Like any other impostor algorithm, the figures are fill rate dependent. The algorithms discussed have been implemented using XNA 3.1 and HLSL 3.0. The timings mentioned above have been measured using a GTX 520.

## References

1. Wang, L., Wang, X., Tong, X., Lin, S., Hu, S., Guo, B., Shum, H.Y.: View-dependent displacement mapping. *ACM Trans. Graph.* 22(3), 334–339 (2003)
2. Policarpo, F., Oliveira, M.M., Comba, J.L.D.: Real-time relief mapping on arbitrary polygonal surfaces. In: *SI3D 2005, Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, pp. 155–162. ACM Press, NY (2005)

# Crowdsourcing 3D Motion Reconstruction

Yueh-Tung Chen<sup>1</sup>, Cheng-Hsien Han<sup>1</sup>, Hao-Wei Jeng<sup>1</sup>, and Hao-Chuan Wang<sup>1,2</sup>

<sup>1</sup> Department of Computer Science,  
National Tsing-Hua University

<sup>2</sup> Institute of Information Systems and Applications  
National Tsing-Hua University

No.101, Sec. 2, Kuang-Fu Rd., East Dist., Hsinchu 300, Taiwan (R.O.C.)  
{dan801212, richo2192000, locky4567}@gmail.com,  
haochuan@cs.nthu.edu.tw

**Abstract.** Reconstructing 3D motions from 2D video recordings is a useful yet difficult task. In this paper, we present a crowdsourcing system for motion reconstruction. Our insight is that we may outsource motion reconstruction to indefinite users online by allowing the users to contribute their motions with popular motion sensors shipped with game consoles (e.g., Microsoft's Kinect). The system we prototyped, *Motionary*, provides functions for requesters to post videos of human motions on our website, and for interested contributors to help reconstruct these motions by performing these motions to our system. The system then records the 3D motions with Kinect during their performance. Our experience shows that the design has the potential to obtain quality motion data without great cost.

**Keywords:** Crowdsourcing, human computation, motion reconstruction, motion capture.

## 1 Introduction

Video is a popular way to record and archive human motions. Motion data archived in the 2D format can be difficult to use, thus raising the need of motion reconstruction, or extracting the motion information in 2D videos to appropriate 3D forms [1]. Motion reconstruction from 2D to 3D, however, remains to be a difficult problem. How do we extract and obtain quality 3D motion data of given videos at an affordable cost becomes a crucial research issue.

Existing solutions can be roughly divided into two categories, machine processing or human processing. In terms of machine processing, algorithms and heuristics are available to estimate human body poses in images [3], and it is also possible to reconstruct human motion from video to some extent (cf. [1][4]). Machine processing's strength is on automating the process of reconstruction, yet the quality of results is likely to vary and may depend greatly on the complexity of the human motion to be captured and modeled.

Human processing, on the other hand, involves using motion capture devices to record human movement. It is possible to instruct human actors to perform motions shown in the videos, and then record their performance as a means to reconstruct the motions from 2D to 3D. The quality of motion data collected in this way is arguably

better than machine processing as long as the human actor is proficient and the motion capture hardware and software is of satisfactory resolution. However, human experts and professional equipments can be costly to obtain, and so the financial cost involved in human processing can be high.

In this paper, we propose a crowdsourcing system called *Motionary* that packages the task of motion reconstruction as micro-operations suitable for common Internet users to handle (see [2] for an overview of crowdsourcing as a computational technique). *Motionary* provides web-based interfaces for requesters to post videos and make requests of motion reconstruction, and for interested contributors to contribute motion data with one of the non-expensive and widely available motion sensors, Microsoft's Kinect. We also leverage the crowd to control the quality of motion data contributed by different users through a social rating mechanism. We demonstrate that it is viable to collect quality 3D motion data without relying on professional actors and equipments with the proposed crowdsourcing approach.

## 2 The *Motionary* System

*Motionary* provides web-based interfaces and functions to support users' request making (i.e., posting a video), contribution making (i.e., performing the motion shown in a video), and social filtering (i.e., rating the quality of contributed videos). As shown in Figure 1, requesters can post a request for reconstructing the motion of particular videos in our system and recruit contributors with specified amount of monetary incentive. The system will broadcast and make the requested tasks available for indefinite Internet users to handle. Interested users can then play the role of worker to contribute motion data through their Kinect sensors.

Figure 2 shows the interface for making requests. At the request page, requesters can post videos from YouTube that they want to extract motion data from. When posting a video, the requester also provides associated information, such as which part of the video to reconstruct, the date of expiration for the request, the amount of payment or reward, and any further instruction (e.g., what motion to perform where multiple motions are present in the video at the same time).

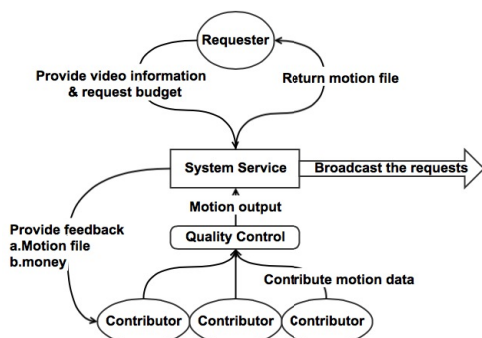


Fig. 1. Main system components

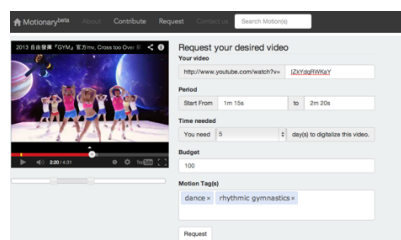


Fig. 2. Interface for users to request reconstructing the motion shown in a video

Contributors are the source of motion data. When the contributors enter the website, they will see a list of non-expired video requests (Figure 3). They can choose tasks interesting to them, and use their Kinect sensors to record their performance of the motion shown in the video. In the contribution page (see Figure 4), we use a JavaScript library called ZDK provided by Motion Arcade (<http://zigfu.com/>) to control and collect data from the Kinect sensor. The ZDK provides us functions to access Kinect from the web browser. The system tracks and records the positional data of 24 body joints. Continuous movement is sampled at the rate of 30 frames per second. The positions of body joints of a performance are stored as an array and externalized as a text-based log file. The system also provides the function to replay the log files for contributors to view the motions they just contributed, and for all the users to review and rate.

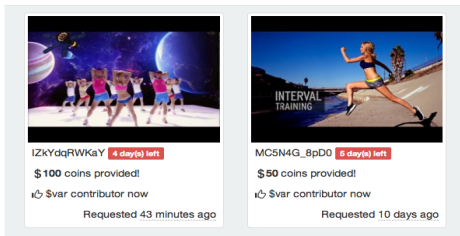


Fig. 3. Requests shown in a list

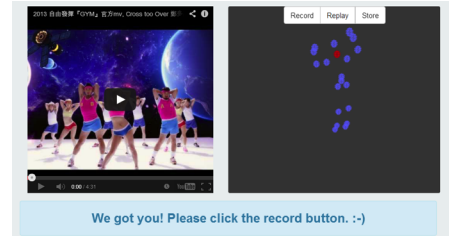


Fig. 4. Contribution page

To control the quality of crowdsourced motion reconstruction, contributed motions for a request are made available for the public to rate after passing the deadline of contribution. During the rating phase, users can rate each of the contributed motions using the number of stars as indicator. All the star ratings are aggregated to rank contributed videos.

Payment is the external reward that we used to attract user for using our system. We built a virtual currency system for this purpose. Depending on different activities, users will receive and pay different amount of payment. At the time requesters make a request, they can choose the budget or the amount of total reward they'd like to provide for the motion reconstruction task. The contributors will receive payment according to the quality of their contribution based determined by social ratings. One rule that we implemented in the current system is as below:

$$\text{Contributor's reward} = \text{budget} * (\text{number of stars the contributor received}) / (\text{total number of stars all contributors received}).$$

With this rule, we expect that contributors will be motivated to maintain the quality of their contributions in order to receive higher amount of reward.

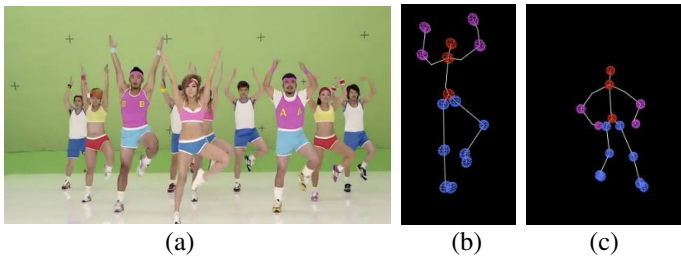
### 3 Pilot Study

As a preliminary study, we chose five videos from YouTube with different contents and levels of difficulty. We set up a booth at the campus of National Tsing Hua

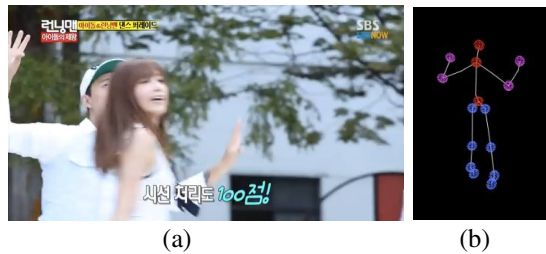
University to showcase the system and invited interested students to contribute their motions and rate the motion data contributed by other users.

After two week of data collection, we collected 117 pieces of motion data across the five videos from 39 participants. In addition, we also had 431 ratings from the users. Based on the aggregated rating score, we could find that data with high score (e.g., Figure 5b) were indeed of better quality than the data with low score (Figure 5c).

Some of the videos are more difficult for motion reconstruction since the human motion to be modeled can be blocked partially by obstacles sometimes. It can be difficult for pure machine processing methods to reconstruct these motions due to missing information in this type of videos. In our study, however, participants still contribute quality motions under this circumstance (see Figure 6), demonstrating the unique processing utility and value of the crowdsourcing system.



**Fig. 5.** Snapshots of (a) original video (b) crowdsourced motion data with rating score 4.5 (c) crowdsourced motion data with score 1.5



**Fig. 6.** Snapshots of (a) original video with a partially blocked figure (b) collected motion data with rating 4 for the blocked figure

## References

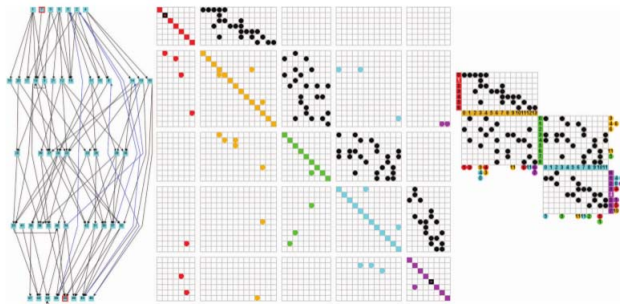
1. DeFranco, D.E., Cham, T.-J., Rehg, J.M.: Reconstruction of 3-D Figure Motion from 2-D Correspondences. In: Computer Vision and Pattern Recognition, CVPR (2001)
2. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing Systems on the World-Wide Web. *Communications of the ACM* 54(4), 86–96 (2011)
3. Gargallo, P., Sturm, P.: Bayesian 3D Modeling from Images Using Multiple Depth Maps. In: IEEE Workshop on Motion and Video Computing, pp. 885–891 (2005)
4. Wei, X., Chai, J.: VideoMocap: Modeling Physically Realistic Human Motion from Monocular Video Sequences. In: Proc. of SIGGRAPH (2010)

# Understanding Which Graph Depictions Are Best for Viewers

Johanne Christensen, Ju Hee Bae, Ben Watson, and Micheal Rappa

North Carolina State University,  
Raleigh, NC 27695, United States  
{jtchrist,jbae3,bwatson,mrappa}@ncsu.edu

**Abstract.** We use data from a study of three different graph depictions: node-link, centered matrix, and quilts to explore how pathfinding time is influenced by the graph structure, measured by the number of nodes, links, skips and layers. We use regressions to determine the influence of these attributes. Furthering this idea, we begin to explore how individual users navigate through graphs.



**Fig. 1.** From left to right, the same graph in: node-link, centered matrix, and quilts

## 1 Introduction

There are many ways to depict graphs, among them the common node-link and the less common matrix depictions. Other depictions, such as quilts, have been developed to address the problem of readability in large graphs. A recent study of these three depictions[1] has shown quilts to be the fastest to navigate in a pathfinding task. Using regressions, we are able to explore the strengths and weaknesses of each depiction type relative to the size attributes of the graph. This gives us a better understanding of what affects readability and navigation in these depictions. We also begin to study user movement through the graph on a pathfinding task, and the effect of depiction type.

## 2 Related Works

Node-link diagrams lose legibility as they increase beyond 50 nodes[2]. Keller et al. [3] has shown node-link is best for small, sparse graphs, while matrices improve understanding of complex graphs. The matrix depiction has been improved, including centered matrices (our name) [4], and quilts [5]. Work has also been done in the area of cognitive modeling for graphs [6] to predict graph reading performance. Other cognitive models focus on path planning [7] in similar domains. Visualization techniques such as eye tracking [8] have been studied to better understand how users navigate through graphs.

## 3 Effects of Graph Attributes

The data used in this study was collected in [1]. The study used a five factor (3 depictions x 3 nodes x 2 links x 2 skips x 2 layers) within-subjects design. Regressions were produced in R using the stepwise function, with the graph attributes and their interactions as the independent variables, and time to solve (normalized within depiction by user) as the dependent variable. The node attribute is log transformed since the curve of node count vs time resembles a log curve. We hypothesize that navigation is initially more difficult as nodes increase, but later outweighed by the resulting increase of links, and thus available paths.

### 3.1 Results

Except where noted, we only discuss coefficients which are significant ( $p \leq .05$ ).

**Table 1.** Significant Coefficients per Depiction

Coefficient	Node-Link	Centered Matrix	Quilts
log(nodes)	-37.44	Not Sig	Not Sig
links	-6.69	-8.32	-4.60
layers	-23.09	7.53	32.97
skips	Not Sig	-9.06	Not Sig
log(node)*links	1.61	1.67	1.20
log(node)*layers	5.12	-1.83	-5.86
log(node)*skips	Not Sig	1.84	Not Sig

In node-link depictions, the coefficient of the log transformed node element indicates that time to solve decreases as number of nodes increases. Links and layers also had significant decreases (see table 1) in time as they increase, but skips have little effect. The effects of log(nodes)\*links and log(nodes)\*layers increase time.

In the centered matrix, skips had the most influence, decreasing the time. Links also decreased the time, but increasing layers increased the time to solve.



The coefficient for  $\log(\text{nodes})$  indicated a large decrease in time similar to node-link, however it was not significant. The interaction of nodes with links and skips increased time, and the interaction with layers decreased time.

In the quilts depiction we see that layers increase time, while an increase in links lowers time. As with centered matrices, the interactions of the nodes with layers and links are significant, while nodes alone are not. It is important to note that quilts were the fastest solved of the three depictions.

### 3.2 Discussion

We infer that more nodes and links create more viable paths, which decreases time in node-link graphs. Since link count is relative to node count, the node interactions effects can be explained as a result of increasing visual confusion in the graph layout, making links harder to follow. Familiarity may be an unobserved effect on time to solve, as all but one of the participants were already familiar with this type of graph depiction.

In centered matrices, the number of nodes relative to the other variables is significant, but nodes alone are not. We suspect the effects of node increases is harmful when links are sparse because they are harder to follow. Similarly, adding layers improves time when nodes are also increased, thus the negative coefficient for  $\log(\text{nodes}) * \text{layers}$ . Skips are colored and encoded spatially (in fig 1, skips are above the diagonal), making them easy to find and improving time.

Quilts results are similar to the matrices, which follows as they are a condensed representation of that depiction. Skips have no significance here, probably because skips in quilts require more work to find and follow. The interactions of nodes and links and layers follows the same logic as with matrices.

Overall, quilts are the fastest, followed by node-link and centered matrices. Quilts handle complexity introduced by nodes well, while node-link improves with the increase of layers. Centered matrices' main strength is in following skip nodes. Across all three increasing links lowers time, supporting the idea that the number of valid paths are increasing as well. Conversely, node increase in conjunction with link increase mutes the positive effect of increasing links alone.

## 4 Exploring User Behaviors

The work thus far gives us insight about into affects of graph depiction on readability. However, a reliable regression prediction model of time to solve has been difficult to find. We believe this is due to the wide variances in user times. This in turn has raised more questions: how do individual users navigate through graphs, and how do these behaviors change as depiction type is changed?

We have begun a more in depth analysis of the log data collected from each click during the tests in the original experiment. Preliminary results show promise. The average number of moves per graph is 13.57 for quilts, 17.7 for node-link, and 21.18 for centered matrices. Furthering this, we intend to analyze the similarities in user behavior across depictions and individual graphs by

looking at the paths taken to solve the graph. We expect these results to further inform us in the cognitive processing behind this type of task.

## 5 Conclusions and Future Work

This work shows the influential factors affecting pathfinding time across graph depictions, and has promise in giving us a better understanding of the strengths and weaknesses of each depiction. The centered matrix depiction is most effective for following skips, but the slowest overall. Quilts do not do well with skip links, but the times from this depiction indicate that it is easier to follow and solve. Node-link graphs have the implicit advantage of being well known, negating the learning curve cost of the others. By pinpointing these factors, we have a better understanding of what users find helpful or impeding to the pathfinding task and an insight into the cognitive process of pathfinding.

Future work will further this analysis for a more fine-grained understanding of user behavior during pathfinding. We also intend to build a cognitive model for testing these and other depictions for their effectiveness in different situations.

## References

1. Bae, J., Watson, B.: Developing and evaluating quilts for the depiction of large layered graphs. *IEEE Transactions on Visualization and Computer Graphics* 17(12), 2268–2275 (2011)
2. Ghoniem, M., Fekete, J., Castagliola, P.: A comparison of the readability of graphs using node-link and matrix-based representations. In: *IEEE Symposium on Information Visualization, INFOVIS 2004*, pp. 17–24 (2004)
3. Keller, R., Eckert, C.M., Clarkson, P.J.: Matrices or node-link diagrams: Which visual representation is better for visualising connectivity models? *Information Visualization* 5(1), 62–76 (2006)
4. Shen, Z., Maz, K.L.: Path visualization for adjacency matrices. In: *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization, EUROVIS 2007*, pp. 83–90. Eurographics Association, Aire-la-Ville (2007)
5. Watson, B., Brink, D., Stallmann, M., Devarajan, R., Rakow, M., Rhyne, T., Patel, H.: Visualizing very large layered graphs with quilts. In: *Proceedings of the IEEE Information Visualization Conference Poster* (2007)
6. Huang, W., Hong, S.-H., Eades, P.: Predicting graph reading performance: A cognitive approach. In: *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation, APVis 2006*, vol. 60, pp. 207–216. Australian Computer Society, Inc., Darlinghurst (2006)
7. Reitter, D., Lebiere, C., Lewis, M., Wang, H., Ma, Z.: A cognitive model of perceptual path planning in a multi-robot control system. In: *IEEE International Conference on Systems, Man and Cybernetics, SMC 2009*, pp. 2897–2903 (October 2009)
8. Huang, W., Eades, P., Hong, S.H.: A graph reading behavior: Geodesic-path tendency. In: *IEEE Pacific Visualization Symposium, PacificVis 2009*, pp. 137–144 (April 2009)

# A New Modeling Pipeline for Physics-Based Topological Discontinuities: The DYNAMe Process

Annie Luciani<sup>1,2</sup>, Nicolas Castagné<sup>1</sup>, Philippe Meseure<sup>3</sup>,  
Xavier Skapin<sup>3</sup>, Saman Kalantari<sup>1</sup>, and Emmanuelle Darles<sup>3</sup>

<sup>1</sup> ICA Laboratory, Grenoble INP, Université de Grenoble, France

<sup>2</sup> ACROE, Grenoble, France

<sup>3</sup> Laboratoire XLIM, Université de Limoges-Poitiers, Poitiers, France

{firstname.lastname}@imag.fr,

{firstname.lastname}@univ-poitiers.fr

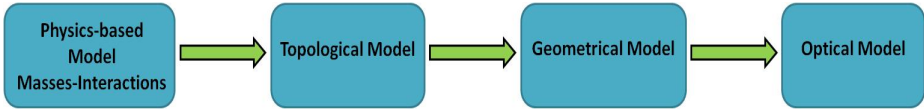
**Abstract.** This work introduces a new modeling/simulation pipeline for handling topological discontinuities as they appear in fracturing, tearing or cracking phenomena. This pipeline combines, in a cascade (1) physics-based particle modeling, which enables a wide variety of temporal phenomena, including physics state changes; (2) explicit topological modeling, which makes it possible to manage a wide variety of shape-independent topologies and robust topological transformations; and, in between, free, non predetermined association, enabling topological transformations all along the animation under control of the point-based movement produced by the physics-based model.

**Keywords:** Computer Animation, Physics-based Modeling, Masses-Interactions, Topological Modeling, Combinatorial Maps, Fractures.

## 1 A Physical→Topological Modeling Pipeline

Handling animation of topological changes, such as breaking, cutting, tearing or merging, is a new challenge in Computer Animation. Despite their differences, most existing works mix spatial features with embedded physical behaviors. Exemplary works are: with finite elements methods [1][2][3]; with meshless methods [4]; with mass-spring meshes [5][6][7][8] or Extended Distinct Element Methods [9]. Such embedding results in complex physico-geometrical re-structuring operations at the time of the topological changes, and can hardly be employed in a general way.

The DYNAMe modeling process introduces a new physical→topological modeling pipeline. This pipeline is rooted first on a clearly-cut dissociation of a physics-based animation stage from topological and geometrical stages. It organizes the complex modeling activity into four sub-activities, in a cascade (Figure 1): 1) physics-based particle modeling, which enables a wide variety of temporal phenomena; 2) explicit modeling of the topological aspects, which makes it possible to manage a wide variety of shape-independent topologies and of robust topological transformations; 3) management of the geometry, out of the topological model; 4) free, non predetermined, association between the physics-based particle modeling and the topological modeling stages.



**Fig. 1.** The Physico-topological modeling pipeline

Upstream, the physics-based particle model provides the necessary number of point-based animations. Downstream, these point-based animations are used first to control the topological model and its transformations, then to animate a geometrical model. With this cascading pipeline, each modeling part can be adjusted independently of the others, taking then benefit for its own optimization. Also, they can be combined to provide a wider variety of animated topological transformations, realistic or not, depending only on the wishes of the programmer or the user, thus extending the creative possibilities of computer animation tools.

## 2 Physics-Based Particle Modeling Stage

Upstream, on the side of physics-based modeling, particle-based modeling is particularly well adapted to the proposed pipeline, as it proved very soon its aptitude to support modeling of a wide variety of complex physical state changes [10][11][12]. In such approaches, physical state changes have so far mostly been handled onto in-between-particles elements, for example by removing or reconnecting springs [7][8], or more generally by employing appropriate physical interactions [13]. Handling topological changes onto these in-between elements usually results in an effect of visual matter removal artifact at the time of the fracture. To overcome it, we introduce a new system, called “Splitting Mass”, that enables modeling and handling fracture phenomena right on physical masses, and not through the interactions. Details on the Splitting Mass method can be found in [14]. At the end of this paper, the first examples have been obtained by employing this method on the physical side of the DYNAMe physical→topological pipeline. Differently than in mass-spring models when used in fractures and cuts, the Splitting Mass method does rely on any addition or removal of physical elements, nor on any modification or reconfiguration of the physical model. Hence, it guaranties invariance of computing time steps and of numerical stability, all along the topological transformations.

## 3 Topological and Geometrical Modeling Stages

Downstream, topological-geometrical modeling determines the transformations of a topological model, and the animation of a geometrical model, both under control of the point-based motions produced by the upstream physical model.

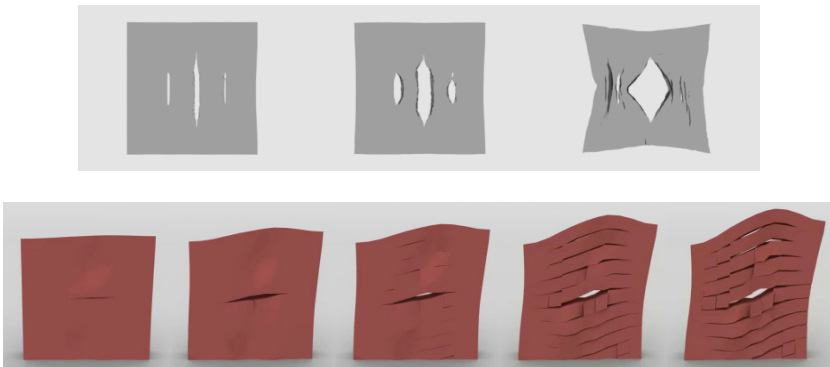
The introduction of an explicit modeling stage of the topology makes possible to manage a wide variety of shape-independent topologies and topological transformations. In order to benefit from a manageable representation of the topology and to ensure topological consistency all along the process, we root on a theoretically consistent topological formalism: the Generalized Combinatorial Maps or G-Maps [15].

The inputs of the topo-geometrical stage are the set of point-based motions generated by the upstream physical simulation, and an initial topological Generalized Map (G-Map) [16]. From both inputs, the topological/geometrical process consists, on each frame, in (1) *animating* the geometrical mapping of the current G-Map; and (2) *transforming* the topological G-Map. Animation consists in computing, on each frame, from the point-based movements, the displacement (*e.g.* the new position) of the geometrical model corresponding to the current state of the topological G-Map.

Handling topological transformations of the topological model is the most complex part. It requires mastering deep modifications of the topological model and consistent handling of their consequences all over the pipeline, at each temporal frame, all along the animation. As detailed in [17], in the topological modeling stage, topological transformations are organized into 4 sub-processes: (1) *Sensing*: It consists in extracting information from the Animation Functions Set, to trigger subsequent topological modifications. (2) *Selection process*: This step computes and outputs the list of topological cells on which the topological modifications will be applied. (3) *Topological modifications*: This sub-process applies the chosen topological modification onto the previously selected cells. (4) *Updating*: The topological transformations may require modifying the whole pipeline, to ensure its consistency with the new topology. On each frame, once these four topological modification steps have been applied, the pipeline turns back to the Animation process, and to the computation of the new geometrical model.

## 4 Modeling Experiments

Several models have been designed using the DNAME modeling pipeline, which exhibit various types of animated topological transformations such as complete and incomplete tears, holes, cracks and multiple fractures. The following figures present two of them: a model modeled with the Splitting Mass method, and a model visualized and animated through the topological modeling stage.



**Fig. 2.** Two experimental works. Top: Modeling tears, cracks and holes in a highly deformable visco-elastic surface, with the Splitting Mass method. Bottom: cracks in 3D plates.

**Acknowledgments.** This research was supported by the French *Agence Nationale de la Recherche* through the cooperative project DYNAMÉ - ANR-2009-CORD-007, and by the French Ministry of Culture.

## References

1. O'Brien, J.F., Bargteil, A.W., Hodgins, J.K.: Graphical modeling and animation of ductile fracture. In: Proc. of SIGGRAPH, pp. 291–294 (2002)
2. Bao, Z., Mo Hong, J., Teran, J., Fedkiw, R.: Fracturing Rigid Materials. *IEEE Transactions on Visualization and Computer Graphics* 13(2), 370–378 (2007)
3. Glondu, L., Marchal, M., Dumont, G.: Real-Time Simulation of Brittle Fracture using Modal Analysis. *IEEE Transactions on Visualization and Computer Graphics* 19(2), 201–209 (2013)
4. Pauly, M., Keiser, R., Adams, B., Dutré, P., Gross, M., Guibas, L.J.: Mesh-less animation of fracturing solids. In: Proc. of ACM SIGGRAPH (2005)
5. Ganovelli, F., Cignoni, P., Montani, C., Scopigno, R.: A multiresolution model for soft objects supporting interactive cuts and lacerations. *Computer Graphics Forum* 19(3), 271–281 (2000)
6. Metaaphanon, N., Bando, Y., Chen, B.-Y., Nishita, T.: Simulation of Tearing Cloth with Frayed Edges. *Comput. Graph. Forum* 28(7), 1837–1844 (2009)
7. Meseure, P., Darles, E.: X Skapin. Topology-based Physical Simulation. In: Proc. of VRIPHYS 2010 - Virtual Reality Interaction and Physical Simulation 2010 (VRIPHYS), Denmark, pp. 1–10 (2010)
8. Fléchon, E., Zara, F., Damiand, G., Jaillet, F.: A generic topological framework for physical simulation. In: 21st International Conference on Computer Graphics, Visualization and Computer Vision, pp. 104–113 (2013)
9. Imagine, T., Johan, H., Nishita, T.: A fast method for simulating destruction and the generated dust and debris. *The Visual Computer* 25(5-7), 719–727 (2009)
10. Luciani, A., Jimenez, S., Florens, J.L., Cadoz, C., Raoult, O.: Computational physics: A modeler simulator for animated physical objects. In: Proc. of Eurographics (1991)
11. Tonnesen, D.: Modeling liquids and solids using thermal particles. In: Graphics Interface 1991 (1991)
12. Greenspan, D.: Particle Modeling. Birkhauser (1997)
13. Luciani, A., Godard, A.: Simulation of Physical Object Construction Featuring Irreversible State Changes. In: Proc of WSCG, pp. 321–330 (1997)
14. Kalantari, S., Luciani, A., Castagné, N.: A new way to model physics-based topology transformations: Splitting MAT. In: Christie, M., Li, T.-Y. (eds.) SG 2014. LNCS, vol. 8698, pp. 146–153. Springer, Heidelberg (2014)
15. Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *Int. J. Comput. Geom. Appl.* 4(3), 275–324 (1994)
16. Darles, E., Kalantari, S., Skapin, X., Crespín, B., Luciani, A.: Hybrid Physical-Topological Modeling of Physical Shapes Transformations. In: Proc. of CASA 2011, pp. 154–157 (2011)
17. Luciani, A., Allaoui, A., Castagné, N., Darles, E., Skapin, X., Meseure, P.: MORPHO-Map, A New Way to Model Animation of Topological Transformations. In: Proc. of the 9th VISIGRAPP, Lisbon, Portugal, January 5-8 (2014)

# Author Index

- Bae, Ju Hee 174  
Bagur, Pranav D. 166  
Bu, Fenglin 39
- Cai, Hongming 39  
Castagné, Nicolas 146, 178  
Chang, Hsin-Yin 51  
Chen, Po-Ming 73  
Chen, Yueh-Tung 170  
Cheng, Wei-Chien 86  
Chi, Ming-Te 27  
Christensen, Johanne 174  
Christie, Marc 121
- Darles, Emmanuelle 178  
Delgado-Mata, Carlos 98
- Han, Cheng-Hsien 170  
Hsu, Che-Chun 51  
Hsu, Wei-Hsien 1  
Hu, Chen-Chi 27  
Hu, Pan 39  
Huang, Yi-Jheng 86
- Ibáñez, Jesús 98
- Jeng, Hao-Wei 170  
Jhan, Yu-Jyun 27  
Jorgensen, Carl-Johan 133
- Kalantari, Saman 146, 178  
Kung, Wen-Yao 51
- Lai, Chan-Yet 63  
Lai, Pei-Chun 121  
Lamarche, Fabrice 133  
Lee, Ruen-Rone 51  
Lee, Yen-Chun 51  
Li, Tsai-Yen 121  
Liao, Isaac 1  
Liao, Wen-Hung 73  
Lin, Wen-Chieh 86  
Lo, Yi-Hsiang 51  
Losada, Antonio G. 15  
Luciani, Annie 146, 178
- Ma, Kwan-Liu 1  
Meseure, Philippe 178  
Morishima, Shigeo 112
- Rappa, Micheal 174
- Sanokho, Cunka 121  
Skapin, Xavier 178  
Sucontphunt, Tanasai 154
- Theron, Roberto 15
- Vaquero, María 15
- Wang, Hao-Chuan 170  
Watson, Ben 174  
Wu, Hui-Yin 121
- Zakaria, Nordin 63  
Zhang, Zhuopeng 112