TeethGNN: Semantic 3D Teeth Segmentation With Graph Neural Networks

Youyi Zheng, Beijia Chen, Yuefan Shen, and Kaidi Shen

Abstract—In this paper, we present TeethGNN, a novel 3D tooth segmentation method based on graph neural networks (GNNs). Given a mesh-represented 3D dental model in non-euclidean domain, our method outputs accurate and fine-grained separation of each individual tooth robust to scanning noise, foreign matters (e.g., bubbles, dental accessories, etc.), and even severe malocclusion. Unlike previous CNN-based methods that bypass handling non-euclidean mesh data by reshaping hand-crafted geometric features into regular grids, we explore the non-uniform and irregular structure of mesh itself in its dual space and exploit graph neural networks for effective geometric feature learning. To address the crowded teeth issues and incomplete segmentation that commonly exist in previous methods, we design a two-branch network, one of which predicts a segmentation label for each facet while the other regresses each facet an offset away from its tooth centroid. Clustering are later conducted on offset-shifted locations, enabling both the separation of adjoining teeth and the adjustment of incompletely segmented teeth. Exploiting GNN for directly processing mesh data frees us from extracting hand-crafted feature, and largely speeds up the inference procedure. Extensive experiments have shown that our method achieves the new state-of-the-art results for teeth segmentation and outperforms previous methods both quantitatively and qualitatively.

Index Terms—3D Teeth segmentation, graph neural network, geometric deep learning, clustering

1 Introduction

Computer-aided-design (CAD) has been widely exploited in modern oral medicine (stomatology) with a broad spectrum of applications ranging from orthodontic diagnosis to presurgery simulation, where 3D dental models of patients are often obtained through scanning and processed by task-specific operations. A ubiquitous procedure shared by all these applications is teeth segmentation, aiming at accurately segmenting each individual tooth in 3D dental models. For example, orthodontists resort to teeth segmentation to enable the rotation, removal, and rearrangement of teeth for simulating the treatment's outcome.

Despite numerous attempts to automating the segmentation process, the performance is still hindered by the following challenges. First, the high variation in both teeth size and arrangement that occurs across different individuals provokes major difficulties for building a generic segmentation method. Second, orthodontic anomalies, such as occlusions, crowded teeth and severe malocclusion, further challenge the robustness of segmentation methods. Of particular common is the situation where two neighbouring teeth are closely clung to each other, resulting in the disappearance of normal interstices. Almost all existing methods fall short in this situation since the boundary between adjoining teeth is implicit. Third, missing/rotten teeth and holes are commonly seen among people, which bring in additional challenges. Common failures resulted from scanning noise and error-prone plaster models, which usually happen deep in the mouth, further complicate the task.

Numerous methods have been proposed to tackle the above problems. Traditional geometry-based methods [1], [2], [3] are often criticized for the lack of robustness to highly complex tooth shapes and arrangements [4]. Other methods such as interactive [5], [6] or image-based ones [7], [8], [9] are either labour-intensive or inaccurate. Deep learning [10] has fertilized several fields in recent years, and tooth segmentation is no exception. Existing deep-learning based approaches [4] leverage convolutional neural networks and pack the hand-crafted geometric features into an image as input. The pre-extraction of features limits the flexibility and expressive power of subsequent representation learning, leading to deficiencies in their results. For example, misclassification of two neighboring teeth as one tooth (Fig. 1b) or wrongly halves one tooth into two parts (Fig. 1a, which are termed as the fused and halved teeth problem respectively in our context), or wrong cuts and incomplete segmentation (Fig. 1c) are often observed in their results.

In this paper, we present TeethGNN, an approach based on the recent advances in graph-based geometric deep learning, for accurate, efficient, and robust 3D teeth segmentation. Rather than resorting to a regular representation, we pass mesh itself as input represented in graph domain and utilize graph neural networks (GNNs) that explicitly captures varying topology and surface geometry of the mesh, thus enabling the network to learn discriminative features of high flexibility and expressive power. As our goal focuses on assigning each facet a label, we extend mesh in its dual space, where we regard each triangle as a graph node and

- *Youyi Zheng, Beijia Chen, and Yuefan Shen are with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China. E-mail: {youyizheng, beibeijia, jhonve}@zju.edu.cn.*
- *Kaidi Shen is with Hangzhou Choho Tech Company, Ltd., Hangzhou 310030, China. E-mail: kd_shen@outlook.com.*

Manuscript received 7 Feb. 2021; revised 4 Feb. 2022; accepted 12 Feb. 2022. Date of publication 23 Feb. 2022; date of current version 30 May 2023.
This work was supported by the National Key Research & Development Program of China under Grant 2018YFE0100900.
(Corresponding author: Kaidi Shen.)
Recommended for acceptance by A. Vaxman.
Digital Object Identifier no. 10.1109/TVCG.2022.3153501

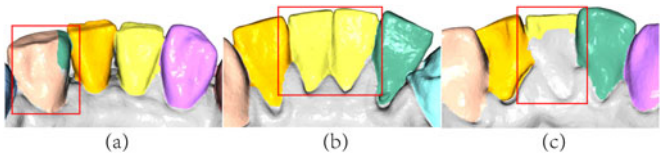


Fig. 1. Common deficiencies in existing teeth segmentation methods: (a) halved tooth, (b) fused teeth, and (c) incomplete segmentation.

model its dependencies with adjacent triangles via edges. To facilitate geometric feature learning, we utilize both static and dynamic graph convolutions to encourage information flows between connected and disconnected nodes. Omitting the hand-crafted feature extraction largely speeds up the inference process and reduces the total time from several minutes to 10 seconds for a mesh of 10,000 facets.

To further address the challenging issues of halved/fused teeth and incomplete segmentation, which may arise from the aforementioned challenges, we adapt the prediction head into two parallel branches, one of which focuses on feature learning for final label prediction while the other regresses on an offset for each node, representing the distance away from its respective tooth centroid. A density-based clustering is then performed on the offset-shifted node coordinates, thus forming tightly gathered node groups and enabling both the separation of two closely clung teeth and the adhesion of a halved tooth or an incompletely segmented tooth.

Extensive experiments show that our method outperforms existing methods both quantitatively and qualitatively. In summary, the main contributions of this paper are:

- We introduce the first GCN-based tooth segmentation method which achieves the new state-of-the-art results while largely speeding up the inference process (an order of magnitude faster).
- We present a two-branched network for predicting labels and offset-to-centroid simultaneously, enabling effective solving of halved/fused teeth and incomplete segmentation problems.
- We extend graph convolution in the dual space of mesh and utilize both static and dynamic convolution for facilitating geometric feature learning.

2 Related Works

We review the literature of 3D tooth segmentation and some recent advances in geometric deep learning operated on mesh.

2.1 3D Tooth Segmentation

Numerous methods have been proposed to solve the challenging 3d teeth segmentation problem, which can be grouped into two broad classes, depending on whether their inputs are 2D images [7], [8], [9] or 3D mesh.

Early methods avoid the complexity of handling 3D-mesh data by manipulating projected 2D images. Yamany *et al.* [7] encode curvature and surface normal information in 2D images and separate tooth exploiting image segmentation tools. Other methods [8], [9] use plan-view range image to extract orthodontic features for panoramic range image generation. 2D Segmentation is then conducted on a panoramic image and is mapped back into 3D space afterwards. A key criticism of the above methods is that they only work in mild malocclusions and introduce missed interstices and wrong cuts in severe situations.

Later methods, which directly manipulate 3D-mesh dental models, either exploit geometric features such as curvature fields [1], [2], [3] and surface contour lines [5], [6]. Curvature field is widely exploited in extracting feature regions that contain tooth boundaries due to the "valley" shape-like characteristics of the fusion region between two adjoining teeth. Tian *et al.* [1] extract region of interest based on the minimum curvatures of the surface, while Zhao *et al.* [2] take one step further by providing an additional interactive tool for post-processing. Besides, morphologic skeleton (MS) [3], "flood-fill" method [11] and "fast marching watersheds" (FMW) [12] are often incorporated with curvature-field-based methods for further segmenting the extracted regions. Though being nearly automatic, the above methods are infamous for lacking robustness to both malocclusions and scanning noise.

Opposed to curvature-field-based methods, contour-line-based methods [5], [6] involve intensive human interventions for more accurate segmentation. Ma *et al.* [6] allow users to label control points on 3d dental model to guide segmentation. An optimal segmentation path located on both control points and mesh surface is calculated via Dijkstra shortest path algorithm. Sinthanayuothin [5] enables users to label landmarks around each tooth and apply region growing to get final segmentation results. Though enabling more fine-grained results, contour-line-based methods rely on intensive human interventions, thus sacrificing the efficiency of the proposed methods.

Inspired by the surge of deep learning, Xu *et al.* [4] propose to use convolutional neural networks for tooth segmentation. To enable learning paradigms on mesh data, they extract hand-crafted geometric features for each facet, which are further adapted to image domains. Our work also follows the inspiration of utilizing deep learning in tooth segmentation, while in a novel geometric perspective. We exploit graph convolutions in the dual space of mesh to facilitate the representation learning directly applied on mesh.

2.2 Geometric Deep Learning on Mesh

Geometric deep learning [13], aiming at generalizing convolutional neural networks to non-euclidean data that widely exists in computer graphics, has gained increasing interest recently.

While lots of efforts have been made for adapting voxels [14], [15], [16] and point cloud [16], [17] for geometric feature learning, the potential of utilizing mesh in deep learning remains unexcavated. This is largely due to that mesh, though being compact and detail-preserved for representing 3d shapes, is essentially a non-uniform, irregular graph that is intractable for convolution neural networks. A packing of extracted geometric features can facilitate the direct use of convolutional neural networks [4]. However, hand-crafted feature extraction, which is computationally demanding, limits the flexibility of subsequent learning procedure.

Other method [18] focuses on designing convolutions specifically for mesh rather than bypassing this problem. MeshCNN [18] operates convolutions on edges and proposes

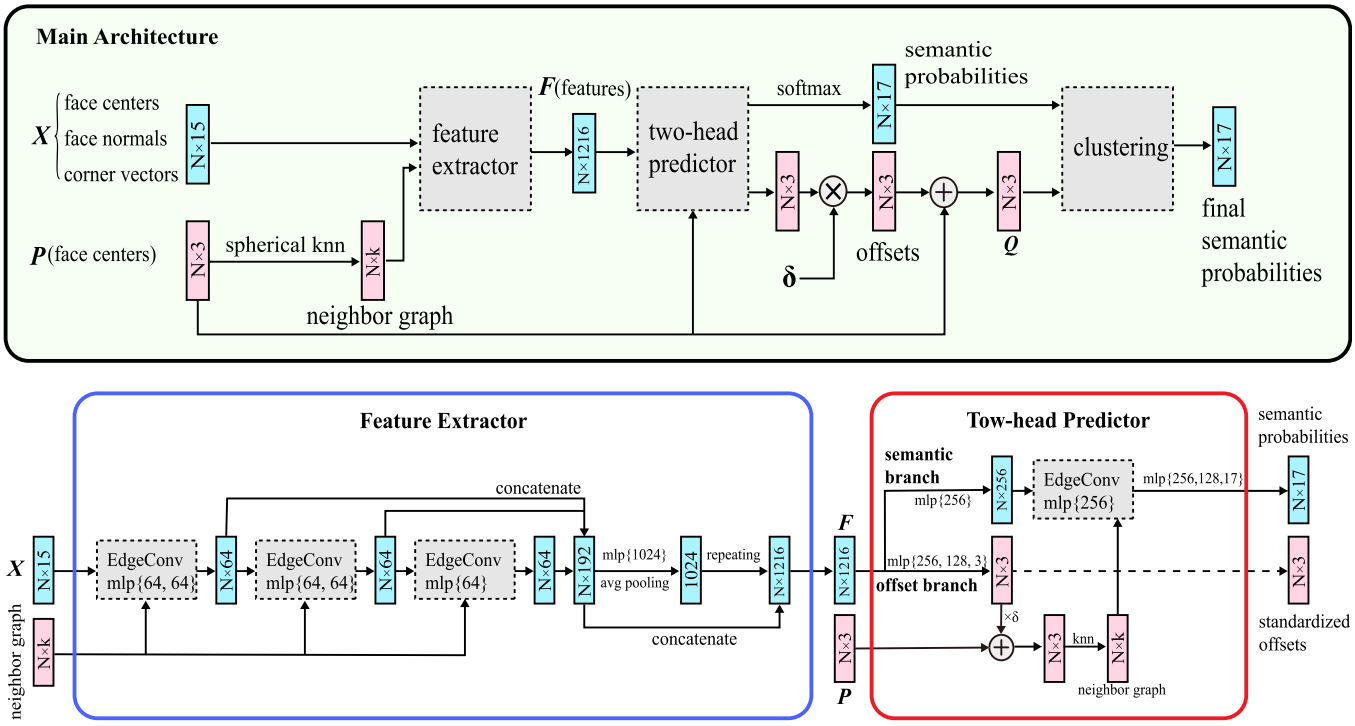


Fig. 2. Network architecture. Our network has three main components: a feature extractor network, a two-head predictor, and a clustering part. The backbone network serves as the feature extractor, which takes X and the neighbor graph for P as input and outputs node-wise features F. The two-head predictor accepts features F and node coordinates P as input and produces node-wise semantic probabilities and standardized offsets. The following clustering part accepts semantic probabilities and shifted node coordinates Q and outputs the final node-wise semantic probabilities. Inside the feature extractor and two-head predictor blocks, multi-layer perceptrons or fully-connected layers are denoted as mlp with the number of neurons in (hidden layers and/or) output layers defined as $\{\cdot, ..., \cdot\}$.

an edge pooling mechanism for eliminating less informative features akin to image feature pooling.

The emergence of graph convolution neural networks [19], [20], [21], [22], [23] further changed the landscape of this area. In analogy to convolutions on image domains, GCN broadly follows a scheme where each node aggregates information from its neighbors defined on a particular coordinate system and update features using linear and non-linear weight functions, thus enabling the learning of invariant, local, and compositional features specifically designed for graphs or manifolds[19], [21]. Masci *et al.* [21] define local patches on manifolds using geodesic polar coordinates while Boscaini *et al.* [24] use anisotropic heat kernels as weight functions. Monti *et al.*[23] generalize existed GCN-based works in terms of pseudo-coordinates and weight functions, and propose a general framework for geometric feature learning, termed as MoNet. Schult *et al.* [22] combine the benefits of Geodesic and euclidean neighborhood to further facilitate the feature learning. Our work also follows this line of works, in particular we conduct graph convolution in the dual space of mesh for facet classification and exploit the inherent mesh neighbor structure for feature learning. Moreover, we introduce a two-header predictor in our network architecture which is specially tailored for the problem of 3D teeth segmentation.

3 Methodology

Each dental mesh (upper or lower row) consists of gingiva and at most 16 teeth. To segment teeth from a dental mesh efficiently and effectively, our method operates as follows. First, we simplify the original mesh to 10000 facets. Second, we train a two-branch graph neural network, which takes initial node features and an initial adjacency graph as input and predicts both node-wise probability distributions over 17 semantic classes and node-wise offsets, and a following density-based clustering algorithm groups the shifted nodes into tooth candidates. Then, we use a label optimization method to further refine the semantic label of each facet. After that, we take a label mapping procedure to map both semantic labels and probabilities from the simplified mesh to the original mesh, followed by a boundary optimization algorithm that refines the final tooth labels on the original meshes.

3.1 Mesh Simplification

Mesh simplification is an essential procedure to avoid GPU memory overload and reduce computational costs. We first conduct mesh registration to align the teeth meshes into a common coordinate system. Specifically, we take a correctly aligned template mesh as the target mesh, uniformly sample 5000 points from source meshes and the target mesh respectively, and use Open3D's global registration method [25] to align source meshes with the target mesh. In the global registration, Open3D uses ICP registration[26] initialized with the method of [27]. Then we use QEM decimation [28] provided in Open3D to simplify original meshes to 10,000-facet meshes.

3.2 Network Architecture

As illustrated in Fig. 2 top, our network consists of three components: a feature extractor, a two-head predictor, and

a density-based clustering method. We elaborate the architectures of feature extractor, a two-head predictor in Fig. 2 down.

3.2.1 Feature Extractor

Since our method aims to assign each facet a label, we explore the dual mesh space where we regard each triangle facet as a graph node and consider facet centroids, facet normals, and three corner vectors as its node feature. A corner vector of a facet is defined as $c_{j=\{0,1,2\}} = v_j - c_{facet}$, where $v_j \in \mathbb{R}^3$ is one of the triangle vertices' coordinate and $c_{facet} \in \mathbb{R}^3$ is the coordinate of the facet centroid. The input to the network is denoted as $X \in \mathbb{R}^{N\times 15}$, where N is the number of graph nodes.

We take a modified Dynamic Graph CNN (DGCNN) [17], shown in Fig. 2, as a backbone network to extract features. DGCNN constructs a graph-based block, named EdgeConv, to dynamically compute the nodes' k-nearest neighbor graph and update node-wise features. We call those EdgeConvs dynamic EdgeConvs, in contrast to the static EegeConvs which utilize an initial mesh neighboring graph only (i.e., do not dynamically construct new edges). We exploit both of them in our architecture. Since all registered meshes have the same orientation and are in a canonical space, the spatial transform net in [17] is no longer needed. We also scrap the top branch in the original DGCNN architecture on account of no other categorical classes existing in our data.

In our network, three static EdgeConv layers extract node-wise local features. A shared fully-connected layer followed by an average pooling aggregates local features into a 1024-dimensional global feature vector. The global feature is then propagated to each node and concatenated with the local feature, forming the node-wise features $F \in \mathbb{R}^{N\times 1216}$. All layers include instance normalization are followed by a LeakyReLU activation function, except for the output layers in the offset branch and semantic branch, which regress offsets and predict semantic probabilities in the two-head predictor, respectively.

For the sake of computation and memory efficiency, we do not dynamically construct a local neighbor graph in the static EdgeConv, but rather utilize an original neighbor graph in both training and test phases. We construct the initial neighbor graph as a spherical k-nearest neighbor graph for all facet centriods $P = \{c_1, ..., c_N\} \in \mathbb{R}^{N\times 3}$, and the neighbors are constrained within the ball with a radius r. Note that we do not construct the neighbor graph on the 15-dimensional input, since otherwise two facets having similar shapes and orientations will have a connection even if they are in two far-away and unrelated locations (e.g., tooth in the left and gum in the right). If the cardinality of the spherical neighborhood of a node is less than k, we add self-loops to the neighbor graph, which means the neighbors of a node can be itself. Empirically, we take $r = 5.0$ and $k = 16$.

3.2.2 Two-head Predictor

Following the backbone network is a two-head predictor: a semantic head to predict the semantic probability distributions over the 17 classes and an offset head to predict an offset vector for each node. This idea is inspired by [29]. The offset head is a multi-layer perceptron (MLP) which takes F as input and regresses node-wise standardized offsets $O = \{o_1, ..., o_N\} \in \mathbb{R}^{N\times 3}$. Each offset is a vector from a node coordinate to its corresponding tooth centroid. A zero vector is produced if the node belongs to gingiva. We constrain the learned offsets by an L_{reg} regression loss as

$$L_{o_{reg}} = \frac{1}{N}\sum_{i}^{N} ||o_i - \frac{\hat{c}_i - c_i}{\delta}||, \tag{1}$$

where c_i is the coordinate of node i, and δ is a non-negative constant to standardize the offset. $\hat{c}_i$ is defined as

$$\hat{c}_i = \begin{cases} \frac{\sum_j c_j \cdot A(i,j)}{\sum_j A(i,j)}, & \text{if node } i \text{ belongs to a tooth} \\ c_i, & \text{else} \end{cases}, \tag{2}$$

where $A(i, j) = 1$ if node i and node j belong to the same tooth and $A(i, j) = 0$ otherwise. We take $\delta = 6.0$. Thus, a following density-based clustering method can finely separate the closely adjoining teeth and group wrongly classified nodes into one instance.

We denote the shifted node coordinates as $Q = \{c_1 + o_1 \cdot \delta, ..., c_N + o_N \cdot \delta\} \in \mathbb{R}^{N\times 3}$. In the semantic branch, a fully-connected layer transforms F to low dimensional features, which are fed into the following dynamic EdgeConv. Instead of the initial neighbor graph, this EdgeConv accepts a dynamically computed k-nearest neighbor graph for Q. Then the following MLP outputs the node-wise probability distributions over 17 classes. We use a cross entropy loss L_{sem} to supervise the semantic branch. In the training process, the total loss is defined as

$$L = L_{sem} + L_{o_{reg}}. \tag{3}$$

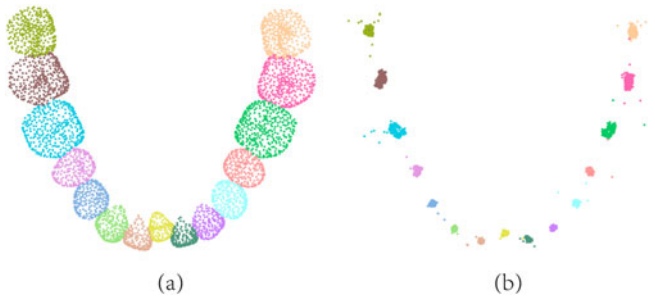


Fig. 3. Comparison between the original and shifted facet centroids. (a) original facet centroids (b) shifted facet centroids.

3.2.3 Clustering

Given the offsets, we obtain a compact representation of facet locations (see Fig. 3). As the number of teeth is uncertain and facets belonging to the same tooth are almost gathered together, we use DBSCAN (Density-based spatial clustering of applications with noise) [30] on the shifted coordinates of facets to group them into clusters. We set $\epsilon = 1.05$ and the minimum number of points as 30 in DBSCAN, which have the best performance on the training set.

For a facet i, we take the class with the maximum probability as its semantic label. Facets predicted as teeth are the candidates to be grouped. We denote the set of those facets as T and their shifted coordinates as $Q_T \subseteq Q$. We use

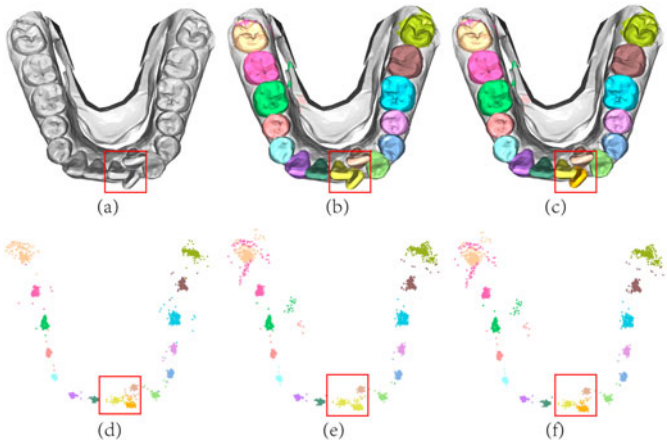


Fig. 4. Mis-grouped lower central incisors. (a) Huddled teeth. (b) Clustering results by DBSCAN. (c) Splitting after PCA and k-means. (d) Shifted centroids of ground truth tooth facets (colored by the ground truth labels). (e) Shifted centroids of predicted tooth facets after clustering (colored by the clustering results). (f) Shifted centroids of predicted tooth facets after PCA and k-means. The misclassified labels scattered in the image will be corrected by the label optimization algorithm.

DBSCAN to group points in Q_T into m initial groups. We regard a clustered object as gingiva if it has less than 60 facets. For one group, we assign it with the semantic label that has the most facets.

In some cases, as shown in Fig. 4, the lower central incisors huddle together and are wrongly clustered into one group by DBSCAN. In fact, with the benefit of offsets, groups of shifted facets belonging to different lower incisors can be well separated (see Fig. 4b, colored by the ground truth labels for better visualization). Thus, We employ PCA (principal component analysis) on each group to calculate the maximum length and k-means to split the group into two groups if the maximum length exceeds a threshold τ. We set $\tau = 6.5$ for anterior teeth, $\tau = 10.0$ for posterior teeth, which is tuned on 100 cases from the training set. The result after applying PCA and k-means is shown in Fig. 4d.

3.3 Label Optimization

As shown in Fig. 4d, some facets are still misclassified after clustering. We adopt Xu's method[4], which accepts the semantic probabilities of each facet as input, to solve this problem. Different from [4], we combine the initial semantic probabilities (generated by the semantic branch) and the clustering results to produce the final semantic probabilities for each facet. The final probability of facet i belonging to class j is defined as

$$p_i(j) = \frac{p_i^{old}(j) + \sigma \cdot m_{ij}}{\sum_{j=0}^{16}(p_i^{old}(j) + \sigma \cdot m_{ij})}, \tag{4}$$

where $p_i^{old}(j)$ is the output probability of facet i belonging to class j, σ is a non-negative constant ($\sigma = 2$), and m_{ij} is 1 if the facet i belongs to class j according to the clustering results and 0 otherwise. Based on the final probabilities, we conduct the following algorithm to further optimize the semantic labels.

Denote the facet set of a dental mesh as $\mathcal{F}$. For facet $i \in \mathcal{F}$, we denote the label as l_i under the probability $p_i(l_i)$, and the set of adjacent facets as $\mathcal{N}_i$. Note that the adjacent facets of facet i are those sharing an edge with facet i, which are different from the aforementioned adjacent nodes. We solve the problem by optimizing

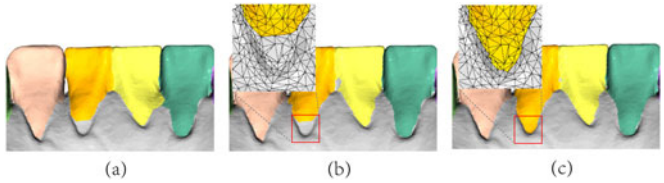


Fig. 5. (a) Results after label optimization. (b) Fuzzy clustering by Xu[4]. (c) Fuzzy clustering by ours.

$$arg \min_{\{l_i, i\in\mathcal{F}\}} \sum_{i\in\mathcal{F}} \xi_U(p_i, l_i) + \lambda \sum_{i\in\mathcal{F}, j\in\mathcal{N}_i} \xi_S(p_i, p_j, l_i, l_j) \tag{5}$$

where λ is a non-negative constant and $\xi_U(p_i, l_i) = -\log(p_i(l_i))$. The second term is defined as

$$\xi_S(p_i, p_j, l_i, l_j) = \begin{cases} 0, & l_i = l_j \\ -\log(\frac{\theta_{ij}}{\pi})\phi_{ij}, & l_i \neq l_j, \theta_{ij} \text{is concave} \\ -\beta_{ij}\log(\frac{\theta_{ij}}{\pi})\phi_{ij}, & l_i \neq l_j, \theta_{ij} \text{is convex} \end{cases}, \tag{6}$$

with $\beta_{ij} = 1 + |\hat{n}_i \cdot \hat{n}_j|$ and $\phi_{ij} = ||c_i - c_j||_2$, where $\hat{n}_i$ is the normal of facet i, c_i is the barycenter of facet i, and θ_{ij} is the dihedral angle between facet i and j. The terms $\xi_U(p_i, l_i)$ and $\xi_S(p_i, p_j, l_i, l_j)$ are identical to [4]. The only difference is that we use a smaller λ ($\lambda = 2$). The reason is that our simplified meshes have less facets and our clustering results are more accurate than Xu's, because of which a large λ brings in a large regularization on adjacent facets and will lead to over-smooth results.

3.4 Label Mapping

After the simplified mesh is labeled, we use knn ($k = 1$) to map both the semantic labels and probabilities from the simplified mesh to the original mesh. For each facet in the original mesh, we assign to it the label and probability distribution of its nearest facet in the simplified mesh. We then employ the label optimization on the boundary.

3.5 Boundary Refinement

In some cases, the optimal boundary cannot be well determined by the above methods (see Fig. 5a). We follow Xu's boundary smoothing procedure[4] to solve this problem. Xu *et al.* [4] employed an improved fuzzy clustering algorithm [31], which is based on the graph cut algorithm, to refine the boundary. For each tooth t, Xu *et al.* constructed a fuzzy region, which includes the desired boundary, by visiting a group of facets nearby the current boundary. For each fuzzy region, Xu *et al.* constructed a graph on the mesh structure, where facets are graph nodes and connected with their adjacent facets. On the border of the fuzzy region, facets adjacent to the tooth t comprise the set $\mathcal{S}$, and others are the set $\mathcal{T}$. Thus the desired boundary can be determined by the graph cut algorithm, where the capacity is defined as

$$C(i,j) = \begin{cases} \frac{1}{1+\exp(-\frac{x^2}{\sigma})}\frac{1}{1+\frac{AD(\alpha_{ij})}{avg(AD)}}, & i,j \notin \mathcal{S},\mathcal{T} \\ \infty, & \text{else} \end{cases}, \tag{7}$$

where x is the geodesic distance from the facet center to the nearest current boundary and σ is a constant value. The term $AD(\alpha_{ij})$ is the angular distance and defined as $AD(\alpha_{ij}) = \eta(1 - \cos\alpha_{ij})$, where α_{ij} is the angle between the normals of facet i and j ($\eta = 0.05$ for convex angles and $\eta = 1$ for concave angles).

But in most cases, cuts on boundaries walk across much more edges than cuts on teeth, and facets on teeth often have larger areas than those on the desired boundaries in our dataset (see Fig. 5). Under those circumstances, the graph cut algorithm might find a non-boundary cut with a small number of facets. To tackle this problem, we add a regularizer to the capacity function. The modified capacity is

$$C^{\dagger}(i,j) = C(i,j) + \gamma * (a_i + a_j), \tag{8}$$

where a_i and a_j are the areas of facet i and j, and γ is a non-negative constant (γ=50). The second term brings influence from facet areas on the corresponding capacity. Facets on the desired boundary have large curvature, while those on tooth surfaces near the boundary often have small curvature. Thus smaller adjacent facets, often on the desired boundary, make the capacity of the corresponding graph edge lower, enforcing the edge to be on the minimum cut. When $\gamma = 0$, there's no regularization on facet areas and $C^{\dagger}$ is identical to that in [4]. Fig. 5 shows the effectiveness of our modification. Note that in some dataset where meshes have a regular triangulation with equally small facets, the regularizer may deviate the result a little from the desired boundary. In this scenario, replacing the regularizer with a scaling factor related to facet areas may be a better solution.

4 Experimental Results

We conduct extensive experiments to validate the effectiveness of our method. The network is trained with PyTorch [32] on an RTX 2080Ti GPU.

Dataset. Our dental dataset is provided by a professional orthodontic company. The dataset includes 3828 meshes (1929 lower dental meshes and 1899 upper dental meshes) which are labeled manually by experts from the company. The dataset is split into training and test subsets, with 3456 and 372 meshes, respectively. Each dental mesh includes 17 semantic classes: gingiva and 16 teeth. Each facet in a mesh is assigned one label out of the 17 semantic classes.

Metrics. We use mean Intersection-over-Union (mIoU) and accuracy to evaluate the performance. The accuracy is identical to [4], which is weighted by facet areas. For a mesh, the mIoU is calculated as

$$mIoU = \frac{\sum_{s\in\mathbb{S}} IoU(s)}{|\mathbb{S}|}, \tag{9}$$

where $IoU(s)$ is the area-weighted IoU of the semantic class s, $\mathbb{S}$ is the set of semantic classes in the ground truth, and $|\mathbb{S}|$ is the cardinality of set $\mathbb{S}$.

Augmentation. We augment the data by jittering facet centers and vertices with random translation of Gaussian distribution $\mathcal{N}(0, 0.1^2)$, whose values are clipped to [-0.5, 0.5].

Results. For simplicity, we denote lower dental meshes as L and upper dental meshes as U. The overall mIoU is 97.37% and the overall accuracy is 98.89% on our dataset. Fig. 11 visualizes some of our final results.

TABLE 1
Results of the Ablation Study on the Dynamic EdgeConv in the Semantic Branch and the Effectiveness of the Point-Based Representation

		mIoU(%)	
		U	L
w/ dynamic EdgeConv	w/o clustering	97.14	96.46
	w/ clustering	**97.54**	**96.83**
w/o dynamic EdgeConv	w/o clustering	97.06	95.89
	w/ clustering	97.37	96.79
mesh representation		93.30	93.82

5 Ablation Studies

In ablation studies, we split the training set into a sub-training set and a validation set, with 2756 and 700 meshes, respectively. We train on the sub-training set for 100 epochs and report performance on the validation set. All experiments are conducted with batch size 2 and the one cycle learning rate policy implemented in PyTorch [32]. For the one cycle learning rate policy, we set the maximum learning rate as 0.001 and other parameters as default. The results are evaluated with mIoU(%) and all on the simplified meshes except for the evaluation of 'the choice of N' where the results are conducted on the original meshes.

Effectiveness of the Offset Head. To validate the effectiveness of the offset branch, we conduct experiments to compare the performance of one-head and two-head networks. Since no offset will be predicted in the one-head network, we cannot dynamically construct the neighbor graph on the shifted facets as we do in the two-head network. Thus, we replace the last dynamic EdgeConv with a static EdgeConv, for which the neighbor graph is identical to the one for the three former EdgeConvs. Table 2 shows the efficacy of the offset branch and the clustering method.

Choice of Dynamic or Static EdgeConvs in the Backbone Network. Table 3 shows the results of dynamic and static EdgeConvs in our backbone network. Considering the first EdgeConv accepts the raw input which is not suitable for constructing a neighbor graph and the last EdgeConv utilizes the shift node coordinates, we only replace the second and the third EdgeConvs with dynamic EdgeConvs, which construct neighbor graphs from the input features. Table 3 indicates that there has been no improvement using dynamic EdgeConvs which means the original mesh structure is already effective enough for dental mesh segmentation. In the dynamic EdgeConv, the dynamic calculation of a node's neighborhood is unstable and reliant on the features extracted by the previous EdgeConv. If features of facets in different teeth are muddled together (adjacent teeth are similar, such as premolars or incisors), a bad neighbor graph might be built inside the EdgeConv and thus may cause a drop of the performance.

Effectiveness of the Dynamic EdgeConv in the Semantic Branch. Since the shifted coordinates Q of facets is a better representation than the original coordinates P, we consider

TABLE 2
Results of the Ablation Study on the Offset Head

	One-head	Two-head	
		w/o clustering	w/ clustering
mIoU(%, U)	96.96	97.14	**97.54**
mIoU(%, L)	96.05	96.46	**96.83**

TABLE 3
Results of the Ablation Study on the Choice of Dynamic and Static Edgeconvs in the Backbone Network

	Dynamic	Static
mIoU(%, U)	97.28	**97.54**
mIoU(%, L)	96.78	**96.83**

Results are evaluated on the simplified meshes.

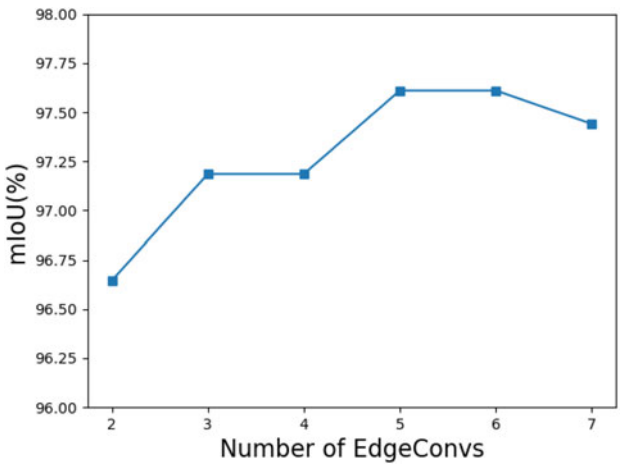


Fig. 6. Results of the ablation study on the numbers of EdgeConvs in the backbone network. Results are evaluated on the simplified meshes.

using a dynamic EdgeConv in the semantic branch to produce facet-wise probabilities more accurately. In doing so, a new neighbor graph is calculated dynamically according to Q and then utilized by the EdgeConv. Table 1 shows the dynamic EdgeConv brings in performance improvements.

Choice of the Number of EdgeConvs. We exploit the representation power of the feature extractor by stacking a different number of EdgeConvs in the backbone network. As illustrated in Fig. 6, when the number of EdgeConvs increases to 5, the mIoU peaks. However, if the number of EdgeConvs is equal to or larger than 5, the memory usage will increase significantly so that we can only set the batch size as 1 when training the network. Besides, adding more EdgeConvs may cause overfitting. Considering the performance and memory-and-computation efficiency, we suggest using 3 or 5 EdgeConvs in the backbone network.

Choice of N. In our method, meshes with roughly 150,000 facets are simplified to those with N facets. A larger N reduces fidelity loss but increases more memory usage and computation time. To determine the best N, we experiment on the simplified meshes with a different number of facets. Since the mIoU computed on meshes in different resolutions are not directly comparable, we evaluate the results on the original meshes. As illustrated in Table 4, mIoU increases as

TABLE 4
Results of the Ablation Study on the Choice of N

N	3000	5000	10000	15000
mIoU(%, w/o post-processing)	92.22	93.70	94.87	**95.47**
mIoU(%, w/ post-processing)	97.18	97.35	97.43	**97.54**

Results are evaluated on the original meshes.

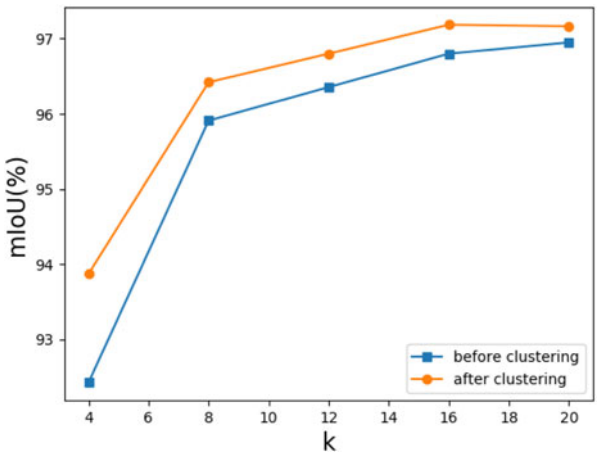


Fig. 7. Results of the ablation study on the choice of k. Results are evaluated on the simplified meshes.

N increases. For a counterbalance of the network weights and efficacy, we set N to 10,000 in our experiments.

Choice of k. k is the number of nearest neighbors in our EdgeConvs. A larger k provides a larger field of view but lacks memory-and-computation efficiency. To choose an appropriate k, we experiment and evaluate the results on different k before and after clustering (see Fig. 7). In the experiments, we take the same k for all EdgeConvs. As we can see in Fig. 7, the immediate results of the network (mIoU before clustering) improve with the increase of k, but the results after clustering do not improve when $k > 16$. Thus, we set $k = 16$ throughout our experiments.

Effectiveness of the Point-Based Representation. We use the point-based representation in our network, which allows simplicity, effectiveness, and efficiency. We can also use the mesh representation in our network, which means constructing the neighbor graph on the mesh structure. However, in the mesh representation, with the same depth of our network, the FoV (field of view) is fixed and insufficient and cannot be enlarged because each facet has a fixed number of neighbors. On the contrary, with point representation we can enlarge the FoV by increasing k in a point-based representation. We conduct an experiment to demonstrate the effectiveness of the point-based representation. In this experiment, we use the same graph constructed on the mesh structure in all EdgeConvs. The results are listed in Table. 1.

6 Comparisons

To validate the effectiveness of our method, we make comparisons with existing methods. In these experiments, we train on the training set and report results on the test set. For our network, we set $k = 16$ and $N = 10,000$, and we

TABLE 5
Comparison on the Simplified Meshes in our Dataset

	mIoU(%)	Acc(%)
Ours	**97.49**	**98.94**
DGCNN[17]	96.54	98.55
MeshCNN[18]	94.38	97.84
PointGroup[29]	86.13	94.51

train the 3-EdgeConv backbone network and evaluate the final results. Results are evaluated with mIoU and area-weighted accuracy on all upper and lower meshes. Although our method is tailored for teeth segmentation, we also conduct a light comparison on the COSEG [33] dataset to attest the efficacy of our method.

We compare our method with four state-of-the-art alternatives: DGCNN [17], MeshCNN [18], PointGroup [29], and Xu *et al.* [4]. In the comparison experiment with DGCNN [17], MeshCNN [18], and PointGroup [29], we train and report the results on the simplified meshes (see Table 5). In the comparison experiment with Xu *et al.* [4], we report the results on the original meshes (see Table 6).

Comparison With DGCNN. As the basis of our network, DGCNN can segment teeth properly if there exist no missing, ectopic, or deciduous teeth. However, DGCNN tends to produce halved/fused teeth and incomplete segmentation problems (see Fig. 10) on those cases for the reason that DGCNN can only classify facets as part of some certain tooth but cannot group facets into a single tooth. For instance, if one incisor is ectopic, DGCNN will tend to halve one of the incisors into two parts (see the second row in Fig. 10). We list the quantitative results in Table 5 and the qualitative comparison in Fig. 10.

Comparison With MeshCNN. To compare our network with MeshCNN, we train MeshCNN on the simplified meshes (10,000 facets/ 16,000 edges). MeshCNN is not suitable for meshes with a large number of facets for the slow training process (3s/mesh), whereas it takes only 0.1s/mesh when training our network. It gobbles up almost all GPU's memory when training on a single mesh with 16,000 edges (10,000 facets) and keeps slogging on in the simplification process. The problems in DGCNN stated above can be also observed in MeshCNN. Besides, the similarities between certain teeth (incisors or premolars) and the left-right symmetry often make it confused because MeshCNN is agnostic to teeth locations. We train MeshCNN for 100 epochs and report the results in Table 5.

Comparison With PointGroup. PointGroup[29] uses a 3D U-Net to extract point-wise features, which is not suitable for the teeth data since the 3D U-Net accepts voxelized data and often fail to produce fine-grained features. However, our network is tailored to the teeth data and can extract facet-wise features finely. The comparative results are reported in Table 5.

TABLE 6
Comparison With [4]

		mIoU(%)	Acc(%)
Our dataset	Ours	**97.49**	**98.95**
	Xu *et al.*[4]	91.99	96.26
[4]'s dataset	Ours	**96.91**	**99.25**
	Xu *et al.*[4]	95.66	98.97

Results are evaluated on the original meshes.

TABLE 7
Quantitative Comparisons (Accuracy) of Different Methods on COSEG

	Vases(%)	Chairs(%)	Telealiens(%)
Ours	**97.38**	**99.75**	**98.02**
DGCNN[17]	97.05	99.03	91.04
MeshCNN[18]	97.27	99.63	97.56
PointNet[34]	91.5	70.2	54.4
PointNet++[35]	94.7	98.9	79.1
PointCNN[36]	96.37	99.31	97.40

Comparison With Xu's Method. In the comparison experiment with Xu *et al.* [4], we conduct two experiments: one on our dataset and the other on Xu's dataset, respectively. The difference between our data and Xu's data is that meshes in our data are not watertight, whereas those in Xu's data are. In the experiment on Xu's data, we train our network on a training set (1050 meshes) and report the results on a test set (150 meshes). We evaluate Xu's method (with author-provided trained model) on the same test set. In the experiment on our data, we retrain Xu's model on 40000-facets meshes simplified from the same original data as they did. As we train and test our models on meshes in a different resolution, we evaluate the final results on the original meshes after the whole procedure of both methods. Xu's method often fails in missing, ectopic or wisdom teeth and lower incisors, whereas ours can segment these teeth accurately (see Fig. 10). Results evaluated on the original meshes in both experiments show that our method outperforms Xu's method by a large margin (see Table 6).

Computation Time. The training of our network takes 10 hours using a single RTX 2080 Ti GPU. At inference, the total procedure takes 8.5s (1.5s for registration, 1s for simplification, 53ms for network inference, and 6s for post-processing) for a mesh with 150,000 facets. The inference time is tested on an Intel Core i7-8700 CPU and an RTX 2080 Ti GPU. Note that the total running time is 1x magnitude faster than the method of Xu *et al.* [4] ($\sim$5 minutes as reported in their paper).

Training Configurations. For DGCNN, we use the original configuration of DGCNN for semantic segmentation and set $k = 16$. For the first dynamic EdgeConv, the neighbor graph is constructed by the coordinates of facet centers, not the 15-dimensional input. We train DGCNN for 100 epochs with a batch size of 4. For PointGroup, we set the voxel size as 1cm, the clustering radius r as 0.5cm, and others as default. We train PointGroup for 384 epochs with a batch size of 8. For MeshCNN, we train it on the simplified 10000-facet meshes(15000 to 16000 edges) and set the input number of edges as 16000. We set batch size as 1 and use instance normalization. The input meshes are sequentially pooled to 11000, 6000, and 3500 edges. Other parameters are set as default. We train MeshCNN for 100 epochs.

Comparison on the COSEG Dataset. We conduct comparisons on the COSEG dataset, which includes 200 aliens, 300

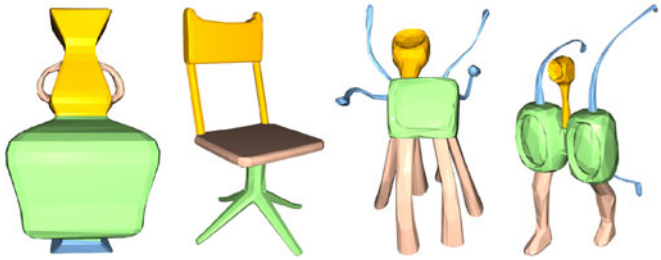


Fig. 8. Some representative segmentation results on the COSEG dataset.

vases, and 400 chairs. On this dataset, [18] reported accuracy of their method in comparison with PointNet[34], PointNet++[35], and PointCNN[36]. We take the reported results directly from [18] and list them in Table 7. As for the experiment of our network, we use the same train-test split as [18], augment the data with random rotation on the y-axis and train our network for 200 epochs. The results are reported in Table 7. Our network outperforms the other methods. Fig. 8 shows some qualitative results of ours.

7 Limitations

Our method has the following limitations. First, for some severely rotten teeth which do not have enough facets (see Fig. 9a), our network tends to classify part of the teeth as gingiva, which is hard to fix by the post-processing algorithm. Second, if the tooth partly erupts (see Fig. 9b), the network may be prone to mislabel the surrounding gingiva as part of the tooth or cannot recognize it as a tooth. This problem is difficult to be solved by the boundary optimization algorithm since the gingiva makes up the major part of the mislabeled tooth. Besides, the accuracy of the boundary optimization relies on the convexness of the boundary between teeth and gingiva. If the boundary area lacks convexness or is almost flat, the boundary optimization algorithm can hardly reach an accurate result[4]. The quality of dental meshes is also essential to our method. If the dental mesh is with significant noises, such as braces on the teeth (Fig. 9c) or severe tooth distortion

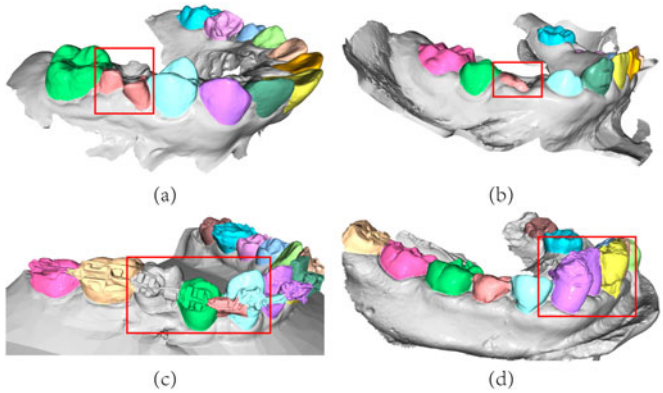


Fig. 9. Imperfect results of our method in the case of (a) severely rotten tooth, (b) partially erupted tooth, (c) teeth with braces and (d) severely distorted teeth. Adding more such training samples may alleviate this problem.

(Fig. 9d), it is hard to segment and separate teeth accurately. Adding more training data may alleviate the aforementioned problems.

8 Conclusion

In this paper, we propose TeethGNN, a graph-based neural network for semantic dental teeth segmentation. To tackle halved/fused teeth and incomplete segmentation problems that commonly exist in previous methods and improve the segmentation accuracy of dental meshes, we introduce a novel two-branch architecture: a semantic branch to produces facet-wise semantic labels and an offset branch to predict a offset-to-centroid vector for each graph node. The output of the two branches are further fused in a clustering algorithm that groups facets into different teeth. To further refine the tooth boundaries, we design a modified boundary smoothing algorithm. Our method has significantly improved the speed of the current deep-learning based methods and reached a new state-of-the-art accuracy for teeth segmentation. It is robust to rotten, missing, crowded, and ectopic-tooth cases.

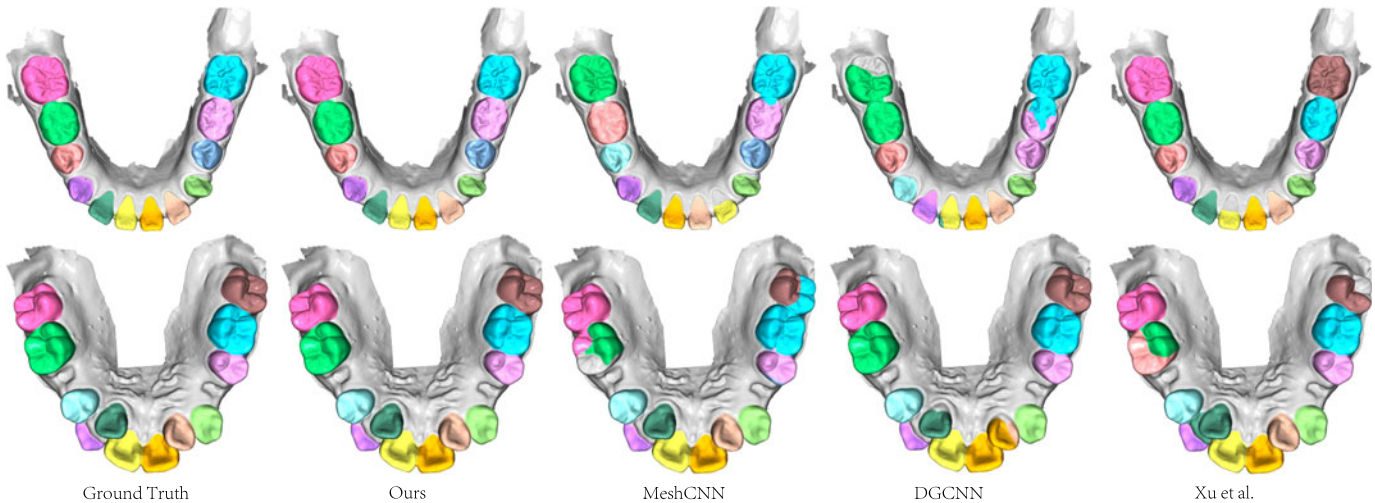


Fig. 10. Comparison with other methods. All are the final results of the original meshes. DGCNN's and MeshCNN's results are also processed by our post-processing algorithm. The two rows are the results of a case with deciduous teeth and a case with missing and ectopic teeth, respectively.

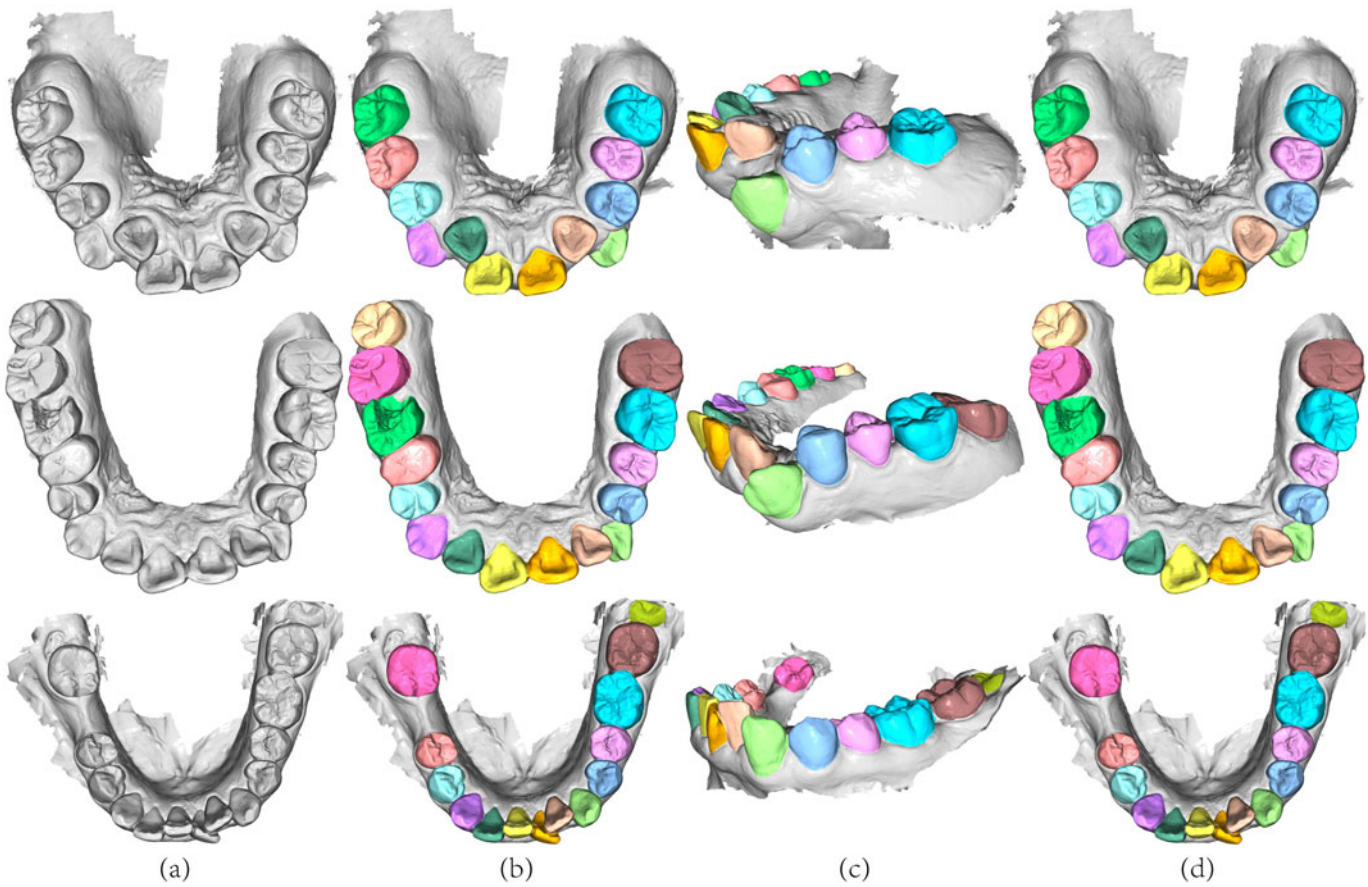


Fig. 11. Our tooth-segmentation results. (a) Original meshes. (b) Our segmentation results. (c) Another view of our results. (d) Ground truth.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their constructive comments. Youyi Zheng and Beijia Chen are co-first authors.

REFERENCES

[1] T. Yuan, W. Liao, N. Dai, X. Cheng, and Q. Yu, "Single-tooth modeling for 3D dental model," *Int. J. Biomed. Imag.*, vol. 2010, pp. 1–14, 2010.

[2] M. Zhao, L. Ma, W. Tan, and D. Nie, "Interactive tooth segmentation of dental models," in *Proc. IEEE Eng. Med. Biol. 27th Annu. Conf.*, 2006, pp. 654–657.

[3] K. Wu, L. Chen, J. Li, and Y. Zhou, "Tooth segmentation on dental meshes using morphologic skeleton," *Comput. Graph.*, vol. 38, pp. 199–211, 2014.

[4] X. Xu, C. Liu, and Y. Zheng, "3D tooth segmentation and labeling using deep convolutional neural networks," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 7, pp. 2336–2348, Jul. 2018.

[5] C. Sinthanayothin and W. Tharanont, "Orthodontics treatment simulation by teeth segmentation and setup," in *Proc. IEEE 5th Int. Conf. Elect. Eng./Electron. Comput. Telecommun. Informat. Technol.*, 2008, pp. 81–84.

[6] M. Yaqi and L. Zhongke, "Computer aided orthodontics treatment by virtual segmentation and adjustment," in *Proc. IEEE Int. Conf. Image Anal. Signal Process.*, 2010, pp. 336–339.

[7] S. M. Yamany and A. M. El-Bialy , "Efficient free-form surface representation with application in orthodontics," in *Three-Dimensional Image Capture and Applications II*, vol. 3640. Bellingham, WA, USA: SPIE, 1999, pp. 115–124.

[8] T. Kondo, S. H. Ong, and K. W. Foong, "Tooth segmentation of dental study models using range images," *IEEE Trans. Med. Imag.*, vol. 23, no. 3, pp. 350–362, Mar. 2004.

[9] N. Wongwaen and C. Sinthanayothin, "Computerized algorithm for 3D teeth segmentation," in *Proc. IEEE Int. Conf. Electron. Informat. Eng.*, 2010, pp. V1–277.

[10] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[11] Y. Kumar, R. Janardan, B. Larson, and J. Moon, "Improved segmentation of teeth in dental models," *Comput.-Aided Des. Appl.*, vol. 8, no. 2, pp. 211–224, 2011.

[12] Z. Li, X. Ning, and Z. Wang, "A fast segmentation method for stl teeth model," in *Proc. IEEE/ICME Int. Conf. Complex Med. Eng.*, 2007, pp. 163–166.

[13] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[14] Z. Wu, *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.

[15] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," 2016, *arXiv:1608.04236*.

[16] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "SEGCLoud: Semantic segmentation of 3D point clouds," in *Proc. IEEE Int. Conf. 3D Vis.*, 2017, pp. 537–547.

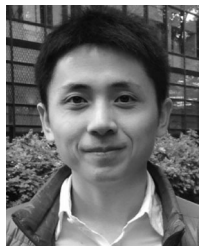
[17] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *Acm Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[18] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or , "MeshCNN: A network with an edge," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, 2019.

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2018, *arXiv:1810.00826*.

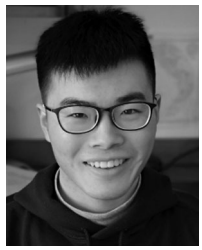
[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[21] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2015, pp. 37–45.
[22] J. Schult, F. Engelmann, T. Kontogianni, and B. Leibe, "DualConvMesh-Net: Joint geodesic and Euclidean convolutions on 3D meshes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8612–8622.
[23] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5115–5124.
[24] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. Adv. Neural Informat. Process. Syst.*, 2016, pp. 3189–3197.
[25] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," 2018, *arXiv:1801.09847*.
[26] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: Control Paradigms and Data Struct.s*, vol. 1611. Bellingham, WA, USA: SPIE, 1992, pp. 586–606.
[27] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 3212–3217.
[28] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proc. 24th Annu. Conf. Comput. Graph. Interactive Techn.*, 1997, pp. 209–216.
[29] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "PointGroup: Dual-set point grouping for 3D instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4867–4876.
[30] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 226–231.
[31] S. Katz and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 954–961, Jul. 2003.
[32] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[33] Y. Wang, S. Asafi, O. Van Kaick , H. Zhang, D. Cohen-Or, and B. Chen, "Active co-analysis of a set of shapes," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 1–10, 2012.
[34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.
[35] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv:1706.02413*.
[36] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on x-transformed points," in *Proc. Adv. Neural Informat. Process. Syst.*, pp. 820–830, 2018.

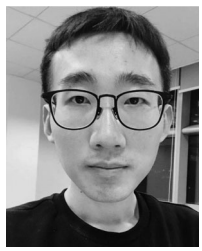


Youyi Zheng received the BS and MS degrees in mathematics from Zhejiang University, China, in 2005 and 2007, respectively, and the PhD degree in computer science from the Hong Kong University of Science & Technology in 2011. He is currently a researcher with the State Key Lab of CAD&CG, Zhejiang University. His research interests include geometric modeling, imaging, and human-computer interaction. He was an associate editor for *The Visual Computer* and *Frontiers of Computer Science*.

Beijia Chen received the BS and MS degrees from the Nanjing University of Science and Technology. She is currently working toward the PhD degree with the State Key Lab of CAD&CG, Zhejiang University. Her research interests include 3D human recovery, image manipulation, and deep learning.



Yuefan Shen received the BS degree from the School of Software Engineering, Shandong University. He is currently working toward the PhD degree with the State Key Lab of CAD&CG, Zhejiang University. His research interests include image-based modeling and 3D data processing with deep learning.



Kaidi Shen received the BMed degree in stomatology from Nankai University, and the MEng degree in computer science from Zhejiang University. He is currently with Hangzhou ChohoTech Co., Ltd. His research interests include 3D segmentation, geometric modeling, and deep learning.