3D Mesh Segmentation via Multi-branch 1D Convolutional Neural Networks

D. George¹, Xianghua Xie¹ and Gary KL. Tam¹

¹Swansea University, United Kingdom

Abstract

3D mesh segmentation is an important research area in computer graphics, and there is an increasing interest in applying deep learning to this challenging area. We observe that 1) existing techniques are either slow to train or sensitive to feature resizing and sampling, 2) in the literature there are minimal comparative studies and 3) techniques often suffer from reproducibility issue. These hinder the research development of supervised segmentation tasks. This study contributes in two ways. First, we propose a novel convolutional neural network technique for mesh segmentation, using 1D data and filters, and a multi-branch network for separate training of features of three different scales. We also propose a novel way of computing conformal factor, which is less sensitive to small areas of large curvatures, and improve graph-cut refinement with the addition of a geometric feature term. The technique gives better results than the state of the art. Secondly, we provide a comprehensive study and implementations of several deep learning techniques, namely, neural networks (NNs), autoencoders (AEs) and convolutional neural networks (CNNs), which use an architecture of at least two layers deep. The significance of the study is that it offers a novel fast and accurate CNN technique, and a comparison of several other deep learning techniques for comparison.

1. Introduction

Automatic mesh segmentation is the decomposition of a 3D mesh into meaningful parts. It aims to produce results as similar to those produced by humans. The ability to properly segment a 3D mesh is important to many downstream applications, such as shape retrieval [SSS°10], matching [KvKSHCO15], editing [YZX°04, CLL°15], deformation [YXG°13] and modelling [CZL°15]. Many of these applications require well-defined mesh segments, making a robust and accurate segmentation algorithm essential.

From a machine learning point of view, mesh segmentation can be broadly categorised as unsupervised and supervised segmentation. Earlier techniques focused on segmenting a single mesh in an unsupervised manner. It first computes features (e.g. shape diameter function [SSCO08], approximate convexity [KFK°14] and curvature [MDS°02]) for the faces of the meshes, and uses an optimisation technique to produce the segmentation results. Notable techniques include k-means [STK02], mean-shift clustering [CM02], normalized and randomized cuts [GF08]. A detailed survey of these techniques can be found in [Sha08, CGF09]. Given the large shape variability of segments, more recent approaches consider consistent segmentation of a collection of shapes, where class labels are consistent throughout the set and not just for a single mesh [SvKK°11, HKG11].

Segmentation often requires a higher level understanding of the 3D shapes, as composition of an object often relates to shapes and functionality of its parts [HZvK°15]. Supervised techniques treat segmentation as a labelling problem and use machine learning to optimise the mapping from features to labels. It requires extensive manual effort to label all data properly for training. With the recent effort from the community (e.g. shape benchmarks [CGF09]), supervised techniques are gaining focus. The work by [KHS10] pioneered to apply joint boosting on a large set of shape features for effective labelling. Recently, [XXLX14] used an extreme learning machine (a single layer wide neural network) and [GZC15] applied Convolutional Neural Networks (CNNs) to mesh segmentation.

Despite these research efforts, there are still many research questions unexplored. First, it is generally unclear which deep learning techniques work and which do not for mesh segmentation, and what features work best. To the best of our knowledge, most supervised techniques use features derived from one face only. Multi-scale features derived from a set of local faces have not yet been used. Also, there has been no comparative analysis of a broader spectrum of deep learning techniques. Second, the reproducibility of these techniques depends on the architecture, exact implementation and the set of training datasets used. This information and along with complete source code is largely unavailable. Coupled with these, there are also challenges in training the networks properly due to the variability in CNN architectures, large number of samples (200K samples per set) and lengthy training time (in terms of months). All these elements hinder the development of supervised 3D segmentation techniques.
In this paper, we try to address three research questions. (i) Compared to existing literature that use only features defined per face, are multi-scale features that derived from a set of faces useful? (ii) Compared to [GZC15] that reshapes features into 2D images and applies a basic image-based CNN pipeline for shape segmentation, can we treat input features as a single 1D feature vector? This avoids the tuning of image size whilst significantly improves efficiency and retains the performance of CNNs. (iii) Finally, how much improvement can CNNs have over existing techniques in deep learning. Our contributions of this paper are two-folds:

- We introduce a novel, fast and accurate CNN technique to 3D mesh segmentation, with three subsequent contributions:  
  1. We introduce a multi-branch network that separately trains features of different scales. These multi-scale features are derived from features that associates to an increasing local neighbourhood of faces. The use of 1D feature vectors also removes most of the assumed feature relationships that are imposed by an image-based CNN when reshaping the feature vector into a 2D image. This leads to a higher accuracy and comparable efficiency (being able to handle over three times more features), compared to the state of the art [GZC15].
  2. We propose a novel feature vector of conformal form which is computed from incremental smoothing of geometry. It is less sensitive to high curvature noise, and consistently provides higher segmentation accuracy than [BCG08] alone.
  3. We also improves on existing graph-cut technique for mesh segmentation by introducing a geometric feature term, offering higher refinement accuracy.

- Second, we perform a comprehensive comparison of deep learning techniques (at least two layers deep) for supervised mesh segmentation, specifically neural network (NN), autoencoder (AE) and CNN [GZC15], showing the strengths and limitations of each technique by comparing their accuracies and time complexities.

The rest of the paper is structured as follows: Section 2 provides a more detailed summary of both supervised and unsupervised mesh segmentation techniques. Section 3 discusses different methods we compared, and our proposed technique using multi-branch 1D CNN and multi-scale features for 3D segmentation. Section 4 discussed our experiments and the results of all methods tested. Finally, Section 5 concludes this paper.

2. Related Work

This section first surveys existing techniques, with an emphasis on supervised segmentation. We then discuss the problems of existing supervised techniques, leading to our contributions.

Unsupervised Segmentation Early work focused on simple, yet effective ideas for segmenting a single mesh [STK02, SSC008]. They often performed clustering on geometric features (e.g. shape diameter function [SSC008], geodesic distances [HSKK01], curvature [GCO06]) or partitioned based on properties that can be derived from the mesh itself (e.g. skeleton [SSC008], convexity [KFK∗14], fitting primitive shapes [AFS06]). Many of these ideas have been shown effective, giving rise to a wide range of shape descriptors, and segmentation techniques, supporting many downstream applications [Sha08, CFG09]. However, segmenting a single mesh using a few features is often difficult due to the large variations in terms of shape and topology, even within the same class of objects. Recent research has adopted the co-analysis framework to investigate consistent segmentation of a collection of shapes from a single object class [SvKK∗11, MHLH13, WWS∗13, SQX∗16]. For example, all legs in a chair set should be labelled the same. Such constraint is powerful yet requires less human effort. However, these methods rely on consistent geometric similarity within the set and a reliable shape/part matching algorithm in order to perform well. The large variations between different shapes in the same set and the sparse number of shapes in the set often cause problems in the final segmentation [TPT15]. More importantly, segments of a shape are often associated to its functionality - a high-level understanding of shapes [HZvK∗15]. Therefore, there is an increasing interest in supervised segmentation techniques, trying to learn a high-level mapping directly from feature to segment.

Supervised Segmentation techniques treat 3D mesh segmentation as a labelling problem and use machine learning to optimise the mapping from features to labels. It requires extensive manual effort to label all data. The recent effort from the community contributed to a large set of segmentation benchmarks (e.g. [CFG09]).

Existing supervised techniques rely on local features. The work by [KHS10] proposed a method for mesh segmentation where a large pool of geometric features are ranked using jointBoost so that the best features are used to describe specific segments. Similarly, the work by [BLVD11] ranks a large pool of features in order to detect the optimal segment boundaries for a given mesh, and in [XXLX14], an extreme learning machine was trained to classify labels. However, supervised methods can perform poorly on very complex meshes, due to insufficient training data or large variations within label classes [XXLX14, GZC15].

Recently, [GZC15] extended the CNN idea to label 3D segmentation. They use a large pool of geometric features, reshape the features into a matrix resembling that of a 2D image, and then trained a CNN on these “images” using the ground truth labels. It tries...
to solve the problem that geometric features are not linearly separable, by non-linearly and hierarchically compressing the features through the network to obtain a better feature description of different classes. However, it reshapes the feature vector into an “image” to fit the image-based CNN pipeline. The reshaping and the use of 2D filters may infer relationships between adjacent rows of features that may have no correlation.

From the literature, we have two further observations. First, most existing techniques use local features developed by the influential work [KHS10]. These features are defined per face, and to the best of our knowledge, multi-scale features that derived from set of local faces have not been considered before in supervised segmentation. We hypothesise that multi-scale features would be useful because face-based [SvKK∗11] and patch-based techniques [WWS∗13] have both shown good performance in co-segmentation. Second, whilst deep learning is useful, there is not much analysis as how CNN performs compared to other techniques in the deep learning family.

In this paper, we show that by using multi-scale features, treating them as three 1D vectors and applying multi-branch 1D CNN filters through the network, we avoid the parameter tuning problem of reshaping a 2D matrix. Our method is able to produce better accuracy and handle more than three times more features compared to our faithful implementation of the state of the art [GZC15]. Further, we provide comprehensive evaluation of deep learning techniques, and show how CNN, though more complex, can improve performance over simpler architectures (NN, AE). It is worth noting that we have found existing methods lack reproducibility. Some existing work have not provided any or complete experimental codes. Despite using the exact architecture and setting stated, some high performing results are hard to reproduce.

3. Methodology

This section discusses the deep learning techniques proposed and evaluated. Section 3.1 discusses geometric features used. We then summarise several techniques, Fully Connected Neural Network (NNs), Autoencoder(AE) + Random Forests(RFs), and Convolutional Neural Networks (CNNs), in Section 3.2-3.4, focusing on models which are at least two layers deep. Each technique is broken down into stages, namely, feature extraction, pre-processing, learning and classification, and post-processing (Figure 1). Section 3.5 describes the use of graph-cut [BVZ01] for final refinement.

3.1. Feature Extraction

To obtain a good feature representation of the meshes, we compute 11 types of geometric features, namely, the Gaussian curvature (GC) [MDS∗02], conformal factor (CF) [BCG08], principal curvature (PC) [GCO06], principal component analysis (PCA) of local face centres [KHS10], shape diameter function (SDF) [SSC08], distance from medial surface (DMS) [LZSCO09], average (integral) geodesic distance (AGD) [HSKK01], shape contexts (SC) [BMP02], spin images (SI) [JH99], heat kernel signature (HKS) [SOG09], and scale invariant HKS (SIHKS) [BK10]. They are calculated with different scales and normalisations. Most of these have been shown useful in earlier studies [KHS10, GZC15].

HKS and SIHKS have not yet been used in supervised mesh segmentation. They are effective point descriptors, designed for shape retrieval and correspondence [BK10]. As they are shown consistent in similar local regions, we hypothesise that they may be useful and include them in the feature set.

CF has been used in unsupervised techniques (e.g. [WWS∗13]), shown highly useful, but have not been used in supervised segmentation. When CF is computed on meshes in small regions with large curvatures (e.g. the propeller of the left plane or wing tip of the bird in Figure 3), CF is seriously distorted. To resolve this issue, we introduce a multi-resolution version of CF. We generate meshes with increasing number of smoothing iterations, using non-shrinking Laplacian smoothing [Tan95], and compute CF on these meshes. These new CFs alleviate the geometry issues, making the computed CF much more consistent across similar meshes (columns (c) and (d)).
output of the previous. This gives us $M'_i = \{F'_i, V'_i\}$ for iterations $i = 1, \ldots, 5$. To compute the conformal factor ($\Phi$) on the unsmoothed meshes we follow [BCG08], by solving:

$$L\Phi = K^T - K^{orig}$$

where $L$ is the Laplace-Beltrami operator, $K^{orig}$ is the Gaussian curvature of the mesh [MDS02] and $K^T$ is the target Gaussian curvature, which is the uniform curvature given by:

$$k^T_v = \left( \sum_{j \in V} \kappa_j \right) \frac{\text{area}(f)}{\sum_{f \in F} \text{area}(f)}$$

where $k^T_v$ is the target Gaussian curvature of vertex $v$, $\kappa_j$ is the $j^{th}$ element of $K^{orig}$, $F_v$ the set of faces that share vertex $v$ and $\text{area}(f)$ the surface area of face $f \in F$. With a smoothed mesh $M'_i$, the smoothed conformal factor $\Phi^T_i$ is computed by solving:

$$L\Phi^T_i = K_{orig}^T - K^T$$

where $\Phi^T_i$ is the desired conformal factor and $K^T_{orig}$ is the target Gaussian curvature for the smoothed mesh $M'_i$, and $K^T$ is the target Gaussian curvature of the original mesh $M$. The rationale of using $K^T$ in our formula instead of $K^T_{orig}$ (which is $K^{orig}$ for smoothed mesh $M'_i$) is because we would like the new CF to model the changes due to geometry smoothing alone. We do not want it to be affected by the underlying tessellation as in the original CF formula. This makes our CF more robust.

To show the impact of the proposed CF features we ran several experiments on the Princeton Segmentation Benchmark [CGF09]. Each experiment used one or many of the CF features to train a random forest (RF) classifier for mesh segmentation. Leave-one-out cross validation was performed on each set, with 3 replicates run per tested mesh. The average result for each experiment and each set is shown in Figure 4. This shows that, in the majority of cases, the proposed CF features have a large positive impact on the performance for classification, and also in some cases, just using a single smoothed CF feature is better than using the original CF feature.

In total, we obtain an 800-component feature vector for each face. The vector consists of 593 features from [KHS10], 1 original + 5 new CF features, 1 GC feature, 100 HKS features and 100 SIHKS features. They are used in all our techniques.

3.2. Fully Connected Neural Network

The first deep learning technique we analysed for mesh segmentation was conventional Fully Connected Neural Networks (NN). NNs consist of several fully connected layers followed by a classification layer to produce prediction probabilities. Each layer consists of a set of neurons, each with a weight and a bias. Each feature is passed to every neuron, which is activated based on the output of an activation function. The output from each neuron is then fed to every neuron in the next layer.

NNs iterates between two stages, a feed-forward and a back-propagation pass. The feed-forward pass is an unsupervised stage where all input feature is passed through the network. The neurons and their internal parameters across all layers determine the response. The back-propagation pass is supervised which minimizes the errors between the ground truth and the predicted labels. Error values are passed back through the network in order to tweak all the internal parameters. This process is repeated (typically several hundred times) in order to fine tune all parameters such that the network best describes the mapping between features and labels [LWL*16].

We further perform PCA to select the most important 50 principal components as features. The dimension reduction significantly reduces training time but does not affect the accuracy empirically. The reduced features are used to train a three-layer NN. The first layer has the same number of neurons as input features (50 neurons), and each subsequent layer reduces the number of neurons by half (25 and 12 neurons). The third layer is then fed into a softmax layer to compute an error cost. This then propagates back through the network to optimise the parameters. Once the network is trained, a new mesh is fed through for testing. The network returns a set of probabilities, specifying a class that a face may belong to, and allowing further refinement via graph-cut (Section 3.5). Results are reported in Table 1, PCA & NN column.

3.3. Autoencoder and Random Forest

Next, we analyse the use of Autoencoders (AE) for feature reduction and random forest (RF) for classification. Autoencoder is a type of artificial neural network, which aims to encode features for dimensionality reduction. The idea of AE is to learn the optimal representation of the original features through a network by recovering the original data through encoding and decoding [LWL*16]. It is powerful for its ability to non-linear dimension reduction. AEs can be stacked and optimised, so that the encoded features from
one AE can be fed to another for further reduction. Once trained, the encoded features are used to train a classifier.

In our technique, we pre-trained two AE layers separately. The first layer takes 800 features and reduces it to 400. The second layer takes the 400 encoded features and reduces it to 200. These encoded features are then used to train a softmax layer. Once it is trained, all three (two AEs and a softmax) layers are stacked together and re-trained. The model can then be used for testing. Results are reported in Table 1, AE & NN column.

The encoded features from the stacked AEs network can also be used to train an RF classifier. A RF classifier is a learning technique that takes a large set of random decision trees (we used 100 trees) and averages their prediction. It offers high performance in accuracy and speed whilst avoids overfitting. Results are reported in Table 1, AE & RF column. For both AE & NN and AE & RF, a graph-cut post-refinement (Section 3.5) is applied.

3.4. Multi-scale 1D Convolutional Neural Network

Here we discuss our new CNN mesh segmentation technique. We first outline the proposed multi-scale features, then describe the seven types of layers, and then the CNN architecture (Figure 5).

Multi-scale feature extraction

Existing techniques extract features per face [KHS10]. It has been shown in co-segmentation and relevant studies [HKG11, WWS13] that patch would also be useful for segmentation. We thus hypothesise that multi-scale features derived from a set of neighbouring faces would be useful. Given a face \( u \), we define a set of surrounding faces \( N^k(u) \) of \( u \) as the surrounding faces that are at most \( k \) step away. We then compute two extra feature vectors \( X^k \) where \( k = 2, 3 \) by averaging feature values of all faces in \( N^k(u) \). This leads to three feature vectors \( X^k \) where \( k = 1 \ldots 3 \) and \( X^1 \) is the original feature (Section 3.1). Each of these feature vectors \( X^k \) are trained separately by the proposed CNN network, and then merged before classification.

Convolution layers

In CNN, convolution (conv) layer simulates the organisation of humans' visual cortex, and the neurons responses of local receptive field. Given an input \( X \) (e.g. \( X^k \)), conv is often implemented as:

\[
\bar{X}_j = \sum_{i=1}^{C} W_{ij} \ast X_i + b_j, j = 1, \ldots, N
\]

where \( W_{ij} \) are the convolutional kernels, \( \ast \) represents the convolution operator, \( b_j \) the biases, \( C \) defines the number of input channels in \( X \), and \( N \) is the number of filters in the layer. All filters in the layer are convolved across \( X \) to produce \( N \) new feature maps, \( \bar{X} \).

Batch normalisation layers

typically follow convolutional layers to normalise the output. It allows much higher learning rates, and makes the network less sensitive to the initialisation [IS15].

ReLU layers

A Rectified Linear Units (ReLU) layer simulates the firing of a neuron by means of an activation function. We use \( \text{ReLU}(x) = \max(x, 0) \), where \( x \in X \), instead of others (e.g. sigmoid), as it is takes much less time to compute.

Max pooling layers

Pooling layers are used to down-sample the output features of the previous layer to better manage the high feature size. This typically performs after the convolutional layers. A filter (a maximum function, in our case) is passed across the feature map with a large stride to produce new maps.

Depth concatenation layer

To merge feature maps from different branches into a single feature map, we use a depth concatenation layer to concatenate via the depth dimension. This is pioneered in [SLJ15] to provide a mechanism for separate learning of features and later merging for classification. To the best of our knowledge, it is the first time to be used in mesh segmentation, and is essential for our multi-scale features learning.

Figure 5: The architecture of our multi-scale 1D CNN. Given an 800-dimension feature vector \( X^1 \) of a face \( u \), we compute a set of surrounding faces \( N^k(u) \) that are \( k - 1 \) steps away (\( k = 2, 3 \)). We average all features of all faces in \( N^k(u) \) leading two extra feature vectors \( X^2, X^3 \). These multi-scale features \( X^1, X^2, X^3 \) are used in the CNN, and trained separately through the network. They are then concatenated by the depth concatenation layer before reaching the fully connected and classification layers. Each conv layer is followed by a batch normalization and a ReLU layer, and the first fully connected layer is followed by a ReLU layer.
Fully connected layer. Fully connected layers (like NNs) have full connections to all activations in the previous layers, and act as the function approximators to learn the non-linear mapping.

Softmax loss layer. A softmax loss layer is used for classification. It penalises any differences between the predicted and true labels, so that the network parameters can be tuned during back propagation. Softmax predicts a single class out of $K$ classes.

Architecture and training. The architecture of our CNN (Figure 5) allows the three individual feature vectors to be trained independently before being merged back for the fully connected layers and classification. All three feature vectors undergo the same training process, with their own distinct layers and parameters. In our implementation, each convolutional layer is followed by a batch normalisation and a ReLU layer. For clarity, these two layers are not shown in Figure 5.

First, each feature vector $X^k$ is separately passed through a convolutional layer to extract some low-level features. Sixteen $15 \times 1$ filters are used to produce sixteen new feature vectors, and padding is used to ensure the vectors remain a constant length. Once the output is normalised and passed through a ReLU layer, it is then max-pooled to reduce the size in half (400 components, 16 channels). This process is repeated with a convolutional layer with thirty-two $11 \times 1$ filters. After the final pooling stage, each of the three branches provides thirty-two 200 component feature vectors.

The three branches are then merged, via a depth concatenation layer, producing ninety-six 200 component feature vectors. These are passed through two fully connected layers. The first layer further increases the number of feature channels to 172, and the second layer reduces it back to the number of distinct classes in the set. The output from the final layer can be classified by the softmax layer, and the loss can be computed.

Similar to conventional NNs, there are two learning passes. The feed-forward pass sends all the features through the network to produce a label prediction. Then the back-propagation passes the prediction error back through the network, updating parameters accordingly in order to reduce the error. These passes are repeated for a set number of iterations (we set to 50, and use a learning rate in the log-space between -2 and -4).

After training, the network produces label probabilities for each test mesh, which are subsequently used for graph-cut refinement. Results are reported in 1D CNN column, Table 1.

3.5. Graph-Cut Refinement

A trained model (NN, RF, CNN) can predict a label for a face with a set of probabilities. The probability indicates how likely a face belongs to a particular class. However, inconsistencies of predicted labels can arise between adjacent faces on the mesh because the classification does not take face adjacency into account. This causes incorrect segmentations and reduced accuracies. Here, we utilise the multi-label alpha-expansion graph-cut technique [BVZ01] to refine the segmentation results.

Let $u, v \in T$ be two faces in a mesh, where $T$ is the set of all faces. Let $N_u$ be the set of neighbouring faces of $u$. We can optimise the labels of all $u \in T$ by solving:

$$
\min_{l_u \in \mathbb{T}} \sum_{u \in \mathbb{T}} \xi_D(u, l_u) + \lambda \sum_{u \in \mathbb{T}, v \in N_u} \xi_S(u, v, l_u, l_v, f_u, f_v)
$$

where $\lambda$ is a non-negative constant used to balance the influence of the two terms. $\xi_D(u, l_u) = -\log(p_u(l_u))$ is a data term that penalises low probability of assigning a label $l_u$. The second term $\xi_S$ incurs a large penalty when the dihedral angle between two adjacent faces is small (i.e. the faces cause a concavity) or the distance between two features is high, and is given by:

$$
\xi_S(u, v, l_u, l_v, f_u, f_v) = \begin{cases} 
0, & \text{if } l_u = l_v \\
\psi(u, v, f_u, f_v), & \text{otherwise}
\end{cases}
$$

where $\psi(u, v, f_u, f_v) = -\log(\theta_{uv}/\pi)\psi_{uv}$ is the cost based on the dihedral angle ($\theta_{uv}$), edge length ($\psi_{uv}$). This formulation has been applied commonly in [GZC15, SvKK11, MXLH13].

Here, we propose to modify $\psi(u, v, f_u, f_v) = -\log(\theta_{uv}/\pi)\psi_{uv} + \omega||f_u - f_v||_2$, by introducing a geometric feature term. It promotes
similar classification label if the Euclidean distance between features \( f_u, f_v \) of face \( u \) and \( v \) is small. A constant \( (\omega) \), is used to balance the weight of the concavity and feature terms. We use AGD (Section 3.1) as the feature \( f \) as it helps to smooth out inconsistent labels, and improves the refinement accuracy. An evaluation of this approach will be discussed in Section 4.

4. Experiments and Results

In this section, we discuss the experiments and evaluation measure for these deep learning techniques on mesh segmentations. Then we discuss the results and put forward our observations.

All experiments were conducted on the Princeton Segmentation Benchmark [CGF09], which is a widely used dataset for evaluating mesh segmentation techniques [KHS10, GZC15]. The dataset contains 19 sets of meshes with 20 meshes per set. Similar to [SQX∗16], we omit three sets (Bust, Bearing and Mech (further discussion is provided at the end of this section)) from our results in Table 1, because these sets are either inconsistently labelled or contain meshes with too much variance within the set. For each set, we perform leave-one-out cross validation: a model is trained on 19 out of 20 meshes, whilst the remaining mesh is used for testing. This is repeated for every mesh in a set. The final result is the average of all individual accuracies. Similar to [KHS10, GZC15, SvKK∗11], we use the accuracy measure:

\[
\text{Accuracy}(l, gt) = \frac{\sum_{t \in T} a_t \delta(l_t = gt_t)}{\sum_{t \in T} a_t}
\]

where \( a_t, l_t \), and \( gt_t \) are respectively the area, the predicted label and the ground truth label of triangle \( t \). \( \delta(l_t = gt_t) \) is assigned to 1, if the predicted label is the same as the ground truth; otherwise 0.

Results & Discussions All experimental results are shown in Table 1, where the columns PCA & NN, AE & RF, AE & NN and 1D CNN correspond to the techniques discussed in Sections 3.2-3.4. Some visual results of our 1D CNN technique is shown in Figure 6. We also faithfully reimplement [GZC15] (state of the art), which uses a deep learning architecture, and show their results in the TOG15 column.

Table 1: Experimental results. Bold: highest accuracy; underlined: accuracy within 0.5% of the highest. Our 1D CNN technique performs comparably or better in majority of the sets. In the comparison row, the first number is the number of sets with the highest accuracy or accuracy comparable (within 0.5%) to the highest. The number in [ ] is the number of sets with an accuracy higher than all other methods more than 0.5%.

|       | PCA & NN | AE & RF | AE & NN | TOG15 [GZC15] | 1D CNN |
|-------|---------|---------|---------|----------------|--------|
| Airplane | 92.97   | 92.62   | 92.53   | 94.56          | 95.30  |
| Ant    | 95.15   | 95.17   | 95.15   | 97.55          | 98.43  |
| Armadillo | 88.21   | 88.43   | 87.79   | 90.90          | 92.24  |
| Bird   | 95.14   | 88.93   | 88.20   | 86.20          | 89.98  |
| Chair  | 95.55   | 95.69   | 95.61   | 97.07          | 98.04  |
| Cup    | 95.09   | 97.95   | 97.82   | 98.95          | 98.78  |
| Fish   | 94.41   | 96.21   | 95.31   | 96.16          | 95.97  |
| Fourleg | 83.61   | 83.99   | 82.32   | 81.91          | 85.05  |
| Glasses | 94.22   | 96.57   | 96.42   | 96.95          | 96.69  |
| Hand   | 78.33   | 73.76   | 70.49   | 82.47          | 86.78  |
| Human  | 87.03   | 86.69   | 81.45   | 88.90          | 88.32  |
| Octopus | 96.93   | 96.99   | 96.52   | 98.50          | 98.16  |
| Plier  | 93.75   | 92.59   | 91.53   | 94.54          | 95.29  |
| Table  | 99.22   | 99.18   | 99.17   | 99.29          | 99.17  |
| Teddy  | 98.07   | 98.24   | 98.20   | 98.18          | 98.12  |
| Vase   | 79.73   | 82.07   | 80.24   | 82.81          | 82.00  |
| Average| 91.09   | 91.57   | 90.61   | 92.81          | 93.64  |
| Comparison | 2 [0] | 4 [0]   | 2 [0]   | 8 [2]          | 14 [8] |
A direct comparison of our 1D CNN and [GZC15] shows that, on average, our 1D CNN method has higher accuracy. Among all the 16 sets, our 1D CNN performs better than TOG15 [GZC15] on 8 sets (Airplane, Ant, Armadillo, Bird, Chair, Fourleg, Hand and Piler), Figure 7, and performs worse on 2 sets (Human and Vase). For the other sets, the accuracy of both is comparable where the difference is all within 0.5% (underlined, Table 1).

We further investigate the ones with poor results. We observe two problems. First, the Human set is badly and non-consistently labelled in general, and there are insufficient support to train a proper model (Figure 8, see arrows). This is challenging for any machine learning techniques, making both 1D CNN and [GZC15] fail to score high (over 90%). Second, the Vase set contains meshes that are significantly different from the rest. This can be seen in Figure 8 (see arrows), where there is a mesh (back row, second from right), which contains a segment usually defined as the base of a vase (purple segment), in place of the part which is typically on the top of the vase (blue segment in other meshes).

For completeness, the results for the sets we omitted from Table 1 are as follows. Accuracies of 88.67%, 70.06% and 88.53% [GZC15], and 89.69%, 61.97% and 88.14% (our 1D CNN) were achieved for the Bearing, Bust and Mech sets respectively. These sets were omitted due to ground truth inconsistencies (Figure 9) and lack of sufficient training data [SQX∗16]. These are reflected in the lower accuracies of both methods.

Our conclusion is that if there are sufficiently large number of good meshes and consistent labels across the set, the new features and 1D CNN architecture is useful (with improvement overall, in terms of number of sets that achieved higher accuracy). Compared to [GZC15], our technique does not require parameter tuning for features reshaping or sampling to 2D images.

Table 2: Experimental results of graph-cut refinement. **Bold:** proposed technique consistently performs better than traditional one.

| Set       | Unrefined | Traditional | Proposed |
|-----------|-----------|-------------|----------|
| **Graph-cut** |           |             |          |
| Airplane  | 95.12     | 95.14       | 95.30    |
| Ant       | 98.41     | 98.42       | 98.43    |
| Armadillo | 92.18     | 92.18       | 92.24    |
| Bird      | 89.94     | 89.94       | 89.98    |
| Chair     | 98.01     | 98.02       | 98.04    |
| Cup       | 98.59     | 98.60       | 98.78    |
| Fish      | 95.94     | 95.96       | 95.97    |
| Fourleg   | 84.72     | 84.77       | 85.05    |
| Glasses   | 96.61     | 96.64       | 96.69    |
| Hand      | 86.63     | 86.66       | 86.78    |
| Human     | 88.28     | 88.31       | 88.32    |
| Octopus   | 98.14     | 98.12       | 98.16    |
| Plier     | 95.30     | 95.28       | 95.29    |
| Table     | 99.14     | 99.14       | 99.17    |
| Teddy     | 98.08     | 98.08       | 98.12    |
| Vase      | 81.97     | 81.99       | 82.00    |
| **Average** | 93.57     | 93.58       | 93.64    |
Figure 9: **Ground truth** examples from the 3 omitted sets (Mech, Bearing, Bust). Label inconsistencies can be seen throughout. Mech (a) shows segment inconsistencies where cylindrical shapes are labelled both purple and green (front row, centre and back row). Also, the blue segments shown on the two meshes are the only blue segments in the set, and are both topologically dissimilar. Bearing (b) shows segment inconsistencies where similar shaped regions (threaded parts) have several different labels. Bust (c) shows poor segment boundaries where the neck extends on to the clothing (front row, centre). Additionally, it contains inconsistent segments where the hats and hair are one segment but back left has a clothing segment over the top of the head. Finally, a few labels are missing throughout. For example, not all lips, noses and eyes are properly and consistent labelled. Some models are missing some of these segments and others are missing them all from the ground truth (e.g. nose of back right, eyes of 4 of the shown models, lips of front left and back right). Arrows show examples of badly or inconsistent ground truth labelling.

Next we analyse the performance of our other deep learning techniques, PCA & NN, AE & RF, and AE & NN. We note that, although they perform worse than CNN techniques in general, their performance is only marginally worse. In sets (Fish and Teddy) that have consistent and well-defined labels, AE & RF performs better than CNN techniques. This is interesting as these neural network models consists of 2-3 layers, and require much shorter training time than CNN techniques.

We also compare the use of two different classifiers AE & RF and AE & NN on the same set of encoded features. As shown in Table 1, the results from using an RF classifier are almost exclusively better than using only the NN model alone. This may be explained by the fact that both the AE network and the RF classifier are two different techniques, and are separately trained. There would be complementary improvement overall.

Finally we analyse the performance of our proposed refinement technique, using graph-cut in conjunction with a feature included in the smoothness term. We have compared this to unrefined results and also results obtained from a traditional graph-cut refinement technique. Table 2 shows the results on all sets in the Princeton Segmentation Benchmark [CGF09]. Our method consistently outperforms the existing technique used in other literature, and also consistently improves upon the unrefined result, except for the Plier set. This shows the usefulness of including a feature in the smoothness term.

**Computation Time** Our experiments were carried out on a workstation with 32 Xeon 2.3GHz CPUs (total 64 cores). We run all our experiments on both CPUs and GPUs so as to collect results as fast as possible, along with hardware upgrade to 512GB memory and one Nvidia Titan X (Pascal) GPU card over the past year. To provide a general timing, to train a model with 6 meshes (each with 20-30K faces) with 6 classes, it would take 3 minutes, 30 minutes, 25 minutes, 4 hours, and 4 hours, respectively, for PCA & NN, AE & RF, AE & NN, 1D CNN and our implementation of [GZC15].

In general, CNN techniques take longer time to train due to the deeper network and larger number of parameters. Our time for our 1D CNN is comparable to [GZC15] because we train the network to a converged state in a shorter time, yet, our technique can handle over three times more features than [GZC15], because of the use of 1D filters and multi-branching techniques (separate training).

**Challenges & Limitations** In our experiments, we observe that some sets are very challenging (e.g. the Bust set [CGF09]). There are an insufficient number of meshes in the set to cover a large variation of shape and topology, leading to poorly trained models.

Further, some of the segmentation boundaries in the ground truth labels are not well-defined or consistent (e.g. Human, Bearing, Bust). Additionally, some sets have meshes with segments missing from the ground truth (Human, Bust), Figure 9. This makes the accuracy measure less meaningful. We believe that a more accurate ground truth could improve the accuracies in these sets.

5. **Conclusion**

In this paper, we have shown a novel way of using CNNs to perform automatic mesh segmentation. Instead of casting 3D geometric features into 2D images and using 2D filters to fit an image-based CNN pipeline, we show that the use of 1D data and filters can perform better, but avoids the problem of parameter tuning for reshaping and resampling features. The time cost is comparable but ours can support a more complex and deeper network. Further, we have also shown a novel way of computing more consistent and robust conformal factor (a curvature based geometry feature), such that it is less sensitive to small areas of large curvature. We also propose an addition of a feature similarity term into the graph-cut refinement, which provides better overall segmentation.
Additionally, we performed a comprehensive and comparative study of several deep learning techniques for mesh segmentation. We showed that simpler network architectures (e.g., AE’s, NN’s and RF’s) can still perform reasonably well using the same set of geometric features, compared to more complex CNN models. They also have a significantly shorter training time. This study suggests that if only a reasonable (not perfect) segmentation is required for downstream application, AE, NN and RF would be a good choice, given the time and efforts to train the models.

References

[AFS06] Atten E., M., Falcidieno B., Spagnuolo M.: Hierarchical mesh segmentation based on fitting primitives. Vis. Comput. 22, 3 (2006), 181–193.

[BG08] Ben-Chen M., Gotsman C.: Characterizing shape using conformal factors. In 3D Obj. Retrieval (2008), pp. 1–8.

[BK10] Bronstein M. M., Kokkinos I.: Scale-invariant heat kernel signatures for non-rigid shape recognition. In Proc. IEEE Conf. CVPR (2010), pp. 1704–1711.

[BLVD11] Benhabiles H., Lavoué G., Vandanborre J.-P., Daoudi M.: Learning Boundary Edges for 3D-Mesh Segmentation. Comput. Graphics Forum 30, 8 (2011), 2170–2182.

[BMP02] Belongie S., Malik J., Puzicha J.: Shape matching and object recognition using shape contexts. IEEE Trans. Pat. Anal. & Mach. Intell. 24, 4 (2002), 509–522.

[BVZ01] Boykov Y., Veksler O., Zabih R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pat. Anal. & Mach. Intell. 23, 11 (2001), 1222–1239.

[CGF09] Chen X., Golovinskiy A., Funkhouser T.: A benchmark for 3D mesh segmentation. ACM Trans. on Graphics 28, 3 (2009).

[CLL*15] Chen X., Li J., Li Q., Gao B., Zou D., Zhao Q.: Image2scene: Transforming style of 3d room. In Proc ACM Int. Conf on Multimedia (2015), pp. 321–330.

[CM02] Comaniciu D., Meer P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pat. Anal. & Mach. Intell. 24, 5 (2002), 603–619.

[CZL*15] Chen X., Zhou B., Lu F., Wang L., Bi L., Tan P.: Garment modeling with a depth camera. ACM Trans. on Graphics 34, 6 (2015), 203.

[GCO06] Gal R., Cohen-Or D.: Salient geometric features for partial shape matching and similarity. ACM Trans. on Graphics 25, 1 (2006), 130–150.

[GF08] Golovinskiy A., Funkhouser T.: Randomized cuts for 3D mesh analysis. ACM Trans. on Graphics 27, 5 (2008).

[GZC15] Guo K., Zou D., Chen X.: 3d mesh labeling via deep convolutional neural networks. ACM Trans. on Graphics 35, 1 (2015), 3:1–3:12.

[HKG11] Huang Q., Koltun V., Guibas L.: Joint shape segmentation with linear programming. ACM Trans. on Graphics 30, 6 (2011), 125.

[HSKK01] Hilaga M., Shinagawa Y., Kohmura T., Kunii T. L.: Topology matching for fully automatic similarity estimation of 3D shapes. In Proc. ACM SIGGRAPH (2001), pp. 203–212.

[HIZ*15] Hu R., Zhu C., van Kaick O., Liu L., Shamir A., Zhang H.: Interaction context (icon): Towards a geometric functionality descriptor. Proc. ACM SIGGRAPH ASIA 34, 4 (2015), 83:1–83:12.

[IS15] Ioffe S., Szegedy C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML (2015), pp. 448–456.
[YZX’04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with poisson-based gradient field manipulation. ACM Trans. on Graphics 23, 3 (2004), 644–651.