International Journal of Computer Vision (2022) 130:3100–3122
https://doi.org/10.1007/s11263-022-01682-w

RegGeoNet: Learning Regular Representations for Large-Scale 3D Point Clouds

Qijian Zhang[1,2] · Junhui Hou[1,2] · Yue Qian[1,2] · Antoni B. Chan[1] · Juyong Zhang[3] · Ying He[4]

Received: 10 October 2021 / Accepted: 22 August 2022 / Published online: 27 September 2022

Abstract

Deep learning has proven an effective tool for 3D point cloud processing. Currently, most deep set architectures are developed for sparse inputs (typically with a few thousand points), which are unable to provide sufficient structural statistics and semantic cues due to low resolutions. Since these architectures suffer from unacceptable computational and memory costs when consuming dense inputs, there is a pressing need in real-world applications to handle large-scale 3D point clouds. To bridge this gap, this paper presents a novel unsupervised neural architecture called RegGeoNet to parameterize an unstructured point set into a completely regular image structure dubbed as deep geometry image (DeepGI), such that spatial coordinates of unordered points are recorded in three-channel grid pixels. Intuitively, our goal is to embed irregular 3D surface points onto uniform 2D lattice grids, while trying to preserve local neighborhood consistency. Functionally, DeepGI serves as a generic representation modality for raw point cloud data and can be conveniently integrated into mature image processing pipelines. Driven by its unique structural characteristics, we are motivated to customize a set of efficient feature extractors that directly operate on DeepGIs for achieving a rich variety of downstream tasks. To demonstrate the potential and universality of our proposed learning paradigms built upon DeepGIs for large-scale point cloud processing, we conduct extensive experiments on various downstream tasks, including shape classification, object part segmentation, scene semantic segmentation, normal estimation, and geometry compression, where our frameworks achieve highly competitive performance, compared with state-of-the-art methods. The source code will be publicly available at https://github.com/keeganhk/RegGeoNet.

Keywords Deep learning · Large-scale 3D point clouds · Regular representation · Geometry image · Unsupervised learning

1 Introduction

Point clouds depict surface geometry with a set of irregular and unordered spatial points, which are characterized as an unstructured representation modality for three-dimensional structures. Different from common regular visual modalities defined on canonical lattice grids, such as 2D images/videos and 3D voxels, which can be naturally modeled by powerful 2D/3D convolutional neural networks (CNNs) (He et al., 2016; Karpathy et al., 2014; Krizhevsky et al., 2012; Long

Communicated by Yasutaka Furukawa.

This project was supported in part by the Hong Kong Research Grants Council under Grants 11219422, 11202320 and 11218121, and in part by the Natural Science Foundation of China under Grant 61871342.

✉ Junhui Hou
jh.hou@cityu.edu.hk

Qijian Zhang
qijizhang3-c@my.cityu.edu.hk

Yue Qian
yueqian4-c@my.cityu.edu.hk

Antoni B. Chan
abchan@cityu.edu.hk

Juyong Zhang
juyong@ustc.edu.cn

Ying He
yhe@ntu.edu.sg

[1] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR

[2] City University of Hong Kong Shenzhen Research Institute, Shenzhen, China

[3] School of Mathematical Sciences, University of Science and Technology of China, Hefei, China

[4] School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

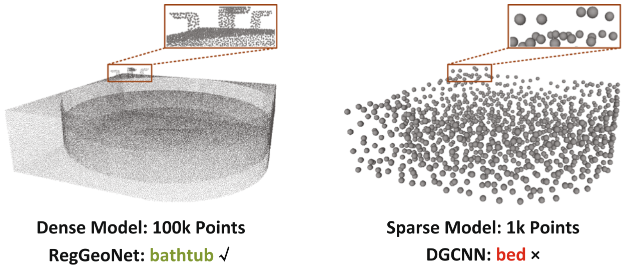

Fig. 1 A typical example for illustrating the necessity of exploiting large-scale point clouds. In this 3D shape classification task, the sparse model (right) fails to provide sufficient details for the *tap* component, which is crucial for distinguishing *bathtub* from other similar object categories such as *bed*

et al., 2015; Ren et al., 2015; Simonyan and Zisserman, 2014; Tran et al., 2015), point cloud data are known to be incompatible with conventional learning frameworks. This causes significant inconvenience in developing powerful and efficient learning pipelines for point cloud modeling.

Motivated by remarkable success of deep convolutional architectures in the last decade for visual recognition, the current point cloud research community has been devoted to adapting the existing mature design experience of CNNs and accordingly customizing "convolution-like" aggregators defined on point sets. Typically, one common approach to imitate the learning pattern of standard spatial convolution is to collect neighboring points using k-nearest neighbor (k-NN) search or ball query, then conduct local feature aggregation through channel pooling (Qi et al., 2017b) or kernel matching (Hua et al., 2018; Li et al., 2018b; Thomas et al., 2019; Verma et al., 2018). The concept of "neighborhood" can be further generalized from static spatial domain to dynamic feature domain, which deduces graph-based point cloud convolution (Wang et al., 2019). To mimic hierarchical feature abstraction mechanism of CNN workflows, one can introduce either heuristic down-scaling algorithms like farthest point sampling (FPS), as adopted in Qi et al. (2017b), Wu et al. (2019), or learning-based subset selection techniques (Nezhadarya et al., 2020; Yan et al., 2020; Yang et al., 2019) to achieve multi-scale and multi-level feature extraction.

Despite the proliferation of specialized deep set architectures, there still lacks a unified learning paradigm for generic point cloud processing. Moreover, as revealed in recent studies (Le et al., 2020; Liu et al., 2019d; Xu et al., 2020), most of the existing learning frameworks *incur extremely high computational complexity and memory consumption, where a large amount of superfluous calculations are wasted on the data structurization process*. This issue significantly limits model efficiency and scalability, especially when dealing with an increasing number of points. Consequently, previous studies mainly focus on processing sparse point clouds with several thousand points, and basically ignore exploiting large-scale point clouds.

In real-world application scenarios, however, there is still a pressing need for large-scale point cloud processing, since sparse models usually cannot provide sufficient structural or semantic information required for accurate and robust shape analysis or visual recognition. As illustrated in Fig. 1, the *tap* component is supposed to be the key clue for recognizing the *bathtub* object, without which it becomes impossible to distinguish it with the *bed* category, even for human observers. By contrast, the dense model is able to detailedly capture the crucial part of *tap*, making the subsequent decision much more reliable.

To bridge the gap with large-scale point cloud learning, we present a novel neural architecture called RegGeoNet, which is designed to convert an irregular and unstructured 3D point cloud into a completely regular 2D representation modality denoted as deep geometry image (DeepGI). As illustrated in Fig. 2, the three-dimensional spatial coordinates of input points are re-organized with learned "canonical" orders and captured in the color pixels of a three-channel image. Such a geometric modality transformation process can be described as "opening" a given 3D object with arbitrary geometric and topological structures and "flattening" the 3D surface onto 2D planar domains, while trying to maintain local neighborhood consistency between the original 3D domain and the parameterized 2D planar domain.

Just like conventional rule-based voxelization and mesh generation pipelines, RegGeoNet can be functionally viewed as a *neural pre-processing* procedure for point cloud structurization. Given a benchmark shape dataset, we separately fit a RegGeoNet on each of the point clouds to generate the corresponding DeepGI, which serves as a generic geometry representation modality and can be permanently preserved once created. The regular representation structure of DeepGI enables us to conduct downstream processing in a highly efficient manner, since *computationally expensive data structurization* can be naturally avoided. For example, spatial neighbors are inherently encoded in adjacent pixels, and point cloud down-sampling (*e.g.*, FPS) can be achieved by image pooling. In terms of the working mechanism, RegGeoNet shares the same big picture as the classic learning paradigm of "deep priors", as investigated in image/geometry processing areas (Chu et al., 2021; Gadelha et al., 2019; Gandelsman et al., 2019; Hanocka et al., 2020; Heckel and Hand, 2019; Ulyanov et al., 2018). Instead of training on a large amount of task-specific data, we treat the network structure itself as geometric prior and independently achieve regular geometry parameterization for given point clouds.

Technically, as illustrated in Fig. 3, we investigate a global-to-local (coarse-to-fine) surface parameterization scheme, which is composed of global anchor embedding (GAE) and local patch embedding (LPE). In the GAE module, we sim-

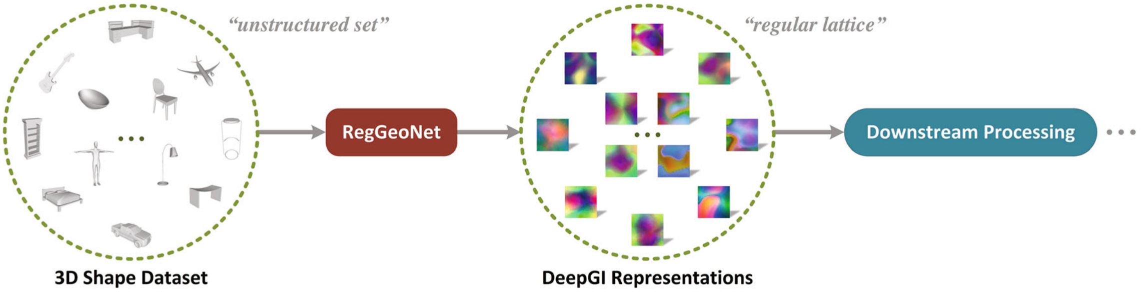

Fig. 2 Illustration of the proposed large-scale point cloud processing pipeline driven by the proposed regular geometry representation. Given an unstructured 3D point set, RegGeoNet converts it into a 2D lattice structure called DeepGI, where irregular surface points are captured in regular image pixels. As a generic 3D representation modality, all the downstream point cloud processing tasks can be directly performed on DeepGIs

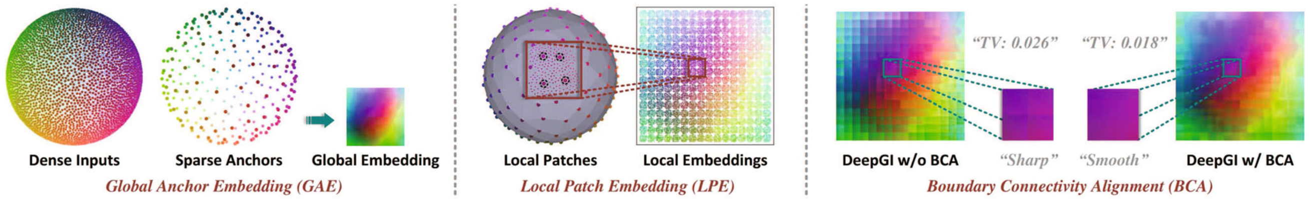

Fig. 3 A toy example for converting a set of uniformly sampled points on a unit sphere into DeepGI, where 3D points and 2D pixels are color-coded to indicate mappings. In the generated 2D DeepGI, the actual values for each 2D pixel are the corresponding 3D point coordinates

plify an input dense point cloud into a sparse set of "anchor" points, after which we explore a generative framework to create a global geometry parameterization in the form of 2D image lattice. In the LPE module, we construct local patches centered at every anchor point, and separately embed 3D patch points onto the pre-defined 2D lattice space. By assembling the global and local embeddings, we can generate a complete DeepGI structure for the whole input point cloud. Finally, we design a boundary connectivity alignment (BCA) module, serving as a post-refinement procedure, to enhance spatial continuity between adjacent local patch parameterizations. As an unsupervised learning framework, each core component of the proposed RegGeoNet has clear geometric motivation and is functionally independent, which decomposes the complex geometry parameterization task into several sub-problems that are easier to solve. Moreover, motivated by the unique structural characteristics of the proposed DeepGI representation modality, we investigate different possibilities of deep feature extraction, and further customize a set of powerful and efficient learning operators that directly operate on DeepGIs.

To validate the superiority of the overall learning framework driven by regular geometry representation, we conduct extensive evaluations on in varieties of downstream large-scale point cloud processing scenarios, including shape classification, object part segmentation, scene semantic segmentation, normal estimation, and geometry compression, where our frameworks achieve highly competitive performance, compared with the current state-of-the-art learning approaches in the fields.

In summary, this paper mainly makes the following contributions:

(1) We propose a new modality called DeepGI for representing large-scale 3D point clouds as 2D images with regular geometry.
(2) We design an unsupervised neural architecture called RegGeoNet to create DeepGIs from unstructured point clouds with arbitrary geometry and topology.
(3) We present an efficient large-scale point cloud learning framework that directly operates on DeepGIs, which achieves state-of-the-art performances in various tasks.
(4) We reveal the potential of exploiting large-scale point cloud data, which is ignored by previous studies that only focus on sparse inputs.

The remainder of this paper is organized as follows. In Sect. 2, we review previous studies that are closely related to the general scope of our work. In Sect. 3, we introduce specific techniques involved in the proposed RegGeoNet for regular point cloud geometry parameterization. In Sect. 4, we discuss different possibilities for extracting deep features directly from the proposed DeepGI representation structures. We document the implementation details, experimental settings, and comparative results in Sect. 5. We discuss the main features and potentials of DeepGI and RegGeoNet, and point out open issues in Sect. 6 for future exploration. In the end, we conclude this paper in Sect. 7.

2 Related Work

In this section, we start by reviewing common deep learning-based 3D shape analysis frameworks, including rasterization-based (Sect. 2.1) and point-based (Sect. 2.2) learning approaches. Besides, since the core idea in this paper is to create regular 2D geometry parameterizations for 3D point clouds, we particularly discuss parameterization-based learning pipelines in Sect. 2.3, whose processing objects are not confined to point clouds but can involve more diverse types such as polygon meshes. Moreover, there also exists a limited number of studies focusing on using large-scale point clouds, which are summarized in Sect. 2.4.

2.1 Rasterization-Based Learning Approaches

One of the most straightforward schemes to overcome irregularity and unstructuredness of point clouds is to transform raw point sets into regular representation modalities through rasterization procedures, before feeding them into the subsequent processing pipelines.

Voxelization has been widely applied to produce volumetric grids for generic 3D shape representation, in which discrete spatial points are quantized into densely and uniformly distributed 3D occupancy grids. As investigated in previous works (Maturana and Scherer, 2015; Qi et al., 2016; Wu et al., 2015), this voxel-based learning paradigm naturally supports standard 3D convolutional architectures. Unfortunately, these approaches are not suitable for consuming high-resolution volumetric data due to cubic growth of computational complexity and memory footprint. This limitation leads to significant information loss during shape modeling, since geometric details can not be preserved in low-resolution voxels. Despite the later efforts in enhancing computational efficiency (Riegler et al., 2017; Wang et al., 2017), there still exists a performance trade-off, which needs to be carefully balanced in practical applications.

Another common alternative is multi-view projection, in which a single 3D object from multiple viewpoints is projected onto 2D planes, leading to a collection of multi-view images (Gojcic et al., 2020; Kalogerakis et al., 2017; Kanezaki et al., 2018; Su et al., 2015; Yu et al., 2018). This learning paradigm enables to introduce mature 2D deep convolutional networks pretrained on large-scale annotated image databases (Krizhevsky et al., 2012) for discriminative shape recognition. However, despite its leading performance in classification and retrieval tasks, it is highly non-trivial to extend such view-based approaches to fine-grained (point-wise) prediction/labeling application scenarios, such as semantic segmentation and normal estimation. In fact, DeepGI also serves as a 2D image representation structure. In contrast to rendered multi-view images of 2D projections, however, geometry information is explicitly encoded in the DeepGI structure, making it a generic 3D representation modality.

2.2 Point-Based Learning Approaches

Currently, point-based learning networks are attracting major attention in the research community. This learning paradigm gets rid of cumbersome pre-processing procedures and can directly operate on point sets, making it possible to capture fine-grained spatial structures naturally.

The first deep set architecture PointNet (Qi et al., 2017a) proposes to learn point-wise embeddings using shared multi-layer perceptrons (MLPs) and apply channel max-pooling to obtain the global shape descriptor in a permutation-invariant fashion. Inspired by conventional CNN architectures, PointNet++ (Qi et al., 2017b) integrates local geometry modeling by aggregating features from spatial neighbors and further introduces farthest point sampling (FPS) to achieve hierarchical feature abstraction. These two pioneering works cast profound influences on follow-up studies in terms of defining "convolution-like" operators on unstructured point sets.

PWCNN (Hua et al., 2018) waives the assumption that points should be unordered, and partitions spatial neighbors into kernel cells for convolving with weights. PointCNN (Li et al., 2018b) uses learned transformation matrices to adaptively specify local convolutional orders. FeaStNet (Verma et al., 2018) establishes soft correspondence between filter weights and graph nodes. SO-Net (Li et al., 2018a) builds the two-dimensional self-organizing map to explicitly model the spatial distribution of given points. SPLATNet (Su et al., 2018) performs sparse bilateral convolution by projecting point features into high-dimensional lattice. PointConv (Wu et al., 2019) formulates convolutional kernels by approximating continuous weight and density functions conditioned on local coordinates. RS-CNN (Liu et al., 2019c) exploits geometric relations among points to achieve explicit reasoning about spatial layouts. DGCNN (Wang et al., 2019) is built upon learnable relations and dynamically aggregates information from feature-level neighbors under global context. ShellNet (Zhang et al., 2019) collects statistics from concentric spherical shells, which allows structured feature fusion across different levels of receptive fields. PAT (Yang et al., 2019) implements adaptive down-scaling based on Gumbel-Softmax (Jang et al., 2017; Maddison et al., 2017). PointASNL (Yan et al., 2020) chooses a different implementation of differentiable adaptive sampling that fine-tunes the initial uniformly sampled points with learned shifts. KPConv (Thomas et al., 2019) learns a spatially deformable point cloud convolution with weights located in Euclidean space based on kernel points, which is more robust to varying densities. GDANet (Xu et al., 2021b) dynamically disentangles input points into contour and flat parts to capture holistic and complementary geometric semantics. PAConv (Xu et al., 2021a) constructs dynamic convolutional kernels by adaptively learning to assemble weight matrices in a predefined weight bank. CurveNet (Xiang et al., 2021) focuses on long-range feature modeling by grouping and aggregating sequences of points (i.e., curves). Particularly, FPConv (Lin et al., 2020) introduces a surface-style convolutional operator that directly works on the underlying surface geometry of point clouds, where local flattening is achieved by learning weight maps that softly project surrounding points onto 2D grids. Different from our work focusing on the big picture of regular geometry representation modality for irregular point cloud data, FPConv is still limited as an implicit local aggregation mechanism that operates in the feature space.

Conclusively, despite the great efforts in investigating various "convolution-like" operators defined on irregular point sets, these methods are still computationally inefficient due to expensive data structurization, which can be avoided on regular DeepGI structure.

2.3 Parameterization-Based Learning Approaches

In the geometry processing community, there exists a family of intrinsic representation and learning approaches, in which 3D surface geometries can be encoded into a series of local parameterizations (Boscaini et al., 2016; Masci et al., 2015; Monti et al., 2017) or a complete global parameterization (Haim et al., 2019; Maron et al., 2017; Sinha et al., 2016, 2017). It is worth noting that these methods are particularly developed for 3D mesh models, instead of point clouds, and thus beyond the scope of this paper. Nevertheless, these efforts lay the foundation for our exploration in creating regular geometry representation directly from point clouds, and hence are also included for completeness.

More specifically, GCNN (Masci et al., 2015) constructs geodesic polar coordinates from the local patch around each point, where convolutions can be naturally extended to non-Euclidean manifolds. This learning paradigm was further improved by introducing anisotropic diffusion (Boscaini et al., 2016) and Gaussian mixture (Monti et al., 2017) kernels as patch operators. Considering that local parameterizations fail to incorporate global contextual information, Sinha et al. (2016) globally computed spherical parameterization to create a regular 2D geometry image (GI) (Gu et al., 2002) from a given 3D mesh model, after which standard 2D CNNs can be directly used for deep feature extraction. SurfNet (Sinha et al., 2017) attempts to generate consistent GIs from a category of 3D shapes by solving a large-scale correspondence problem, and further applies GIs to generative surface modeling. Maron et al. (2017) investigated a global seamless parameterization defined on a flat torus. Haim et al. (2019) introduced a broad family of low-distortion surface-to-image representation based on a covering map. However, since computing global parameterization for complex topology and/or geometry can be difficult, these approaches only deal with clean, single-component, and manifold polygonal meshes. Additional pre-processing procedures are needed to repair erroneous manifolds and transform high-genus mesh models into sphere (genus-zero) topologies. GWCNN (Ezuz et al., 2017) presents a parametric and differentiable metric alignment layer amenable to gradient-based optimization, which can create regular representations for richer types of geometric data like meshes, point clouds, and general graphs. However, this method relies on prior knowledge for choosing appropriate shape descriptors. In practice, it is sensitive to topological noises and suffers from training difficulties due to quadratic parameter growth.

2.4 Deep Learning for Large-Scale Point Clouds

There exists a limited number of studies dealing with large-scale point cloud learning. SPG (Landrieu and Simonovsky, 2018) partitions an entire point cloud into geometrically simple parts and constructs a superpoint graph to learn contextual information. However, the actual geometric partition and graph construction processes are computationally intensive. TangentConv (Tatarchenko et al., 2018) projects local surface geometry onto precomputed tangent planes and apply planar convolutions. This strategy relies on heavy calculations for normal estimation, which can be sensitive to noises. FCPN (Rethage et al., 2018) combines point-based and voxel-based aggregation operators to implement efficient processing frameworks. A specialized scene flow estimation framework can be found in HPLFlowNet (Gu et al., 2019). RandLA-Net (Hu et al., 2020) particularly explores the potential of simple random sampling to build remarkably efficient hierarchical feature abstraction pipelines. In contrast to the existing studies that focus on enhancing specific network architectures, we are interested in exploring a regular geometric data representation structure, where we can omit expensive data structurization when conducting downstream processing.

3 Proposed Method

As a generic geometry representation modality, the proposed DeepGIs serve as permanent records of 3D contents. Given a shape dataset, we pre-convert all point cloud models into DeepGI structures for storage, and all downstream applications are directly conducted on DeepGIs. In what follows, we first provide an overview of our DeepGI representation

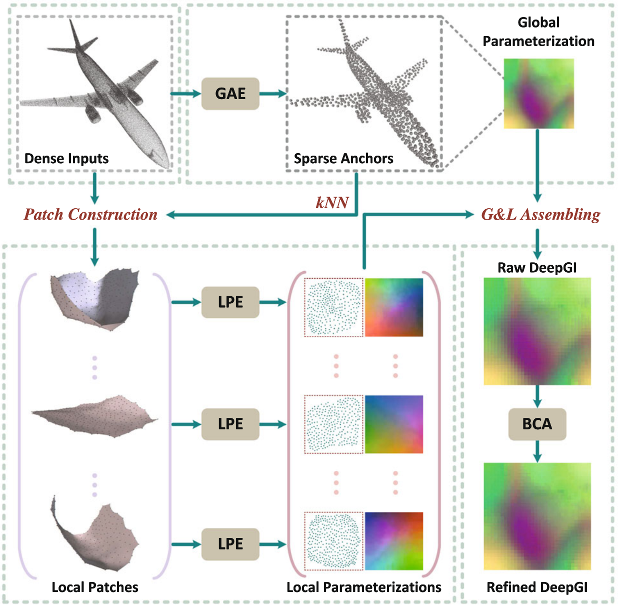

Fig. 4 Overall workflow of RegGeoNet for converting large-scale 3D point clouds into DeepGIs under a global-to-local surface parameterization framework. In the GAE module, we generate a global parameterization by embedding a sparse set of uniformly sampled anchor points onto a coarse 2D lattice. In the LPE module, we separately generate local parameterizations for patches centered at previously parameterized anchor points. Global and local parameterizations can be directly assembled to form a raw DeepGI, which is further fed into the BCA module for iterative refinement of topological consistency between adjacent local patch parameterizations

framework in Sect. 3.1, then sequentially introduce technical details of core modules in Sects. 3.2, 3.3, and 3.4.

3.1 Problem Overview

Given an unstructured 3D point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$, we aim at creating a three-channel image representation structure $\mathbf{I} \in \mathbb{R}^{m\times m\times 3}$ dubbed as DeepGI, whose $M = m \times m$ pixel values correspond to the original 3D spatial coordinates. Intuitively, we expect that local neighborhood consistency can be effectively maintained between the three-dimensional shape space and the two-dimensional grid space. This implies that spatially-adjacent 3D points are supposed to be accordingly parameterized onto neighboring 2D pixel locations, yielding a smooth geometry image whose pixel values (*i.e.*, 3D spatial coordinates) can be visualized as a continuous color distribution.

To generate 2D DeepGIs from 3D point clouds, we decompose the complicated underlying problem of surface parameterization into several independent sub-problems that can be separately solved by specialized sub-networks. Specifically, we investigate a global-to-local processing pipeline for generating regular geometry embedding and an iterative boundary alignment mechanism for enhancing topological consistency. As shown in Fig. 4, the proposed RegGeoNet correspondingly consists of three core modules: (1) *Global Anchor Embedding* (GAE) for generating a global parameterization of downsampled sparse anchors that coarsely depict shape geometry; (2) *Local Patch Embedding* (LPE) for separately generating a series of local parameterizations of all patches centered at anchor positions; and (3) *Boundary Connectivity Alignment* (BCA) for enhancing spatial continuity at adjacent patch boundaries, thus reducing blocking effect in the complete DeepGI structure.

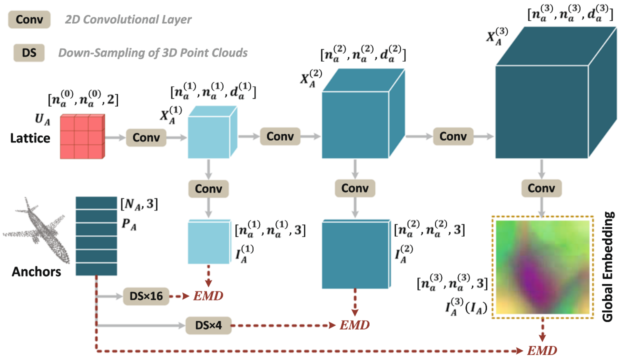
Conv
2D Convolutional Layer
DS
Down-Sampling of 3D Point Clouds
U_A
Lattice
$[n_a^{(0)}, n_a^{(0)}, 2]$
Conv
$X_A^{(1)}$
$[n_a^{(1)}, n_a^{(1)}, d_a^{(1)}]$
Conv
$X_A^{(2)}$
$[n_a^{(2)}, n_a^{(2)}, d_a^{(2)}]$
Conv
$X_A^{(3)}$
$[n_a^{(3)}, n_a^{(3)}, d_a^{(3)}]$
Conv
Conv
Conv
Anchors
$[N_A, 3]$
P_A
$[n_a^{(1)}, n_a^{(1)}, 3]$
$I_A^{(1)}$
$[n_a^{(2)}, n_a^{(2)}, 3]$
$I_A^{(2)}$
$[n_a^{(3)}, n_a^{(3)}, 3]$
$I_A^{(3)}(I_A)$
Global Embedding
DS×16
EMD
DS×4
EMD
EMD

Fig. 5 Illustration of the GAE module, where an initial 2D lattice is progressively embedded and decoded into different levels of image structures

Note that each of the three core modules (*i.e.*, GAE, LPE, and BCA) is designed as a fully unsupervised learning process with independent functionality and interpretable geometric motivation. Under such a decoupled problem formulation, we can conveniently customize different modeling schemes and optimization objectives for the corresponding sub-networks. Hence, we tend to divide the overall narrative in each of the following sections into (1) *design philosophy*, (2) *network structure*, and (3) *fitting/training strategy*, to facilitate the understanding of our methodology.

3.2 Global Anchor Embedding (GAE)

3.2.1 Design Philosophy

The goal of the GAE module is to globally create a regular lattice parameterization for a sparse set of anchor points uniformly downsampled from the dense input. Inheriting conventional learning paradigms in image generation and super-resolution, we explore a fully convolutional geometry embedding framework, in which an initial low-resolution lattice is progressively upscaled and decoded into different levels of high-resolution image representation structures, whose pixels are supposed to reconstruct spatial coordinates of the target anchor points.

The core intuition behind our design is that the underlying neighborhood aggregation mechanism in spatial convolution implicitly imposes strong regional smoothness and continuity constraints on the generated image pixel values (i.e., generated 3D spatial coordinates), especially under a super-resolution workflow with feature interpolation. Such an embedding strategy naturally ensures that spatially-adjacent points tend to be mapped to neighboring pixels on the image lattice.

3.2.2 Network Structure

As illustrated in Fig. 5, the first step in the GAE module is to uniformly downsample the original large-scale point set $\mathbf{P} \in \mathbb{R}^{N \times 3}$ into a sparse subset of anchor points $\mathbf{P}_A \in \mathbb{R}^{N_A \times 3}$ ($N_A \ll N$). For efficiency purposes, the process of simplifying $\mathbf{P}$ into $\mathbf{P}_A$ is implemented by two stages, i.e., starting by applying grid-subsampling to produce a simplified version of input $\mathbf{P}$, after which further applying farthest point sampling (FPS) to generate $\mathbf{P}_A$ containing the required number of points. Particularly, we make a minor modification in all experiments for the process of FPS by explicitly specifying the point that is closest to the coordinate origin as the starting point, instead of random selection, which can avoid being influenced by index permutation. At the output end, we expect to generate a three-channel image representation structure $\mathbf{I}_A \in \mathbb{R}^{n_a \times n_a \times 3}$ ($N_A = n_a \times n_a$), where the irregular 3D spatial points are placed onto the regular 2D lattice space.

More specifically, we pre-define a low-resolution lattice $\mathbf{U}_A \in \mathbb{R}^{n_a^{(0)} \times n_a^{(0)} \times 2}$, whose values are $N_A^{(0)} = n_a^{(0)} \times n_a^{(0)}$ discrete grid coordinates uniformly distributed in the unit square space $[0, 1]^2$. This can be be considered as a two-channel image structure. Hence, we can feed $\mathbf{U}_A$ into a specialized "image super-resolution" architecture consisting of r_a convolutional stages, each with 2× feature upscaling. This produces a series of feature maps $\{\mathbf{X}_A^{(i)} \in \mathbb{R}^{n_a^{(i)} \times n_a^{(i)} \times d_a^{(i)}}\}_{i=1}^{r_a}$, where $n_a^{(i)} = 2^i \times n_a^{(0)}$. Then, we adopt hierarchical feature decoding to generate different resolutions of regular image parameterizations, where each feature map $\mathbf{X}_A^{(i)}$ is fed into a separate set of convolutional layers to output a three-channel image structure $\mathbf{I}_A^{(i)} \in \mathbb{R}^{n_a^{(i)} \times n_a^{(i)} \times 3}$ containing $N_A^{(i)} = n_a^{(i)} \times n_a^{(i)}$ pixels. For simplicity, we denote $\mathbf{I}_A$ and n_a as the highest resolution versions, $\mathbf{I}_A^{(r_a)}$ and $n_a^{(r_a)}$, respectively.

3.2.3 Fitting Strategy

Here, the generated image pixels are fit to anchor points by optimizing network parameters to minimize a reconstruction loss between different levels of I_A and ground truth P_A. Specifically, treating image pixels as point coordinates naturally motivates us to supervise the reconstruction process by minimizing point set similarity. In our implementation, we introduce hierarchical supervision built upon earth mover's distance (EMD) (Fan et al., 2017), which is formulated as

$$\mathcal{L}_{rec}\left(\left\{\mathbf{I}_A^{(i)}\right\}; \left\{\mathbf{P}_A^{(i)}\right\}\right) = \sum_{i=1}^{r_a} \phi_{emd}\left(\mathbf{I}_A^{(i)}; \mathbf{P}_A^{(i)}\right), \tag{1}$$

where $\mathbf{P}_A^{(i)} \in \mathbb{R}^{N_A^{(i)} \times 3}$ is uniformly downsampled from $\mathbf{P}_A^{(r_a)}$ (identified with $\mathbf{P}_A$) serving as a coarser ground truth, and $\phi_{emd}(*; *)$ computes EMD between two point sets. By optimizing objective function $\mathcal{L}_{rec}$, we obtain a global parameterization $\mathbf{I}_A$ that nicely approximates anchor points in $\mathbf{P}_A$. Note that the network itself serves as the deep prior, where we separately optimize network parameters (*i.e.*, convolutional filters) to generate I_A for every single input model. See Sect. 5.1.2 for more training details.

3.3 Local Patch Embedding (LPE)

3.3.1 Design Philosophy

The goal of the LPE module is to separately create planar parameterizations for local patches centered at each of the global anchors. Different from the GAE module implemented as a generative framework driven by standard spatial convolutions, the LPE module is built upon shared MLPs conditioned on a unique patch signature, which tends to directly build a smooth point-wise mapping between 3D patch points and learned 2D planar embeddings.

Despite the fact that the GAE module is applicable to local patch parameterization, its working mechanism can be highly inefficient to handle a large number of patches. Compared with a complete shape, local patches covering small regions usually have much simpler geometric and topological structures, which implies that we can "unfold/flatten" the underlying 3D surface of target patches onto 2D planes in a geometrically-meaningful manner. Therefore, instead of separately fitting on every single patch, we can train the LPE module offline on a large amount of patch samples, and then apply the trained model to any given patch without cumbersome overfitting/fine-tuning.

3.3.2 Network Structure

As shown in Fig. 6, the LPE module takes as inputs a collection of local patches decomposed from the whole point set $\mathbf{P}$. Treating each global anchor $\mathbf{a}^{(i)} \in \mathbf{I}_A$ as patch centroid, we collect N'_L spatial neighbors through k-NN search, which totally deduces N_A local patches $\{\mathbf{P}_L^{(i)} \in \mathbb{R}^{N'_L \times 3}\}_{i=1}^{N_A}$ by querying for all anchor points $\{\mathbf{a}^{(i)}\}_{i=1}^{N_A}$. At the output end,

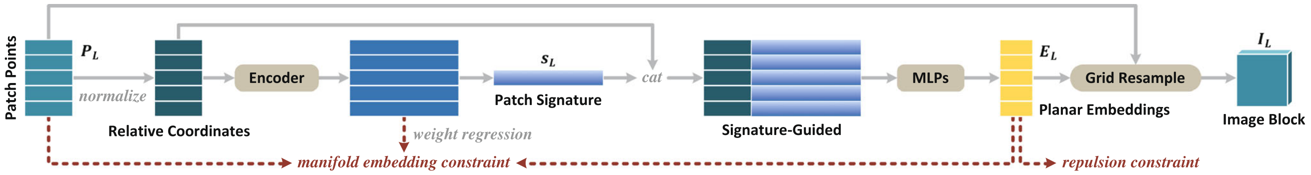

Fig. 6 Flowchart of the LPE module, where local 3D patches are unfolded into 2D planar embeddings and further resampled on lattice grids to generate regular image blocks

we expect to correspondingly generate N_A planar embedding sets $\{\mathbf{E}_L^{(i)} \in \mathbb{R}^{N'_L \times 2}\}_{i=1}^{N_A}$, where 3D points in $\mathbf{P}_L^{(i)}$ and 2D points in $\mathbf{E}_L^{(i)}$ form row-wise one-to-one mappings. To ensure spatial coverage, it is required that $N_A \times N'_L \geq N$ for redundant query of patch points. After that, we conduct grid resampling between $\mathbf{P}_L^{(i)}$ and $\mathbf{E}_L^{(i)}$ on pre-defined lattice grids to generate a collection of regular image structures $\{\mathbf{I}_L^{(i)} \in \mathbb{R}^{n_l \times n_l \times 3}\}_{i=1}^{N_A}$, such that the original N'_L spatial points in local patch $\mathbf{P}_L^{(i)}$ are parameterized into $N_L = n_l \times n_l$ image pixels. Without loss of generality, we remove superscript "(i)" in the subsequent notations, restricting our attention to a local patch $\mathbf{P}_L$ centered at an anchor point $\mathbf{a}$, which is mapped as planar embedding $\mathbf{E}_L$ to create regular parameterization $\mathbf{I}_L$.

More specifically, we perform patch encoding on a pre-normalized local patch through a deep set architecture (Qi et al., 2017a) followed by channel average pooling, which produces point-wise embedding representations and a vectorized feature signature denoted as $\mathbf{s}_L$. We then replicate and concatenate $\mathbf{s}_L$ with the normalized point coordinates, and deploy shared MLPs to generate point-wise planar embeddings within a unit square $[0, 1]^2$. The overall processing pipeline can be formally described as

$$\mathbf{E}_L = \varphi_2\left(\varphi_1([\bar{\mathbf{P}}_L; \mathbf{s}_L])\right), \tag{2}$$

where $\bar{\mathbf{P}}_L$ represents relative coordinates after normalizing $\mathbf{P}_L$ into a unit ball, $[*; *]$ represents channel concatenation, $\varphi_1(\cdot)$ and $\varphi_2(\cdot)$ denote non-linear transformation built upon shared MLPs. In our implementation, we employ the sigmoid activation function in the output layer to restrict values of the generated planar embeddings $\mathbf{E}_L$ within the range of $(0, 1)$.

3.3.3 Training Strategy

Benefiting from the implicit smoothness of such point-wise embedding operations, we can naturally learn a continuous 3D-to-2D mapping by imposing a "repulsion" constraint on the generated planar embedding points.

Given a certain embedding point $\mathbf{e}_j \in \mathbf{E}_L$, we search its nearest neighbor $\tilde{\mathbf{e}}_j$, and constrain that they can be separated by an appropriate distance. Formally, we can define

$$\mathcal{L}_{rep}(\mathbf{E}_L) = \sum_{j=1}^{N'_L} \max\left(0,\ \alpha \cdot \tau_u - \left\|\mathbf{e}_j - \tilde{\mathbf{e}}_j\right\|_2\right), \tag{3}$$

where $\tau_u = 1/(\sqrt{N'_L} - 1)$ generally denotes the minimum threshold for pulling apart nearest neighbors over a uniform $\sqrt{N'_L} \times \sqrt{N'_L}$ lattice. We empirically choose the relaxation factor as $\alpha = 0.5$, which works robustly in all experiments. Since we will re-scale the generated planar embedding points into $[0, 1]^2$ for making full use of the unit square space after the training phase, we can safely set small value for the fixed distance threshold.

Furthermore, inspired by the classic non-linear dimensionality reduction algorithm LLE (Roweis and Saul, 2000), we also add a manifold embedding constraint (MEC) to explicitly preserve neighborhood consistency between source $\mathbf{P}_L$ and target $\mathbf{E}_L$. Given a typical source point $\mathbf{l}_j \in \mathbf{P}_L$ and its planar embedding $\mathbf{e}_j \in \mathbf{E}_L$, we jointly minimize the following two reconstruction errors as

$$\mathcal{L}_{mec\_pat}(\mathbf{l}_j) = \left\|\mathbf{l}_j - \sum\nolimits_{k \in \mathcal{N}(\mathbf{l}_j)} (w_{jk} \cdot \mathbf{l}_k)\right\|_2, \tag{4}$$

and

$$\mathcal{L}_{mec\_ebd}(\mathbf{e}_j) = \left\|\mathbf{e}_j - \sum\nolimits_{k \in \mathcal{N}(\mathbf{l}_j)} (w_{jk} \cdot \mathbf{e}_k)\right\|_2, \tag{5}$$

where $\mathcal{N}(\mathbf{l}_j)$ is the index set that specifies K_L neighboring points around $\mathbf{l}_j$, and we deploy a separate learning branch built upon a three-layer shared MLP to adaptively regress from point-wise embeddings the corresponding linear combination weights $\mathbf{W}_L \in \mathbb{R}^{N'_L \times K_L}$, which satisfy $\sum_{k=1}^{K_L} w_{jk} = 1$ for $j = 1, \cdots, N'_L$. Thus, the complete manifold embedding constraint is given by

$$\mathcal{L}_{mec}(\mathbf{P}_L; \mathbf{E}_L) = \sum_{j=1}^{N'_L} \left[\mathcal{L}_{mec\_pat}(\mathbf{l}_j) + \mathcal{L}_{mec\_ebd}(\mathbf{e}_j)\right]. \tag{6}$$

Intuitively, we expect that the same set of locally linear combination weights $\mathbf{W}_L$ for 3D source points $\mathbf{P}_L$ can also

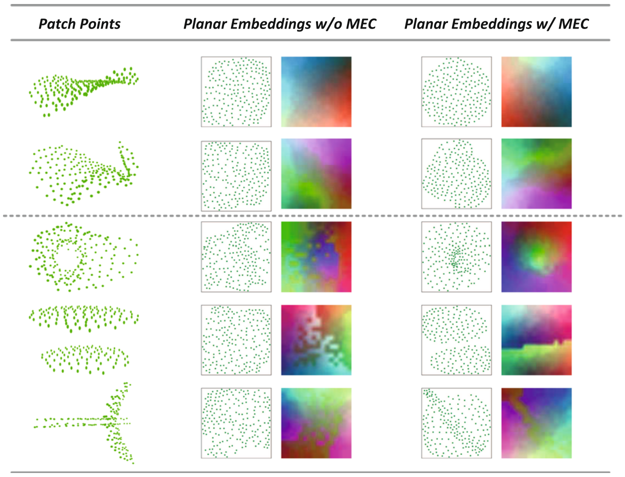

Fig. 7 Local patch embeddings generated by the LPE module with and without the manifold embedding constraint (MEC). Rows 1 and 2: patches with disk topology; Row 3: multiply connected domain; Row 4: patch with disconnected components; Row 5: non-manifold configuration

describe neighborhood distribution of their 2D embedding points $\mathbf{E}_L$. When combined with $\mathcal{L}_{rep}$ as the overall training objective of the LPE module, $\mathcal{L}_{mec}$ is multiplied by a scalar γ_{mec} and thus serves as an adjustable regularizer. The overall objective function is written as

$$\mathcal{L}_{lpe} = \mathcal{L}_{rep} + \gamma_{mec} \cdot \mathcal{L}_{mec}, \tag{7}$$

where γ_{mec} is initialized as 1 in the beginning and linearly decays to 0 with the training epoch. Intuitively, this means imposing a stronger constraint on neighborhood consistency in earlier training stages. As the training process goes on, the overall distribution of planar embedding points becomes stable, and thus we gradually pay more attention to the repulsion loss to ultimately reduce points clustering.

Figure 7 shows some typical examples of patch embeddings. Note that the LPE module is offline trained on a large dataset of patches, instead of separately fitting on every local patch of the given point cloud model. We refer the readers to Sect. 5.1.2 for more details.

3.3.4 Grid Resampling

To create regular parameterization $\mathbf{I}_L \in \mathbb{R}^{n_l \times n_l \times 3}$ in the form of image lattice, we perform grid resampling to redistribute patch points $\mathbf{P}_L \in \mathbb{R}^{N'_L \times 3}$ onto uniform grids based on planar embedding locations $\mathbf{E}_L \in \mathbb{R}^{N'_L \times 2}$.

To this end, we pre-define $n_l \times n_l$ uniform lattice grids in $[0, 1]^2$ denoted as $\mathbf{U}_L \in \mathbb{R}^{N_L \times 2}$, and build a Bipartite graph $\mathcal{G}_{ebd} = (\mathbf{E}_L, \mathbf{U}_L, \mathbf{V}_L)$ based on nearest neighbor matching, where graph edges $\mathbf{V}_L$ indicate the matching relationships between the N'_L planar embedding points in $\mathbf{E}_L$ and the N_L lattice grid positions in $\mathbf{U}_L$. After that, since $\mathbf{E}_L$ point-wisely corresponds to $\mathbf{P}_L$, we directly transfer the same matching relationships between $\{\mathbf{E}_L, \mathbf{U}_L\}$ to $\{\mathbf{P}_L, \mathbf{U}_L\}$, and accordingly obtain a Bipartite graph $\mathcal{G}_{pat} = (\mathbf{P}_L, \mathbf{U}_L, \mathbf{V}_L)$ that specifies a mapping between irregular patch points in $\mathbf{P}_L$ and regular grid positions in $\mathbf{U}_L$. We follow such mapping relationships to fill in all $n_l \times n_l$ lattice grids with patch points, which can deduce a regular patch parameterization $\mathbf{I}_L \in \mathbb{R}^{n_l \times n_l \times 3}$. Here, we do not require $N'_L = N_L$. Instead, we typically specify $N_L \geq N'_L$ to perform grid resampling in a redundant manner for reducing the loss of points.

Table 1 Notations of key concepts involved in the proposed method

Symbol	Modality	Dimension	Description
$\mathbf{P}$	3D point set	$[N, 3]$	Dense input
$\mathbf{P}_A$	3D point set	$[N_A, 3]$	Global anchors
$\{\mathbf{P}_L^{(i)}\}_{i=1}^{N_A}$	3D point sets	$N_A \times [N'_L, 3]$	Local patches
$\{\mathbf{E}_L^{(i)}\}_{i=1}^{N_A}$	2D point sets	$N_A \times [N'_L, 2]$	Planar ebd.
$\mathbf{I}_A$	2D image	$[n_a, n_a, 3]$	global para.
$\{\mathbf{I}_L^{(i)}\}_{i=1}^{N_A}$	2D images	$N_A \times [n_l, n_l, 3]$	Local para.
$\mathbf{I}$	2D image	$[m, m, 3]$	DeepGI

It is also worth noting that, in order to construct the regular lattice structure, grid resampling can be "softened" into neighborhood interpolation operators to avoid the repetitive pixel representation. However, this strategy destroys raw information of point-wise coordinates and may produce surface outliers when dealing with complex patches.

3.4 Boundary Connectivity Alignment (BCA)

Going back to our preceding notations with superscripts. In the GAE and LPE modules, we decompose the input large-scale point set $\mathbf{P} \in \mathbb{R}^{N \times 3}$ into a global anchor set $\mathbf{P}_A \in \mathbb{R}^{N_A \times 3}$ and multiple local patch sets $\{\mathbf{P}_L^{(i)} \in \mathbb{R}^{N'_L \times 3}\}_{i=1}^{N_A}$, which are correspondingly encoded in a global lattice parameterization $\mathbf{I}_A \in \mathbb{R}^{n_a \times n_a \times 3}$ and multiple local lattice parameterizations $\{\mathbf{I}_L^{(i)} \in \mathbb{R}^{n_l \times n_l \times 3}\}_{i=1}^{N_A}$. To ease reading, we summarize the notations of key concepts in Table 1.

Now, we can assemble global and local embeddings into a complete image representation structure by assigning square image block $\mathbf{I}_L^{(i)}$ onto the pixel location of its anchor point $\mathbf{a}^{(i)} \in \mathbf{I}_A$. This produces a raw DeepGI representation structure $\mathbf{I} \in \mathbb{R}^{m \times m \times 3}$ that encodes the original N spatial points in M image pixels. Apparently, we can deduce $M = N_A \times N_L$ ($m = n_a \times n_l$). Structurally, DeepGI is an $m \times m$ three-channel image locally composed of $n_a \times n_a$ square image blocks, each of which contains $n_l \times n_l$ grid pixels.

3.4.1 Design Philosophy

Unfortunately, although the proposed global-to-local processing pipeline meets the need of computational efficiency, it will inevitably cause discontinuities along boundaries between adjacent local blocks in the complete DeepGI. This is because local patches are *separately* parameterized into regular images without interaction. Since two adjacent 3D patches can be freely unfolded at arbitrary angles, their original boundary regions may not stay connected in the 2D embedded domain, which inevitably results in sharp block edges.

Such topological mismatch is unfavorable and causes inconvenience in downstream applications and learning tasks. Therefore, we investigate the BCA module to align local patch parameterizations. Intuitively, we adaptively rotate the current planar embedding points $\{\mathbf{E}_L^{(i)}\}_{i=1}^{N_A}$ around local block centers with a set of appropriate angles $\{\theta_i\}_{i=1}^{N_A}$. Despite the fact that there exists no optimal situation where distortion can be completely avoided, we expect that the updated local embeddings can enhance boundary connectivity of patch parameterizations, reducing blocking effect and generating a smoother DeepGI.

3.4.2 Network Structure

Technically, we design an iterative searching and updating mechanism by optimizing a deep convolutional network, as shown in Fig. 8, to generate a rotation map $\Theta \in \mathbb{R}^{n_a \times n_a}$. Briefly, taking the current DeepGI $\mathbf{I}$ as input, we adopt an $n_l \times n_l$ large kernel in the first convolutional layer with stride n_l to capture distribution patterns of local patch parameterizations and produce an $n_a \times n_a$ feature map, which is further embedded in the subsequent convolutional layers to generate a one-dimensional rotation map, whose values are within the range of $(0, 2\pi)$. At each iteration, we apply the learned Θ to rotate the corresponding patch embeddings and assemble an updated DeepGI. In our implementation, a differentiable image rotation operator given by Riba et al. (2020) is adopted to make the whole learning process trainable.

3.4.3 Fitting Strategy

We adopt total variation (TV), a widely used regularization for image smoothing/denoising purposes, as optimization objective, which can be formulated as

$$\mathcal{L}_{tv}(\mathbf{I}) = \frac{1}{M} \sum_{x=1}^{m} \sum_{y=1}^{m} \left(|\partial_x \mathbf{I}(x, y)| + \left|\partial_y \mathbf{I}(x, y)\right| \right), \tag{8}$$

where we approximate gradients for network updating using differences between neighboring pixels, as implemented in (Riba et al., 2020). Considering that patch rotation does not change relative pixel distributions inside local blocks, we can mask out pixels that are not located at block edges, so that we only conduct calculations on edge pixels, instead of the entire image. Note that the BCA module is separately applied on each DeepGI in an online manner.

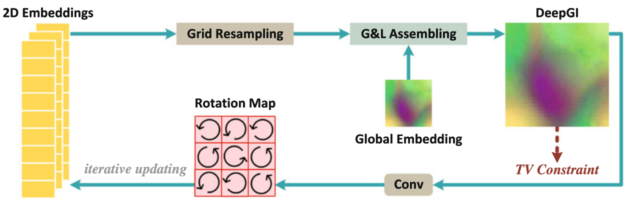

Fig. 8 Flowchart of the BCA module, where a rotation map is adaptively learned from the DeepGI to achieve boundary alignment through iterative updating

4 Deep Feature Extraction on DeepGIs

DeepGI serves as a generic representation modality for point clouds. Therefore, in addition to storage and transmission, we are also interested in learning deep features directly from DeepGIs for achieving downstream scenarios of point cloud processing and understanding.

4.1 Standard 2D Convolution on DeepGIs

The most straightforward scheme can be transferring mature deep convolutional neural architectures directly to consume DeepGIs as inputs. Although standard 2D convolution is known to be incompatible with irregular point set structure, it is applicable to the proposed regular DeepGI structure for the following reasons:

(1) Since pixel values of the created DeepGI are spatial coordinates of the source point cloud, feature representations extracted on 2D DeepGI describe the corresponding 3D surface, which means that 2D convolution operation on DeepGIs is analogous to convolution-like neighborhood aggregation on 3D point sets.

(2) Locality is one of the most basic and essential inductive biases of convolutional networks concealed in the learning paradigm of neighborhood information aggregation. In terms of DeepGI, pixels (*i.e.*, points) within the local block come from the same local surface and are geometrically "unfolded" onto regular grids. Therefore, applying the standard spatial convolution on DeepGI is equivalent to an operation on the local surfaces of the 3D point set. Moreover, spatial proximity also maintains across blocks, since adjacent blocks come from neighboring patches.

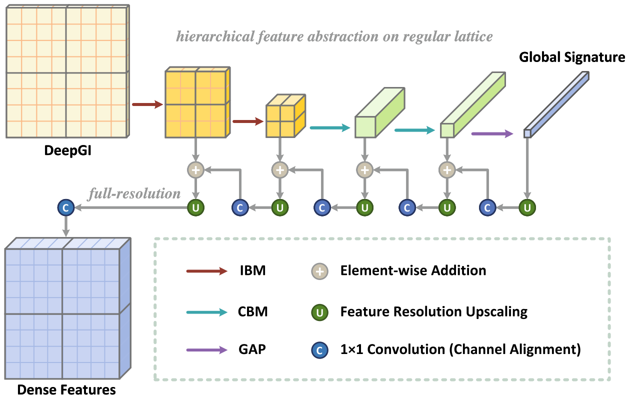
hierarchical feature abstraction on regular lattice
Global Signature
DeepGI
full-resolution
Dense Features
IBM
CBM
GAP
Element-wise Addition
Feature Resolution Upscaling
1×1 Convolution (Channel Alignment)
U
C

Fig. 9 Flowchart of our DeepGI-driven large-scale point feature extraction framework. We sequentially stack IBM and CBM layers, and obtain a global shape descriptor by GAP. Dense point-wise features can be generated by top-down feature propagation

(3) Hierarchical feature extraction becomes natural. Different from previous point-based networks (Qi et al., 2017b; Wu et al., 2019) that apply FPS to construct different levels of sparser point clouds, we can achieve this by common image pooling operators (*e.g.*, max-pooling, average-pooling).

(4) There is no need to explicitly conduct spatial matching for multi-scale feature fusion and propagation, since we can conveniently align feature maps with different sizes via appropriate downscaling or interpolation operations.

Empirically, we design different variants of convolutional architectures operating on DeepGIs to validate its compatibility with generic image analysis tools (see Sect. 5.2.2).

4.2 Customized Feature Extractor on DeepGI

Despite the potential of applying standard 2D convolution to DeepGI, we also notice the following unfavorable factors that may limit its learning capability:

(1) Spatial convolution is sensitive to different parameterization patterns. Since we cannot ensure that similar object surfaces correspond to consistent parameterizations, the search space of such a learning paradigm can be greatly enlarged, which may hurt network robustness.

(2) Since we typically adopt redundant grid resampling in the LPE module to enhance representation accuracy, there exist an arbitrary number of repetitive pixels in local blocks of DeepGI, which may disturb local aggregation.

(3) Spatial discontinuity between block edges cannot be eliminated, despite being smoothed by the BCA module. This may produce unstable outputs when convolutional kernels sliding across boundary pixels.

Therefore, here we propose a highly customized learning architecture for the proposed DeepGI representation modality. As illustrated in Fig. 9, we present a two-stage feature extraction workflow consisting of intra-block modeling (IBM) and cross-block modeling (CBM).

Generally, in the first IBM stage, each local block $\mathbf{I}_L^{(i)}$ is embedded into a one-dimensional feature vector $\mathbf{v}^{(i)} \in \mathbb{R}^{d_v}$ that encodes local geometry information around the corresponding patch centroid. This produces an $n_a \times n_a$ feature map denoted as $\mathbf{V} \in \mathbb{R}^{n_a \times n_a \times d_v}$. In the second CBM stage, we present a hierarchical graph convolution framework, where patch descriptors $\{\mathbf{v}^{(i)}\}_{i=1}^{N_a}$ are interacted and fused into high-level feature abstractions. In what follows, we detail these two stages.

4.2.1 Inner-Block Modeling (IBM)

To facilitate the mathematical formulation, we denote the input tensor of the i^{th} IBM unit as $\mathbf{T}_i \in \mathbb{R}^{m_i \times m_i \times d_i}$, where $\mathbf{T}_1$ is the given DeepGI $\mathbf{I} \in \mathbb{R}^{m \times m \times 3}$ at the first IBM unit. Then, the output feature map is denoted as $\mathbf{T}_{i+1} \in \mathbb{R}^{m_{i+1} \times m_{i+1} \times d_{i+1}}$, where $m_i = m_{i+1} \times \xi$ and ξ represents feature down-scaling ratio. Formally, we can describe the learning process of a single IBM unit as

$$\mathbf{T}_{i+1} = MP_\xi\left(Conv_{1\times1}\left([\mathbf{T}_i; \mathbf{T}_i - NI_\xi(AP_\xi(\mathbf{T}_i))]\right)\right), \quad (9)$$

where $AP_\xi(\cdot)$ and $MP_\xi(\cdot)$ represent average-pooling and max-pooling operators with both kernel size and stride set as ξ, $NI_\xi(\cdot)$ represents nearest interpolation operator with scale factor set as ξ, and $Conv_{1\times1}(\cdot)$ represents 1×1 convolution.

In our design, we slide an average-pooling kernel to produce “mean feature centroids”, which are further subtracted from the input features to form “relative features”. This step aims to achieve neighborhood association. We concatenate the original input features with the obtained relative features and use a 1×1 convolutional layer followed by max-pooling for feature fusion and resolution reduction.

In our experiments, we specify hyperparameter ξ in accordance with block size n_l to deduce an $n_a \times n_a$ feature map $\mathbf{V}$. We uniformly use two IBM units for modeling inner-block statistics by accordingly configuring ξ as $\sqrt{n_l}$.

4.2.2 Cross-Block Modeling (CBM)

The preceding IBM stage separately produces vectorized encodings for each of the N_a blocks. In the CBM stage, we introduce EdgeConv, a popular graph convolution operator (Wang et al., 2019), for dynamic interactions and relationship modeling across blocks under global shape context. This can be adapted without any technical modification. Still, different from the original processing pipeline, we additionally add a 2× max-pooling operation after each graph convolution layer, which achieves neighborhood aggregation in the spatial domain and significantly reduces computational complexity and memory footprint in deeper EdgeConv layers. By stacking several CBM units, we obtain a high-level feature map denoted as $\mathbf{F} \in \mathbb{R}^{n_f \times n_f \times d_f}$.

Thus, we construct a set of concise and efficient learning operators for large-scale point cloud learning, which neatly overcome the aforementioned limitations brought by 2D convolution, concretely:

(1) In the IBM stage, we only adopt 1 × 1 convolutions for feature embedding, which is insensitive to different parameterization patterns.
(2) When applying max-pooling for local aggregation, repetitive pixels are naturally ignored.
(3) Since we split the overall learning process into intra- and inter-block phases, we avoid aggregating statistics across block boundaries.

4.2.3 Task Networks

Following common practices in designing deep image classifiers, we apply global average pooling (GAP) to extract from $\mathbf{F}$ a vectorized global shape signature $\mathbf{f} \in \mathbb{R}^{d_f}$. By feeding it into the subsequent fully-connected layers, we can build a complete shape classification network. In our implementation, we apply GAP to intermediate feature maps and then concatenate multi-level signature vectors to form a global descriptor of the input model.

To produce dense features and achieve fine-grained tasks that require point-wise prediction/labeling, we consider multi-scale feature propagation for progressively integrating high-level features into low-level features. We conveniently resort to classic image segmentation frameworks (Ronneberger et al., 2015) to deploy deconvolutional layers and conduct top-down feature propagation for deducing a full-resolution feature map $\mathbf{Y} \in \mathbb{R}^{m \times m \times d_y}$. Specifically, the higher-level feature map is upscaled and fed into a 1 × 1 convolutional layer to align both the resolution and number of channels with the lower-level feature map, which are fused by element-wise addition for further propagation. Finally, to remove redundant features in $\mathbf{Y}$ that encode repetitive pixels, we map them back to input point cloud $\mathbf{P}$ based on nearest neighbor assignment and generate point-wise feature embeddings $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times d_y}$.

5 Experiments

We start by introducing our specific development protocols including benchmark datasets and implementation details in Sect. 5.1. In the following four subsections, we report performance on downstream large-scale 3D point cloud processing and understanding tasks: (1) shape classification (Sect. 5.2); (2) object and scene segmentation (Sect. 5.3); (3) normal estimation (Sect. 5.4); and (4) geometry compression (Sect. 5.5), which demonstrate the potential and superiority of our learning frameworks driven by DeepGI representation modality. Finally, we also validate the advantage of our frameworks for handling sparse point clouds in Sect. 5.6.

5.1 Development Protocols

Following common development protocols for benchmarking 3D point cloud learning frameworks, we conducted shape classification on both the ModelNet40 (Wu et al., 2015) and ScanObjectNN (Uy et al., 2019) datasets, object part segmentation on the ShapeNet-Part (Yi et al., 2016) dataset, indoor scene segmentation on the S3DIS (Armeni et al., 2016) dataset, and normal estimation on the ModelNet40 (Wu et al., 2015) dataset. For comparison with mesh parameterization-based learning schemes, we also included an additional human body segmentation task using the same dataset as collected in Maron et al. (2017), which we name the "M-HBS" dataset for convenience. Finally, we collected testing models from the MIT-Animation (Vlasic et al., 2008) dataset to validate the potential of DeepGI in large-scale point cloud geometry compression.

In the following, we provide the details of the above four benchmark datasets and our specific data preparation procedures in Sect. 5.1.1. After that, we introduce more implementation details in terms of building RegGeoNet and the subsequent deep feature extractors for achieving different downstream evaluation tasks in Sect. 5.1.2.

5.1.1 Benchmark Datasets

ModelNet40 Wu et al. (2015) is a large-scale object dataset of 3D CAD models, which totally provides 12311 mesh models covering 40 man-made categories. Under the official split, we have 9843 training models and 2468 testing models.

We adopted Poisson Disk Sampling (PDS) to uniformly discretize all mesh models into dense point clouds, each of which is composed of 100K spatial points. In the meantime, we can obtain point-wise normals, which are then normalized into unit vectors.

ScanObjectNN Uy et al. (2019) is a more recent and challenging real-world object dataset collected from scanned indoor scene data covering 15 categories, which only provides sparse point cloud models composed of 2048 uniform points each. Although our framework is customized for large-scale point clouds, it is still valuable to explore its effectiveness when dealing with real-scanned object data. Following common development protocols, we conducted experiments on its OBJ_ONLY (*w/o* background) and OBJ_BG (*w* background) settings, in which we have 2309 models for training and 581 for testing.

ShapeNet-Part Yi et al. (2016) is an annotated object dataset providing per-point semantic labels for 50 part categories from 16 object categories, including totally 16881 3D CAD models collected from the ShapeNet-Core (Chang et al., 2015) repository. Following common practices, we have 14007 shapes for training and the rest 2874 for testing.

We correspondingly collected the original mesh models from the ShapeNet-Core repository, and then mapped sparse per-point labels provided by Yi et al. (2016) to mesh faces. This allows us to discretize 100 K points with part annotations from labeled meshes using PDS.

M-HBS Maron et al. (2017) is a 3D surface segmentation dataset containing 370 training meshes and 18 testing meshes, whose faces are annotated according to 8 semantic partitions of human body. All meshes in this dataset are sphere-like models, since the corresponding approaches cannot deal with complex topologies.

Similar to the preceding processing procedures, we used PDS to resample 100 K densely-labeled points from each of the labeled meshes. To facilitate benchmarking other point-based learning approaches, we passed semantic labels from faces to vertices and thus constructed a sparse vertex set with part annotations, where the number of vertices ranges approximately from 6 to 12 K.

S3DIS Armeni et al. (2016) is a widely-used indoor scene dataset for semantic segmentation of large-scale colored point clouds, consisting of 271 medium-sized single rooms belonging to 6 areas. Following common development protocols as employed in Fan et al. (2021); Hu et al. (2020); Qiu et al. (2021), in our experiments, we sub-sampled raw point clouds and then cropped them into overlapping sub-regions each containing 65536 uniform points.

MIT-Animation Vlasic et al. (2008) provides high-quality 3D mesh models reconstructed from multi-view video recordings of diverse human motions. We selected a series of testing models and resampled 800 K points from each of the given meshes using PDS to conduct experiments on geometry compression of large-scale point clouds.

5.1.2 Implementation Details

The overall workflow of our regular geometry driven large-scale point cloud learning framework consists of two independent processing stages: (1) data modality transformation through RegGeoNet; (2) task-specific deep feature extraction.

For the three key modules of RegGeoNet, we collected a large amount of local patches for training the LPE module, and also designed appropriate initialization strategies for the GAE and BCA modules, which represent an iterative fitting process implemented by neural network optimization. Specifically, we constructed a patch dataset by cropping subregions from object models of the large-scale ShapeNet-Core dataset. When combined into the overall processing pipeline of RegGeoNet, the trained LPE module is directly applied to produce local patch parameterizations. For the GAE module, instead of using randomly initialized network parameters, we started by parameterizing N_A points uniformly sampled from a unit sphere for many iterations and used the resulting network weights as initialization, with the purpose of avoiding different parameterization results for the same input. For the BCA module, its warm-up procedure is similarly performed on a unit sphere, i.e., we applied the initialized GAE module and the trained LPE module to generate a raw DeepGI, which is fed into the BCA module for iterative refinement. The warm-up process of the GAE/BCA module can be finished within several minutes, and the training time of the LPE module is about two hours.

Configurations of DeepGI Generation Considering the specific task characteristics and the different number of input points, we correspondingly specified appropriate configurations for the generation of DeepGIs with balance of both representation efficiency and computational cost.

For shape classification and normal estimation experiments on ModelNet40, we set $\{N_A = 1024, N'_L = 200, N_L = 256\}$. For object part segmentation experiments on ShapeNet-Part and M-HBS, we set $\{N_A = 1600, N'_L = 64, N_L = 81\}$. For the indoor scene segmentation experiment on S3DIS, we set $\{N_A = 1024, N'_L = 100, N_L = 144\}$. For the ScanObjectNN dataset where input points clouds are far from dense, we set $\{N_A = 256, N'_L = 16, N_L = 25\}$. For the point cloud geometry compression experiment, we set $\{N_A = 256, N'_L = 3600, N_L = 4096\}$. According to specific data characteristics and task properties, we followed some intuitive principles to configure the above hyperparameters. For example, since segmentation is a more fine-grained and localized geometry understanding task compared with classification, we used larger N_A and smaller N_L in the part segmentation experiment to enhance the semantic consistency within each local patch. For geometry compression, considering that in the refined DeepGI there still exists discontinuity across adjacent image blocks, which has a negative impact on compression efficiency, we changed to use smaller N_A and larger N_L.

Figure 10 exhibits a gallery of typical 2D DeepGI representations for diverse 3D object models from multiple shape datasets involved in our experiments.

5.2 Shape Classification

Shape classification on ModelNet40 is one of the most popular benchmarking scenarios for point cloud learning architectures. In our implementation, we followed common data augmentation practices, such as random translation,

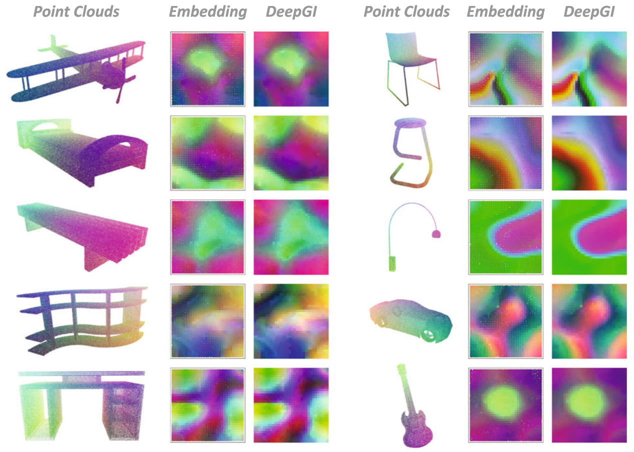

Fig. 10 Visualization of DeepGIs generated from a rich variety of object categories

Table 2 Comparisons with point-based learning networks in terms of overall accuracy (OA) on ModelNet40 with different input types denoted as P (points) and N (normals)

Method	Input	OA (%)
PointNet (Qi et al., 2017a)	P; 1 K	89.2
PointNet++ (Qi et al., 2017b)	P + N; 5 K	91.9
SpecGCN (Wang et al., 2018)	P + N; 2 K	92.1
PointWeb (Jiang et al., 2019)	P + N; 1 K	92.3
PCNN (Atzmon et al., 2018)	P; 1 K	92.3
SpiderCNN (Xu et al., 2018)	P + N; 1 K	92.4
PointConv (Wu et al., 2019)	P + N; 1 K	92.5
FPConv (Lin et al., 2020)	P + N; 2 K	92.5
KPConv (Thomas et al., 2019)	P; 7 K	92.9
DGCNN (Wang et al., 2019)	P; 1 K	92.9
RS-CNN (Liu et al., 2019c)	P; 1 K	92.9
InterpCNN (Mao et al., 2019)	P; 1 K	93.0
ShellNet (Zhang et al., 2019)	P; 1 K	93.1
Grid-GCN (Xu et al., 2020)	P; 1 K	93.1
PCT (Guo et al., 2020)	P; 1 K	93.2
PosPool (Liu et al., 2020)	P; 5 K	93.2
DensePoint (Liu et al., 2019b)	P; 1 K	93.2
SO-Net (Li et al., 2018a)	P + N; 5 K	93.4
GDANet (Xu et al., 2021b)	P; 1 K	93.4
PAConv (Xu et al., 2021a)	P; 1 K	93.6
CurveNet (Xiang et al., 2021)	P; 1 K	93.8
RegGeoNet-Cls	DeepGI (P; 100 K)	**95.2**

The best result is highlighted in bold

gravity-axis rotation, coordinate jittering, and re-scaling (isotropic, anisotropic), to enhance model generality and robustness. Different from many previous frameworks (*e.g.*, PointNet, PointNet++, RS-CNN, KPConv, DensePoint, and PosPool), we did not include any voting techniques in the testing phase to boost performance at the great cost of actual inference efficiency. Besides, we also conducted experiments on ScanObjectNN, which only provides sparse point clouds, to validate our generalization ability in real-scanned object data.

5.2.1 Comparisons with Point-Based Methods

We compared the proposed shape classification framework called RegGeoNet-Cls with state-of-the-art point-based learning approaches, as summarized in Tables 2 and 3.

On the conventional ModelNet40 dataset, despite the great efforts made by the point cloud community to explore various complex and specialized modeling schemes, the current state-of-the-arts are restricted to around 93% accuracy. Comparatively, our learning framework achieves significant performance gains and reaches 95.2% overall accuracy. On the more challenging ScanObjectNN dataset, our method still achieves highly competitive performance, compared with the current state-of-the-art methods.

5.2.2 Comparisons with Parameterization-Based Methods

We further compared our learning framework with other closely-related surface-style modeling schemes, i.e., GWCNN (Ezuz et al., 2017), DLGI (Sinha et al., 2016), and SNGC (Haim et al., 2019), which use input types of global parameterizations computed from repaired sphere-type (genus-zero) manifold meshes. As for the subsequent deep feature extractors, GWCNN (Ezuz et al., 2017) and DLGI (Sinha et al., 2016) deploy a specialized CNN classifier, while SNGC (Haim et al., 2019) adopts the off-the-shelf *Inception-V3* (Szegedy et al., 2016) architecture with modifications. In the preceding Sect. 4.1, we have intuitively analyzed the viability of applying 2D convolutions to DeepGIs for deep feature extraction. Here, we investigated several simple convolutional baselines to empirically verify our observations.

Design of convolutional baselines Specifically, we constructed *"CB_3X3"* consisting of 6 layers of 3 × 3 convolutions, each with 2× max-pooling. We set feature channels as {16, 32, 64, 64, 128, 1024}. This baseline was also applied to raw DeepGIs without the BCA refinement, from which we can observe negative influences of boundary discontinuity across local blocks. Then, we modified it into *"CB_16X16_3X3"* by replacing the first four 3 × 3 layers with a single convolution with kernel size 16 × 16 and stride 16. Thus, we avoided filtering across block edges, and aggregated the whole block statistics via a single spatial convolution. Besides, we created an even simpler baseline *"CB_1X1_3X3"* by applying two 1 × 1 convolutional layers followed by 16× max-pooling for block modeling.

As reported in Table 4, after boundary alignment by the BCA module, the *"CB_3X3"* baseline gains 0.8% accuracy improvement and obviously outperforms GWCNN (Ezuz

Table 3 Comparisons of point-based networks on OBJ_ONLY and OBJ_BG settings of ScanObjectNN under the measurement of overall accuracy (%)

Method	Input	OBJ_ONLY	OBJ_BG
PointNet (Qi et al., 2017a)	P; 1 K	79.2	73.3
SpiderCNN (Xu et al., 2018)	P; 1 K	79.5	77.1
PointNet++ (Qi et al., 2017b)	P; 1 K	84.3	82.3
PointCNN (Li et al., 2018b)	P; 1 K	85.5	86.1
DGCNN (Wang et al., 2019)	P; 1 K	86.2	82.8
GDANet (Xu et al., 2021b)	P; 1 K	88.0	**87.2**
RegGeoNet-Cls	DeepGI (P; 2 K)	**88.3**	86.9

The best results are highlighted in bold

et al., 2017) and DLGI (Sinha et al., 2016), which reveals the necessity of enhancing global consistency. However, it still under-performs SNGC (Haim et al., 2019) designed to create more consistent surface-to-plane representations. *"CB_16X16_3X3"* reaches higher accuracy of 91.9%, benefiting from richer augmentation and separate inner-block aggregation. *"CB_1X1_3X3"* shows another 0.5% improvement by switching to point-wise inner-block embedding, which is insensitive to different parameterization patterns within blocks.

5.3 Object and Scene Segmentation

We evaluated the proposed learning framework (RegGeoNet-Seg) for point-wise semantic labeling to reveal its potential in fine-grained visual tasks. Following common practices for benchmarking point-based networks, we experimented with generic object part segmentation on ShapeNet-Part. Previous works adopted 2048 points uniformly sampled from meshes, some of which, e.g., PointNet++, SO-Net (Li et al., 2018a), and SpiderCNN (Xu et al., 2018), further exploited normals to enrich input information and boost performance. We adopted classic *class-average* and *instance-average* mean intersection-over-union (cmIoU and imIoU) as quantitative metrics. During testing, we mapped dense labels to the same 2048 points as consumed by competing methods for unified measurement. As listed in Table 5, our learning framework achieves obvious performance gains with highly competitive segmentation accuracy on all 16 object categories. Figure 11 also visualizes some typical segmentation results.

Table 4 Comparisons with surface-style learning approaches in terms of overall accuracy (OA) on ModelNet40 with global parameterizations (G.S.) computed from meshes as network inputs

Method	Input	OA (%)
GWCNN (Ezuz et al., 2017)	Mesh G.S.	74.6
DLGI (Sinha et al., 2016)	Mesh G.S.	83.9
SNGC (Haim et al., 2019)	Mesh G.S.	91.6
CB_3X3	DeepGI (*w/o* BCA)	89.0
CB_3X3	DeepGI	89.8
CB_16X16_3X3	DeepGI	91.9
CB_1X1_3X3	DeepGI	92.4

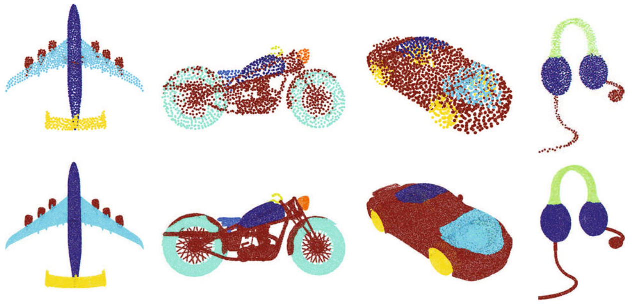

Fig. 11 Visualizations of object part segmentation. Row 1: sparse ground truths; Row 2: our dense predictions

In addition, we experimented with human body segmentation on M-HBS for non-rigid shape analysis. Following (Haim et al., 2019; Maron et al., 2017), we measured segmentation accuracy as the ratio of correctly labeled faces weighted by triangle areas. During testing, we correspondingly mapped point-wise labels to mesh faces to unify measurement. As shown in Table 6, Toric-CNN (Maron et al., 2017) and SNGC (Haim et al., 2019) directly work on repaired meshes. Besides, we included three point-based networks (Qi et al., 2017a, b; Wang et al., 2019), in which sparse vertices of meshes were treated as inputs. Again, our method achieves higher accuracy against both point-based and parameterization-based methods with large margins. Figure 12 visualizes some typical segmentation results.

For indoor scene segmentation on S3DIS, existing methods (Fan et al., 2021; Hu et al., 2020; Qiu et al., 2021) perform both training and inference on cropped sub-regions, each of which contains 40960 points. We adopted commonly-used mean class accuracy (mAcc) and mean intersection-over-union (mIoU) as quantitative metrics. As reported in Table 7, our method still shows satisfactory performance, compared with the current state-of-the-art frameworks that are particularly specialized for the specific task of large-scale point cloud segmentation, while our feature extraction pipeline is

Table 5 Comparisons of generic object part segmentation performance of different methods on ShapeNet-Part

Method	cmIoU	imIoU	ap.	Bag	Cap	Car	Chair	ep.	Guitar	Knife	Lamp	Laptop	mb.	Mug	Pistol	Rocket	sb.	Table
PointNet Qi et al. (2017a)	80.4	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++ Qi et al. (2017b)	81.9	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
SO-Net Li et al. (2018a)	81.0	84.9	82.8	77.8	88.0	77.3	90.6	73.5	90.7	83.9	82.8	94.8	69.1	94.2	80.9	53.1	72.9	83.0
DGCNN Wang et al. (2019)	82.3	85.1	84.2	83.7	84.4	77.1	90.9	78.5	91.5	87.3	82.9	96.0	67.8	93.3	82.6	59.7	75.5	82.0
SpiderCNN Xu et al. (2018)	82.4	85.3	83.5	81.0	87.2	77.5	90.7	76.8	91.1	87.3	83.3	95.8	70.2	93.5	82.7	59.7	75.8	82.8
RS-CNN Liu et al. (2019c)	84.0	86.2	83.5	84.8	88.8	79.6	91.2	81.1	91.6	88.4	86.0	96.0	73.7	94.1	83.4	60.5	77.7	83.6
PointCNN Li et al. (2018b)	84.6	86.1	84.1	86.5	86.0	80.8	90.6	79.7	92.3	88.4	85.3	96.1	77.2	95.3	84.2	64.2	80.0	83.0
PCT Guo et al. (2020)	83.1	86.4	85.0	82.4	89.0	81.2	91.9	71.5	91.3	88.1	86.3	95.8	64.6	95.8	83.6	62.2	77.6	83.7
SPH3D-GCN Lei et al. (2020)	84.9	86.8	84.4	86.2	89.2	81.2	91.5	77.4	92.5	88.2	85.7	96.7	78.6	95.6	84.7	63.9	78.5	84.0
PartNet Yu et al. (2019)	84.7	87.4	87.8	86.7	89.7	80.5	91.9	75.7	91.8	85.9	83.6	**97.0**	74.6	**97.3**	83.6	64.6	78.4	85.8
CurveNet Xiang et al. (2021)	83.7	86.6	85.2	86.1	89.8	80.9	92.0	73.4	91.9	88.0	84.8	96.0	74.3	95.2	82.2	59.8	76.6	83.8
GDANet (Xu et al. (2021b))	85.0	86.5	84.2	**88.0**	**90.6**	80.2	90.7	**82.0**	91.9	88.5	82.7	96.1	75.8	95.7	83.9	62.9	**83.1**	84.4
RegGeoNet-Seg	**86.3**	**88.6**	**89.0**	86.3	89.5	**82.1**	**92.8**	80.6	**92.6**	**90.2**	**86.9**	96.8	**80.0**	97.1	**85.5**	**65.9**	79.8	**86.1**

ap., *ep.*, *mb.*, and *sb.* are the abbreviations of *airplane*, *earphone*, *motorbike*, and *skateboard*, respectively. Note that the testing results of all the listed methods, except for GDANet (Xu et al., 2021b), were produced without voting. The best results are highlighted in bold

Table 6 Comparisons of human body segmentation performances on M-HBS with different input types denoted as M.F. (multiple features), Mesh G.S. (mesh-based global parameterizations), S.V. (sparse vertices), and P (points).

Method	Type and input size	Ratio (%)
Toric-CNN (Maron et al., 2017)	M.F.	88.0
SNGC (Haim et al., 2019)	Mesh G.S.	91.3
PointNet (Qi et al., 2017a)	S.V.; 6K ∼ 12K	87.7
DGCNN (Wang et al., 2019)	S.V.; 6K ∼ 12K	89.7
PointNet++ (Qi et al., 2017b)	S.V.; 6K ∼ 12K	90.8
RegGeoNet-Seg	DeepGI (P; 100K)	**93.2**

The best results are highlighted in bold

Table 7 Comparisons of different models on Area-5 of S3DIS (Armeni et al., 2016) for semantic segmentation of large-scale indoor point cloud scenes

Method	mAcc (%)	mIoU (%)
SPG (Landrieu and Simonovsky, 2018)	66.5	58.0
KPConv-*deform* (Thomas et al., 2019)	72.8	67.1
FPConv (Lin et al., 2020)	68.9	62.8
RandLA-Net (Hu et al., 2020)	71.5	62.5
SCF-Net (Fan et al., 2021)	–	63.4
BAAF-Net (Qiu et al., 2021)	73.1	65.4
Fast-PT (Park et al., 2022)	**76.5**	**68.5**
RegGeoNet-Seg	72.3	63.7

The best results are highlighted in bold

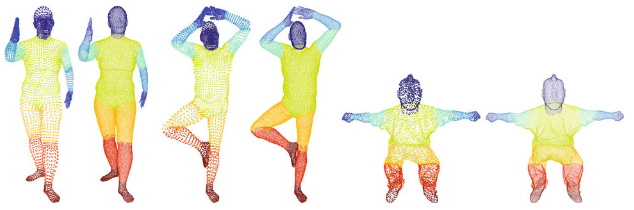

Fig. 12 Visualizations of human body segmentation. For each pair, the left shows ground truth labels of sparse vertices, and the right shows our dense predictions

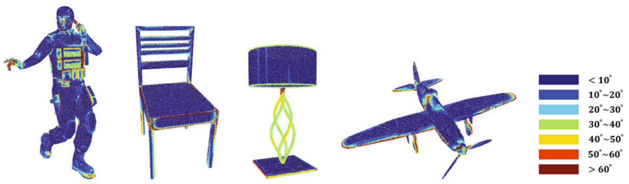

Fig. 13 Visualizations of normal estimation. We color-code points according to angle difference between predicted and ground truth normals

built upon common techniques and applied for rich types of data and tasks.

5.4 Normal Estimation

In addition to high-level visual recognition scenarios, we also explored a low-level geometry processing task, i.e., normal estimation on ModelNet40, which focuses on structural statistics of the underlying manifold surface, instead of abstract semantic cues.

Following common practices, we measure difference between regressed normals and ground truths by average cosine distance. During testing, we similarly mapped dense normals to the same 1024 points as consumed by competing methods for unified measurement. As reported in Table 8, compared with the best-performing competitor (*i.e.*, PCT Guo et al., 2020), our method still achieved 38.5% performance gain. Since normal estimation heavily relies on local geometry modeling, DeepGI is naturally suited to learn structural statistics from pixels (*i.e.*, points) "unfolded/flattened" from the underlying surface. Some typical examples are provided in Fig. 13.

Table 8 Comparisons of normal regression errors measured by average cosine distance on ModelNet40

Method	Input	Error
PointNet (Qi et al., 2017a)	P; 1 K	0.47
PointNet++ (Qi et al., 2017b)	P; 1 K	0.29
PCNN (Atzmon et al., 2018)	P; 1 K	0.19
RS-CNN (Liu et al., 2019c)	P; 1 K	0.15
PCT (Guo et al., 2020)	P; 1 K	0.13
CurveNet (Xiang et al., 2021)	P; 1 K	0.11
RegGeoNet-Reg	DeepGI (P; 100 K)	**0.08**

The best result is highlighted in bold

5.5 Geometry Compression

As the increasing advancement of data acquisition for high-quality point clouds, there is an urgent need to compress such enormous volume of geometric data, considering the limited storage capacity and network bandwidth. However, different from mature and well-developed image/video compression, point cloud compression is still at its infant stage and faces great challenges in straightforwardly adapting existing image/video compression techniques due to the irregular structure of 3D point cloud data. Thus, 3D point cloud data compression is drawing much attention from both academia and industrial (Liu et al., 2019; Schwarz et al., 2018).

Fortunately, when converting unstructured point clouds into regular representation structures of DeepGI, we are able to naturally introduce standard codecs of 2D image/video compression to achieve 3D compression without efforts. This directs a highly promising research paradigm in exploiting existing mature techniques for solving new problems. Accordingly, we experimented with large-scale point cloud geometry compression by integrating DeepGIs into 2D compression pipelines.

In our implementation, we fed DeepGIs into the standard *High Efficiency Video Coding* (HEVC, Sullivan et al., 2012) codec for intra-frame prediction. Higher performance could be expected by adopting more advanced image/video codecs, e.g., VVC (Bross et al., 2021). We compared our compression pipeline with the latest reference software of *Geometry-based Point Cloud Compression* (G-PCC, Schwarz et al., 2018), the current state-of-the-art static point cloud compression codec standardized by *Moving Picture Experts Group* (MPEG). We visualized the testing models as well as the corresponding DeepGI representations in Fig. 14. Figure 15 provides the comparison of the rate-distortion of different methods, where it can be seen that our method consistently outperforms G-PCC. More specifically, at the same compression distortion, our method saves more than 60% bits compared with G-PCC. Moreover, We also observe performance degradation when removing the BCA module from the complete RegGeoNet architecture, demonstrating the necessity and effectiveness of the BCA module in enhancing spatial consistency.

5.6 Additional Verification

Although the proposed framework is customized for large-scale point cloud data, we still would like to verify its applicability under traditional experiment settings working with sparse point cloud data.

We conducted shape classification and normal estimation on ModelNet40 with 1 K points, and object segmentation on ShapeNet-Part with 2 K points. For simplicity, we did not introduce new processing techniques, and only configured the GAE module to generate DeepGIs with resolutions of 32 × 32 and 48 × 48. For deep feature extraction, we deployed a single IBM unit without max-pooling and preserved the subsequent learning process of CBM. As reported in Table 9, despite such naïve adaptations, our framework still achieves highly-competitive performance compared with traditional learning architectures for sparse point cloud processing (Please refer to Tables 2, 5, and 8 for comparison).

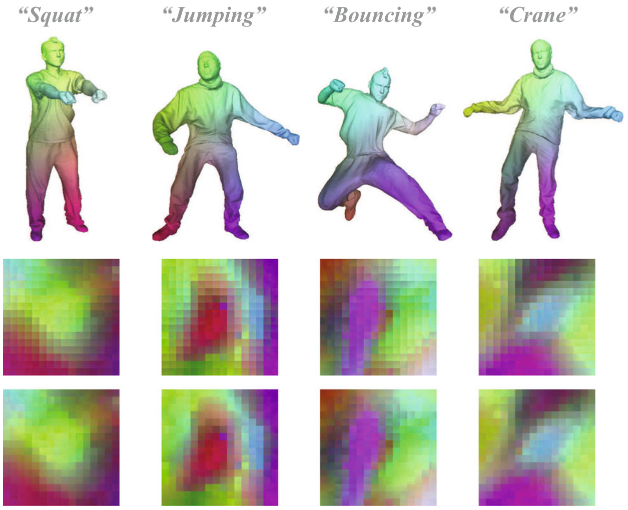

Fig. 14 Typical testing models and the corresponding DeepGIs generated without (Row 2) and with (Row 3) the BCA module

Reversely, we explored the possibility of transferring traditional deep set architectures to large-scale point cloud processing, where input models contain the same 100K points. We selected PointNet that is known to be highly-efficient without expensive feature aggregation mechanisms for verification, since the other common point-based networks turn to be completely impractical to consume such large-scale point clouds in terms of both time and memory costs. Different from our previous training environment of a single RTX 2080 Ti GPU with 11GB memory, we deployed 4 RTX 3090 GPU each with 24 GB memory to train PointNet on dense point clouds, without which the network cannot converge under a small batch size. As listed in Table 10, although PointNet benefits from the increasing number of input points, its modeling capacity still turns to be insufficient, deducing relatively limited performance gains.

Effects of Warming Up the GAE and BCA Modules As mentioned in Sect. 5.1.2, we designed warm-up procedures for the GAE module and the BCA module. In practice, compared with random initialization, it is observed that such warm-up procedures can speed up convergence of the fitting process to some extent, which may bring some benefits to the representation quality of the generated DeepGIs under limited budget of iterations. When removing these warm-up procedures during the creation of DeepGIs, the subsequent task

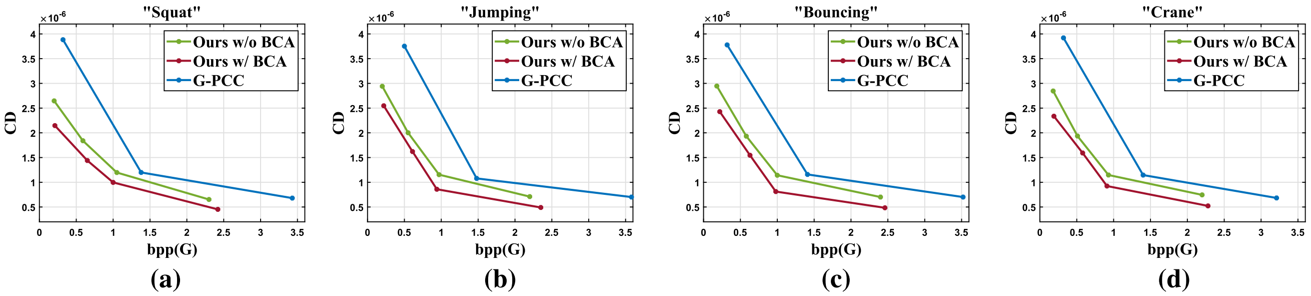

Fig. 15 Rate-distortion performance comparisons of different point cloud compression methods

performance on ModelNet40 classification slightly decreases from the original 95.22% to 95.10%. This also implies the potential value of investigating better initialization schemes for the fitting process of the GAE and BCA modules.

Computational Efficiency In Table 11, we listed time and memory costs of the three core modules involved in our RegGeoNet workflow separately. Compared with the existing deep prior counterparts (Chu et al., 2021; Gadelha et al., 2019; Hanocka et al., 2020) that usually need several hours for processing a single input, our processing pipeline shows satisfactory computational efficiency, which can be finished within seconds. In addition to the efficiency statistics of RegGeoNet, we further calculated the computational efficiency of the subsequent feature extractor on ModelNet40 classification, denoted as DeepGI-CFE-Cls, and made comparisons with representative state-of-the-art learning models in Table 12. Compared with existing methods designed to deal with sparse inputs, the proposed DeepGI-driven feature extractor shows superior time efficiency and satisfactory memory cost for processing dense points.

Representation Trade-off There are several key hyperparameters, i.e., $\{N_A, N'_L, N_L\}$, in RegGeoNet for controlling the structure and resolution of DeepGIs, where it is required that $N_A \times N'_L \geq N$ for redundant query of patch points and $N_L \geq N'_L$ for redundant grid resampling. Intuitively, such two principles aim to pursue *representation accuracy* (i.e., coverage of points) at the cost of *representation redundancy* (i.e., repetition of pixels). To statistically illustrate such trade-off relationships, we conducted detailed verification experiments on five representative categories of ModelNet40, i.e., *airplane*, *car*, *chair*, *person*, and *table*, by generating DeepGIs under different configurations, as shown in Table 13. In general, we can observe that creating higher-resolution DeepGIs with larger redundancy can significantly improve the representation accuracy (Coverage and CD). From Row 1 to Row 5, the value of $N_A \times N'_L$ is basically the same as the value of N, and thus the representation accuracy is limited by inadequate patch points collection, no matter how large N_L is. From Row 6 to Row 9, N'_L becomes larger, leading to satisfactory representation accuracy when $N_L = 256$. Besides, it is not surprised that further increasing the resolution of grid resampling (e.g., $N_L = 324$) becomes less economical. From Row 10 to Row 13, the representation accuracy has become sufficiently high.

Table 9 Performance of our framework in shape classification, object segmentation, and normal estimation, when consuming sparse point cloud data with 1 K points each

Shape classification	Object segmentation	Normal estimation
OA: 93.4%	imIoU: 86.1%	Error: 0.12

Table 10 Performance of PointNet when transferred to consume large-scale models containing 100 K points

Shape classification	Object segmentation	Normal estimation
OA: 90.1%	imIoU: 84.3%	Error: 0.35

Table 11 Time efficiency and GPU memory cost of the three core modules in the whole RegGeoNet workflow for converting a single large-scale point cloud with 100 K points into its DeepGI representation

Modules	Time cost (s)	GPU memory (GB)
GAE	4.45	0.48
LPE	0.16	1.25
BCA	2.17	0.44

6 Open Issues

This work explores a novel idea of learning regular geometry representations for large-scale 3D point clouds, which brings some interesting insights and opens new possibilities. More importantly, our method strongly reveals the value and potential of exploiting large-scale point cloud data, which has been ignored in previous works that only focus on sparse models. Taking a closer look at the proposed deep feature extraction pipeline (Sect. 4.2), DeepGI enables learning from dense point clouds efficiently, and the regularity of its repre-

Table 12 Time efficiency, GPU memory, and overall accuracy (OA) of different methods during inference for ModelNet40 classification. For fair comparison, all evaluations are performed on a single RTX 2080Ti GPU with the same input data

Method	Input	Time (s)	Memory (GB)	OA (%)
DGCNN (Wang et al., 2019)	P; 1 K	0.20	3.35	92.9
GDANet (Xu et al., 2021b)	P; 1 K	0.10	2.76	93.4
PAConv (Xu et al., 2021a)	P; 1 K	0.08	1.12	93.6
CurveNet (Xiang et al., 2021)	P; 1 K	0.18	0.49	93.8
DeepGI-CFE-Cls	DeepGI (P; 100 K)	0.05	2.90	95.2

We record the time and memory cost in a forward pass (with batch size as 24) averaged over multiple independent trials

Table 13 Trade-off between representation redundancy and accuracy (coverage) of DeepGI representations generated under different configurations, where *Redundancy* is defined as the ratio between the number of pixels in the generated DeepGIs and the number of the input points (i.e., 100 K), *Coverage* indicates what proportion of the input points are captured in the generated DeepGIs, *CD* represents the Chamfer Distance between input points and parameterized points in the generated DeepGIs after removing repetitive pixels

N	N_A	N'_L	N_L	DeepGI Resolution	Redundancy	Coverage (%)	CD (10^{-6})
100 K	1024	100	100	320 × 320	1.02×	72.78	9.19
100 K	1024	100	144	384 × 384	1.48×	82.76	7.03
100 K	1024	100	196	448 × 448	2.01×	85.32	6.44
100 K	1024	100	256	512 × 512	2.62×	85.79	6.33
100 K	1024	100	324	576 × 576	3.32×	85.87	6.31
100 K	1024	144	144	384 × 384	1.48×	85.33	3.44
100 K	1024	144	196	448 × 448	2.01×	93.11	1.82
100 K	1024	144	256	512 × 512	2.62×	95.86	1.26
100 K	1024	144	324	576 × 576	3.32×	96.59	1.10
100 K	1024	196	196	448 × 448	2.01×	92.38	1.57
100 K	1024	196	256	512 × 512	2.62×	97.06	0.61
100 K	1024	196	324	576 × 576	3.32×	99.04	0.23
100 K	1024	196	361	608 × 608	3.70×	99.70	0.15

sentation structure allows for simple but powerful modeling schemes (e.g., IBM, CBM) to be used.

Besides, based on our experience in exploring this project, we also list some interesting and promising directions for extensions and improvement of the current learning framework.

(1) *Cross-Block Discontinuity* The current hierarchical processing pipeline cannot be strictly considered as a global parameterization for shape geometry. Despite the existence of the proposed BCA module for topological repairment after global-to-local parameterization, it is still far from satisfactory in terms of maintaining global spatial continuity (*i.e.*, the blocking effect cannot be eliminated). This brings negative influence to the subsequent learning and processing pipelines, as verified in our experiments on convolution-style feature extraction (Sect. 5.2) and compression codec (Sect. 5.5).

(2) *Intra-Category Inconsistency* We are not able to ensure producing consistent parameterization across objects sharing similar geometry and topology due to the uncontrollable randomness of the fitting process. Some typical examples are given in Fig. 16. Ideally, we expect to obtain consistent image patterns on the 2D embedding domain. For example, object parts that have the same semantics can be captured in the same image areas.

(3) *Representation Robustness* The processing pipeline of RegGeoNet is designed to satisfy permutation-invariance. Technically, translation and scaling consistency can also be achieved without effort by normalizing input models into a unit sphere in advance. Besides, a more stringent requirement is that the generated DeepGIs should achieve invariance to rotation. Unfortunately, as illustrated in Fig. 17, since RegGeoNet is applied to a single input under separate optimization, it is sensitive to rotation. Deeper explorations of effective geometry- and topology-aware learning processes and constraints are needed to particularly solve this issue. It it also worth noting that, in the current deep learning-based point cloud processing research community, rotation-invariance still remains a rather challenging problem (Chen et al., 2019; Rao et al., 2019).

(4) *Extensions* There are some promising explorations that can be made to improve our method in the future works. We can investigate a global parameterization learning framework in its real sense, which is computationally efficient for consuming large-scale point clouds. It is also interesting to introduce differential geometry analysis tools to constrain regular embedding of dis-

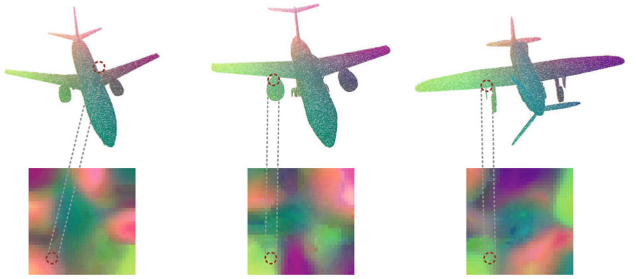

Fig. 16 Illustration of intra-category inconsistency, where we are given three models of the same *airplane* category. However, RegGeoNet fails to map object parts with the same semantics into the same pixel locations

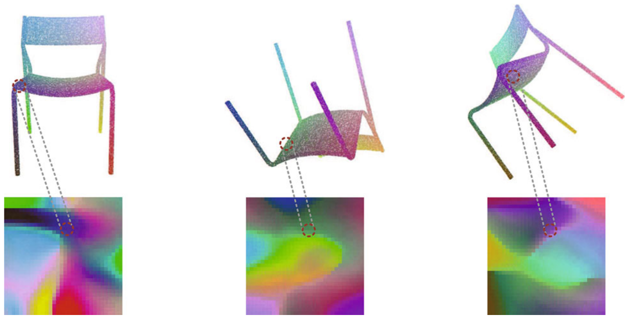

Fig. 17 Illustration of transformation sensitivity. Given the same *chair* model under different rigid transformations, the generated DeepGIs show different parameterization patterns

cretized surface points. Practically, it is possible to jointly solve correspondence and parameterization problems to achieve consistency across shapes and motions. In terms of application scenarios, we can expect that DeepGI can be naturally extended to a wide range of image-related tasks. For example, point cloud downsampling and upsampling can be achieved by image pooling and super-resolution techniques. We can also develop fully end-to-end deep compression framework for point cloud geometry based on DeepGI representations.

7 Conclusion

We explored a generic deep learning-based framework, *i.e.*, RegGeoNet, for large-scale point cloud processing. The core concept is to convert an irregular 3D point cloud into a completely regular 2D image representation called DeepGI. We further tailored an efficient feature extraction pipeline that directly operates on the DeepGI modality. In experiments, our framework shows state-of-the-art performance in diverse point cloud understanding tasks.

In general, our work reveals the necessity and potential of exploiting dense point clouds for visual reasoning, since sparse models cannot always provide sufficient geometry information. Besides, as DeepGI is a generic representation modality for point clouds, it would be promising to see more extensions in many other interesting application scenarios.

References

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S. (2016). 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1534–1543).

Atzmon, M., Maron, H., & Lipman, Y. (2018). Point convolutional neural networks by extension operators. *ACM Transactions on Graphics, 37*(71), 12.

Boscaini, D., Masci, J., Rodoià, E., Bronstein, M. (2016). Learning shape correspondence with anisotropic convolutional neural networks. In *Proceedings of the NeurIPS* (pp. 3197–3205).

Bross, B., Wang, Y. K., Ye, Y., Liu, S., Chen, J., Sullivan, G. J., & Ohm, J. R. (2021). Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology, 31*(10), 3736–3764.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., & Su, H. (2015). Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012.

Chen, C., Li, G., Xu, R., Chen, T., Wang, M., Lin, L. (2019). Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *Proceedings of the CVPR* (pp. 4994–5002).

Chu, L., Pan, H., & Wang, W. (2021). Unsupervised shape completion via deep prior in the neural tangent kernel perspective. *ACM Transactions on Graphics, 40*(3), 1–17.

Ezuz, D., Solomon, J., Kim, V. G., & Ben-Chen, M. (2017). Gwcnn: A metric alignment layer for deep shape analysis. *Computer Graphics Forum, 36*, 49–57.

Fan, H., Su, H., Guibas, L.J. (2017). A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the CVPR* (pp 605–613).

Fan, S., Dong, Q., Zhu, F., Lv, Y., Ye, P., Wang, F. Y. (2021). Scf-net: Learning spatial contextual features for large-scale point cloud segmentation. In *Proceedings of the CVPR* (pp. 14504–14513).

Gadelha, M., Wang, R., Maji, S. (2019). Shape reconstruction using differentiable projections and deep priors. In *Proceedings of the ICCV* (pp. 22–30).

Gandelsman, Y., Shocher, A., Irani, M. (2019). "Double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *Proceedings of the CVPR* (pp. 11026–11035).

Gojcic, Z., Zhou, C., Wegner, J. D., Guibas, L. J., Birdal, T. (2020). Learning multiview 3d point cloud registration. In *Proceedings of the CVPR* (pp. 1759–1769).

Gu, X., Gortler, S. J., & Hoppe, H. (2002). Geometry images. *Proceedings of the SIGGRAPH, 21*(3), 355–361.

Gu, X., Wang, Y., Wu, C., Lee, Y. J., Wang, P. (2019). Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the CVPR* (pp. 3254–3263).

Guo, M. H., Cai, J. X., Liu, Z. N., Mu, T. J., Martin, R. R., Hu, S. M. (2020), Pct: Point cloud transformer. arXiv preprint arXiv:2012.09688

Haim, N., Segol, N., Ben-Hamu, H., Maron, H., Lipman, Y. (2019) Surface networks via general covers. In *Proceedings of the ICCV* (pp. 632–641).

Hanocka, R., Metzer, G., Giryes, R., & Cohen-Or, D. (2020). Point2mesh: A self-prior for deformable meshes. *ACM Transactions on Graphics, 39*(4), 1–12.

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the CVPR* (pp. 770–778).

Heckel, R., Hand, P. (2019). Deep decoder: Concise image representations from untrained non-convolutional networks. In *Proceedings of the ICLR*.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A. (2020). Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the CVPR* (pp. 11108–11117).

Hua, B. S., Tran, M. K., Yeung, S. K. (2018). Pointwise convolutional neural networks. In *Proceedings of the CVPR* (pp. 984–993).

Jang, E., Gu, S., Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In: *Proceedings of the ICLR*.

Jiang, L., Zhao, H., Liu, S., Shen, X., Fu, C. W., Jia, J. (2019). Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proceedings of the ICCV* (pp. 10433–10441).

Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S. (2017). 3d shape segmentation with projective convolutional networks. In *Proceedings of the CVPR* (pp. 3779–3788).

Kanezaki, A., Matsushita, Y., Nishida, Y. (2018) Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the CVPR* (pp. 5010–5019).

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the CVPR* (pp. 1725–1732).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the NeurIPS* (Vol. 25, pp. 1097–1105).

Landrieu, L., Simonovsky, M. (2018) Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the CVPR* (pp. 4558–4567).

Le, E. T., Kokkinos, I., Mitra, N. J. (2020). Going deeper with lean point networks. In *Proceedings of the CVPR* (pp. 9503–9512).

Lei, H., Akhtar, N., & Mian, A. (2020). Spherical kernel for efficient graph convolution on 3d point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*(10), 3664–3680.

Li, J., Chen, B. M., Lee, G. H. (2018a). So-net: Self-organizing network for point cloud analysis. In *Proceedings of the CVPR* (pp. 9397–9406).

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B. (2018b). Pointcnn: Convolution on χ-transformed points. In *Proceedings of the NeurIPS* (pp. 828–838).

Lin, Y., Yan, Z., Huang, H., Du, D., Liu, L., Cui, S., Han, X. (2020). Fpconv: Learning local flattening for point convolution. In *Proceedings of the CVPR* (pp. 4293–4302).

Liu, H., Yuan, H., Liu, Q., Hou, J., & Liu, J. (2019). A comprehensive study and comparison of core technologies for mpeg 3-d point cloud compression. *IEEE Transactions on Broadcasting, 66*(3), 701–717.

Liu, Y., Fan, B., Meng, G., Lu, J., Xiang, S., Pan, C. (2019b). Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the ICCV* (pp. 5239–5248).

Liu, Y., Fan, B., Xiang, S., Pan, C. (2019c). Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the CVPR* (pp. 8895–8904).

Liu, Z., Tang, H., Lin, Y., Han, S. (2019d) Point-voxel cnn for efficient 3d deep learning. In *Proceedings of the NeurIPS*.

Liu, Z., Hu, H., Cao, Y., Zhang, Z., Tong, X. (2020), A closer look at local aggregation operators in point cloud analysis. In *Proceedings of the ECCV* (pp. 326–342).

Long, J., Shelhamer, E., Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the CVPR* (pp. 3431–3440).

Maddison, C. J., Mnih, A., Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the ICLR*.

Mao, J., Wang, X., Li, H. (2019). Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the ICCV* (pp. 1578–1587).

Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N., Yumer, E., Kim, V. G., & Lipman, Y. (2017). Convolutional neural networks on surfaces via seamless toric covers. *ACM Transactions on Graphics, 36*(4), 71–1.

Masci, J., Boscaini, D., Bronstein, M., Vandergheynst, P. (2015). Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the ICCVW* (pp. 37–45).

Maturana, D,, Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proceedings of the IROS* (pp. 922–928).

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the CVPR* (pp. 5115–5124).

Nezhadarya, E., Taghavi, E., Razani, R., Liu, B., Luo, J. (2020). Adaptive hierarchical down-sampling for point cloud classification. In *Proceedings of the CVPR* (pp. 12956–12964).

Park, C., Jeong, Y., Cho, M., Park, J. (2022). Fast point transformer. In *Proceedings of the CVPR*.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L. J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the CVPR* (pp. 5648–5656).

Qi, C. R., Su, H., Mo, K., Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the CVPR* (pp. 652–660).

Qi, C. R., Yi, L., Su, H., Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the NeurIPS* (pp. 5105–5114).

Qiu, S., Anwar, S., Barnes, N. (2021). Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In *Proceedings of the CVPR* (pp. 1757–1767).

Rao, Y., Lu, J., Zhou, J. (2019). Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of the CVPR* (pp. 452–460).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans Pattern Anal Mach Intell, 39*(06), 1137–1149.

Rethage, D., Wald, J., Sturm, J., Navab, N., Tombari, F. (2018). Fully-convolutional point networks for large-scale point clouds. In *Proceedings of the ECCV* (pp. 596–611).

Riba, E., Mishkin, D., Ponsa, D., Rublee, E., Bradski, G. (2020) Kornia: An open source differentiable computer vision library for pytorch. In *Proceedings of the WACV* (pp. 3674–3683).

Riegler, G., Osman Ulusoy, A., Geiger, A. (2017) Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the CVPR* (pp. 3577–3586).

Ronneberger, O., Fischer, P., Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the MICCAI* (pp. 234–241).

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323–2326.

Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P. A., Cohen, R. A., Krivokuća, M., Lasserre, S., Li, Z., et al. (2018). Emerging mpeg standards for point cloud compression.

IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 9(1), 133–148.

Simonyan, K., Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Proceedings of the NeurIPS*.

Sinha, A., Bai, J., Ramani, K. (2016), Deep learning 3d shape surfaces using geometry images. In *Proceedings of the ECCV* (pp. 223–240).

Sinha, A., Unmesh, A., Huang, Q., Ramani, K. (2017). Surfnet: Generating 3d shape surfaces using deep residual networks. In *Proceedings of the CVPR*, (pp. 6040–6049).

Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the ICCV* (pp. 945–953).

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M. H., Kautz, J. (2018). Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the CVPR* (pp. 2530–2539).

Sullivan, G. J., Ohm, J. R., Han, W. J., & Wiegand, T. (2012). Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(12), 1649–1668.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the CVPR* (pp. 2818–2826).

Tatarchenko, M., Park, J., Koltun, V., Zhou, Q. Y. (2018). Tangent convolutions for dense prediction in 3d. In *Proceedings of the CVPR* (pp. 3887–3896).

Thomas, H., Qi. C. R., Deschaud, J. E., Marcotegui, B., Goulette, F., Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the ICCV* (pp. 6411–6420).

Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M. (2015) Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the ICCV* (pp. 4489–4497).

Ulyanov, D., Vedaldi, A., Lempitsky, V. (2018). Deep image prior. In *Proceedings of the CVPR* (pp. 9446–9454).

Uy, M A., Pham, Q. H., Hua, B. S., Nguyen, T., Yeung, S. K. (2019). Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the ICCV* (pp. 1588–1597).

Verma, N., Boyer, E., Verbeek, J. (2018). Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the CVPR* (pp. 2598–2606).

Vlasic, D., Baran, I., Matusik, W., Popović, J. (2008). Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers* (pp. 1–9).

Wang, C., Smari, B., Siddiqi, K. (2018). Local spectral graph convolution for point set feature learning. In *Proceedings of the ECCV* (pp. 52–66).

Wang, P. S., Liu, Y., Guo, Y. X., Sun, C. Y., & Tong, X. (2017). O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics, 36*(4), 1–11.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics, 38*(5), 1–12.

Wu, W., Qi, Z., Fuxin, L. (2019). Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the CVPR* (pp. 9621–9630).

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the CVPR* (pp. 1912–1920).

Xiang, T., Zhang, C., Song, Y., Yu, J., Cai, W. (2021). Walk in the cloud: Learning curves for point clouds shape analysis. In *Proceedings of the ICCV* (pp. 915–924).

Xu, M., Ding, R., Zhao, H., Qi, X. (2021a). Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the CVPR* (pp. 3173–3182).

Xu, M., Zhang, J., Zhou, Z., Xu, M., Qi, X., Qiao, Y. (2021b). Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *Proceedings of the AAAI*.

Xu, Q., Sun, X., Wu, C. Y., Wang, P., Neumann, U. (2020). Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the CVPR* (pp. 5661–5670).

Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y. (2018). Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the ECCV* (pp. 87–102).

Yan, X., Zheng, C., Li, Z., Wang, S., Cui, S. (2020). Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the CVPR* (pp. 5589–5598).

Yang, J., Zhang, Q., Ni, B., Li, L., Liu, J., Zhou, M., Tian, Q. (2019). Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the CVPR* (pp. 3323–3332).

Yi, L., Kim, V. G., Ceylan, D., Shen, I. C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., & Guibas, L. (2016). A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics, 35*(6), 1–12.

Yu, F., Liu, K., Zhang, Y., Zhu, C., Xu, K. (2019) Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9491–9500).

Yu, T., Meng, J., Yuan, J. (2018). Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the CVPR* (pp. 186–194).

Zhang, Z., Hua, B. S., Yeung, S. K. (2019) Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the ICCV* (pp. 1607–1616).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.