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

David George, Xianghua Xie, Gary KL Tam

Swansea University, United Kingdom

Abstract

There is an increasing interest in applying deep learning to 3D mesh segmentation. We observe that 1) existing feature-based techniques are often slow or sensitive to feature resizing, 2) there are minimal comparative studies and 3) techniques often suffer from reproducibility issue. This study contributes in two ways. First, we propose a novel convolutional neural network (CNN) for mesh segmentation. It uses 1D data, filters and a multi-branch architecture for separate training of multi-scale features. Together with a novel way of computing conformal factor (CF), our technique clearly out-performs existing work. Secondly, we publicly provide implementations of several deep learning techniques, namely, neural networks (NNs), autoencoders (AEs) and CNNs, whose architectures are at least two layers deep. The significance of this study is that it proposes a robust form of CF, offers a novel and accurate CNN technique, and a comprehensive study of several deep learning techniques for baseline comparison.

Keywords: Mesh Segmentation, Mesh Processing, Deep Learning

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 [1], matching [2], editing [3, 4], deformation [5] and modelling [6]. 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. They first compute features (e.g. shape diameter function [7], approximate convexity [8] and curvature [9]) for the faces of the meshes, and uses an optimisation technique to produce the segmentation results. Notable techniques include k-means [10], mean-shift clustering [11] and normalized and randomized cuts [12]. A detailed survey can be found in [13, 14]. Given the large shape variability of segments, more recent approaches consider consistent co-segmentation of a collection of shapes, where class labels are consistent throughout the set [15, 16].

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 [17] . 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 [14]), supervised techniques are gaining focus. The work by [18] pioneered to apply joint boosting on a large set of shape features for effective labelling. Recently, [19] used an extreme learning machine (a single layer wide neural network), then later expanded it to a two-layer network [20], however the performance is marginally better than traditional shallow classifiers [21]. Later [22] 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 feature-based mesh segmentation techniques use geometric features derived mostly from one face (except [18] which also computes a few geometric features curvatures, PCAs with different radii). As such, spatial scale information is mostly not considered in their learning architecture. A feature-based deep learning *network architecture* for segmentation that considers multi-scale

geometric features derived from a set of local faces has not been fully explored. 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-2M 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 several research questions. (i) Compared to existing learning techniques that use features mostly defined per face, can a deep learning architecture, considering multi-scale features derived from a set of faces, be useful? (ii) Compared to [22] 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 would avoid the tuning of image size, and improve efficiency and performance of CNNs. (iii) Finally, how much improvement can CNNs have over existing deep learning techniques. Our contributions of this paper are four-fold:

- First, we introduce a novel and accurate CNN technique for 3D mesh segmentation. We introduce a multi-branch network architecture that separately trains features of three 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 remove most of the assumed feature relationships that are imposed by an image-based CNN when reshaping the feature vector into a 2D image. Our novel technique clearly outperforms existing feature-based CNN technique [22].
- Second, we propose a novel feature vector of conformal factor (CF) which is computed from incremental smoothing of geometry. It is less sensitive to high curvature noise, and consistently provides higher segmentation accuracy than [23] alone.
- Third, we perform a comprehensive comparison of deep learning techniques (at least two layers deep) for supervised mesh segmentation, specifically Neural Networks (NNs), Autoencoders (AEs) and CNNs [22], showing the strengths and limitations of each technique by comparing their accuracies.
- Finally, data and our implementations of all the compared techniques are made publicly available for the research community.

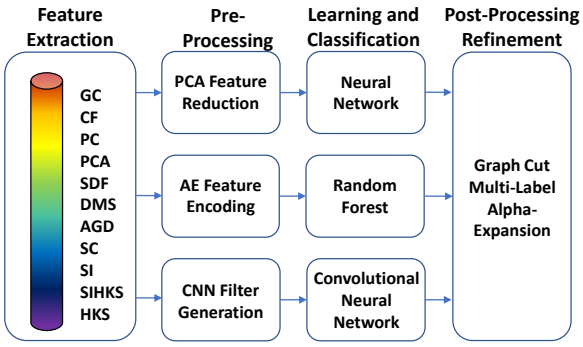


Figure 1: An overview of the stages involved in each of the techniques. Each technique includes all four stages: feature extraction, pre-processing (by encoding or reduction), deep learning and classification, and graph-cut post-processing stage.

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 [10, 7]. They often performed clustering on geometric features (e.g. shape diameter function [7], geodesic distances [24], curvature [25]) or partitioned based on properties that can be derived from the mesh itself (e.g. skeleton [7], convexity [8], fitting primitive shapes [26]). Many of these ideas have been shown effective, giving rise to a wide range of shape descriptors, and segmentation techniques, supporting many downstream applications [13, 14]. 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 [15, 27, 28, 29, 30]. For example, all legs in a chair set should be labelled the same. Such constraint is

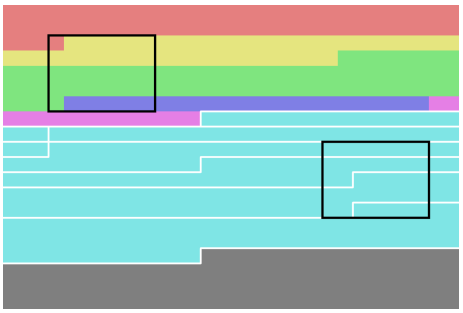
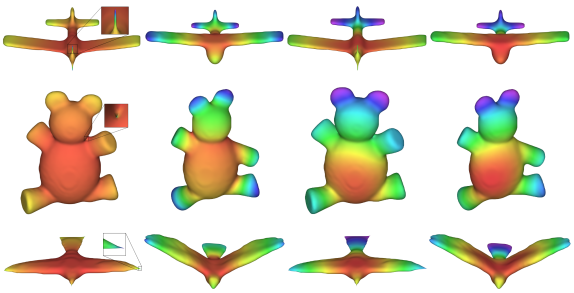


Figure 2: Example of a 30x20 image produced by reshaping the 600 features in [22]. Each colour represents a distinct feature (PC, PCA, SDF, DMS, AGD, SC, SI from top to bottom) and the white boarders outline the 6 different SC histograms. The black rectangles are examples of 7x5 convolutional filters being passed over the image. In the examples, the filters would infer relationships between 4 different features or arbitrary bins in 4 different SC histograms, which in both cases have no correlation.



(a) (b) (c) (d)

Figure 3: Visual comparison of conformal factor (CF) features. Columns (a) and (b) are the original CF, columns (c) and (d) are CF after one stage of Laplacian-smoothing. CF is sensitive and can be easily distorted by small regions of large curvature (propeller tips of the plane, noise on shoulder of teddy and wing tips of the bird in column (a)). Non-shrinking Laplacian-smoothing can alleviate the geometry issues, making the computed CF much more consistent across similar meshes (columns (c) and (d))

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 [31]. More importantly, segments of a shape are often associated to its functionality - a high-level understanding of shapes [17]. Therefore, there is an increasing interest in supervised segmentation techniques, trying to learn a high-level mapping directly from feature to segment.

Supervised Segmentation. These 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. [14]).

Existing supervised techniques rely on local features. The work by [18] 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 [32] ranks a large pool of features in order to detect the optimal segment boundaries for a given mesh, and an extreme learning machine was trained to classify labels using one [19] and two layers [20]. However, supervised methods can perform poorly on very complex meshes, due to insufficient training data or large variations within label classes [19, 22].

Recently, [22] extended the CNN idea to 3D segmentation. They reshape a large pool of geometric features into a matrix resembling that of a 2D image, fitting the 2D image-based CNN pipeline, and then train a CNN on these "images" using the ground truth labels. The technique shows good performance, however, the reshaping and the use of 2D filters may infer relationships between adjacent rows of features that may have no correlation. As Figure 2 shows, passing a convolutional filter over such an "image" would unavoidably infer relationships between (up to 5) unrelated features regardless of the position of the filter.

From the literature, we have two further observations. First, existing feature driven techniques use local features developed by the influential work [18]. These features are mostly defined per face (a few are normalized by different geodesic radii for smoothing purposes), as such, there is no spatial scale information included in the architecture. To the best of our knowledge, a feature-based deep learning network architecture that considers multi-scale geometric features derived from a set of local faces has not been fully explored in supervised mesh segmentation. We hypothesise that multi-scale features would be useful because face-based [15] and patch-based techniques [29] have both shown good performance in co-segmentation. Second, whilst deep learning is useful, there is not much analysis as how CNNs perform compared to other techniques in the deep learning family.

In this paper, we show that by using multi-scale features, treating them as separate 1D vectors per scale, and applying multi-branch 1D CNN filters through the network, we avoid the parameter tuning problem of re-

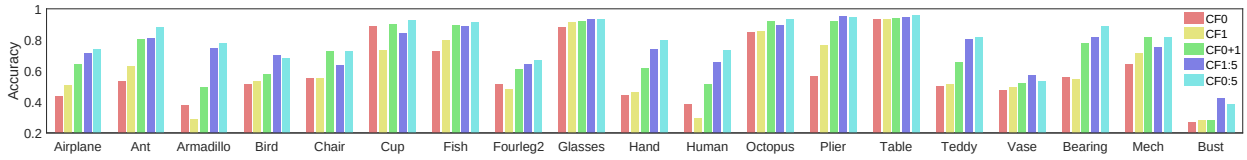


Figure 4: Accuracies of running leave-one-out cross validation on all sets using Random Forest classifier. CF0, CF1, CF0+1, CF1:5 and CF0:5 respectively denote the original CF [23], CF after one stage of Laplacian-smoothing, combination of CF0 and CF1, combination of all smoothed CF features, and combination of all CF (including CF0). The chart shows that the new smoothed CF features improve upon the original in most cases, and also further improves when they are combined with the original CF.

shaping a 2D matrix. Our method clearly out-performs existing work [22] in accuracy. Further, we provide comprehensive evaluation of various deep learning techniques, and show how CNNs, though more complex, can improve performance over simpler architectures (NNs, AEs). It is worth noting that we have found existing methods lack reproducibility. Some existing work have not provided any or complete experimental code. Despite using the exact architecture and settings stated, some high performing results are hard to reproduce.

Concurrent to our work, there are recent efforts focusing on different kind of inputs, such as point clouds [33, 34], octrees [35], multiple projected images [21], graphs [36] or geometry images [37]. Differing from these, our study focuses on feature-based approach, and investigates how deep learning can improve 3D segmentation using insightful geometry features that are developed in the past decade.

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 Networks (NNs), Autoencoders (AEs) + Random Forests (RFs), and Convolutional Neural Networks (CNNs), in Sections 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 [38] 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) [9], conformal factor (CF) [23], principal curvature (PC) [25], principal component analysis (PCA) of local face centres [18], shape diameter function (SDF) [7], distance from medial surface (DMS) [39], average geodesic distance (AGD) [24], shape context (SC) [40], spin images (SI) [41], heat kernel Signature (HKS) [42], and scale invariant HKS (SIHKS) [43]. They are calculated with different scales and normalisations. Most of these have been shown useful in earlier studies [18, 22].

HKS and SIHKS have not yet been used in supervised mesh segmentation. They are effective point descriptors, designed for shape retrieval and correspondence [43]. 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. [29]) and been shown to be highly useful, but has 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 [44], and compute CF on these meshes. These new CFs alleviate the distortion and are more consistent (Figure 3).

Let $M = \{F, V\}$ be a mesh, where F and V are the faces and vertices of the mesh. We first compute 5 iterations of non-shrinking Laplacian smoothing, where the input of the next iteration is the output of the previous. This gives us $M_i^s = \{F_i^s, V_i^s\}$ for iterations $i = 1, \ldots, 5$. To compute the conformal factor (CF) (Φ) on the unsmoothed meshes we follow [23], by solving:

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

where L is the Laplace-Beltrami operator, K^{orig} is the GC of the mesh [9] and K^T is the target GC, which is the uniform curvature given by:

$$k_v^T = \left(\sum_{j \in V} \kappa_j\right) \frac{\sum_{f \in F_v} \frac{1}{3} area(f)}{\sum_{f \in F} area(f)}$$

where k_v^T is the target GC of vertex v, κ_j is the j^{th} element of K^{orig}, F_v the set of faces that share vertex v and $area(f)$ the surface area of face $f \in F$. With a smoothed mesh M_i^s, the smoothed CF Φ_i^s is computed by solving:

$$L\Phi_i^s = K_i^{sT} - K^T$$

where Φ_i^s is the desired CF and K_i^{sT} is the target GC for the smoothed mesh M_i^s, and K^T is the target GC of the original mesh M. The rationale of using K^T in our formula (instead of K^{orig} of the smoothed mesh M_i^s) is that 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, making our CF more robust.

To show the impact of the proposed CF features we ran several experiments on the Princeton Segmentation Benchmark (PSB) [14]. Each experiment used one or many of the CF features to train a RF classifier for mesh segmentation. Leave-one-out cross validation was performed on each set, with 3 replicates ran per tested mesh. The results for each experiment for 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 [18], 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 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 and 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 [45].

We perform PCA to select the most important 50 principal components as features. This 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 AEs for feature reduction and a RFs for classification. An AE is a type of artificial NN, 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 [45]. 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 can also be used to train a 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 fea-

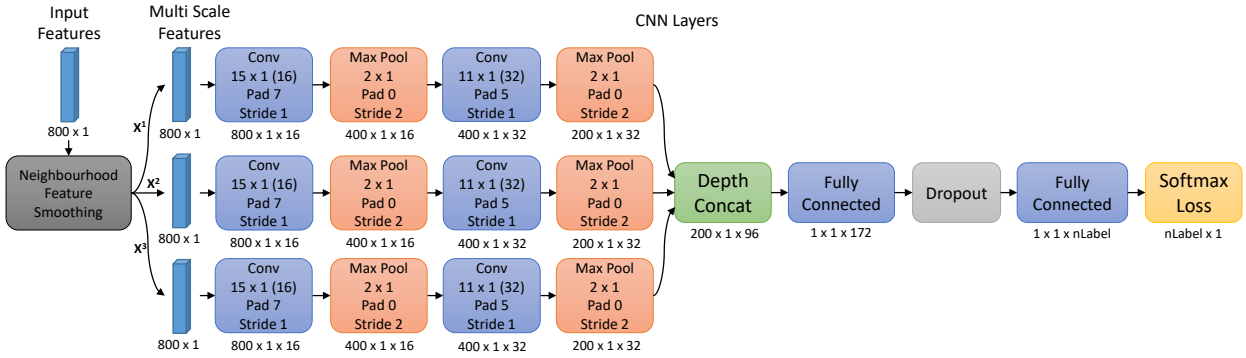


Figure 5: The architecture of our multi-scale 1D CNN. Given an 800-dimension feature vector $\boldsymbol{X}^1$ of a face u, we compute a set of surrounding faces $\mathcal{N}^k(u)$ that are $k-1$ steps away ($k = 2, 3$). We average all features of all faces in $\mathcal{N}^k(u)$ leading two extra feature vectors $\boldsymbol{X}^{2,3}$. These multi-scale features $\boldsymbol{X}^{1...3}$ 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 leaky ReLU layer, and the first fully connected layer is followed by a leaky ReLU layer.

tures, then describe the types of layers we use, and finally the CNN architecture (Figure 5).

Multi-scale feature extraction. Existing techniques extract features mostly per face [18]. It has been shown in co-segmentation and relevant studies [16, 29] 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, and should be considered in a network architecture. Given a face u, we define a set of surrounding faces $\mathcal{N}^k(u)$ of u as the surrounding faces that are at most $k-1$ step away. We then compute two extra feature vectors $\boldsymbol{X}^k$ where $k = 2, 3$ by averaging feature values of all faces in $\mathcal{N}^k(u)$. This leads to three feature vectors $\boldsymbol{X}^k$ where $k = 1...3$ and $\boldsymbol{X}^1$ is the original feature (Section 3.1). Each of these feature vectors $\boldsymbol{X}^k$ are trained separately by the proposed CNN network, and then merged before classification.

Network layers. We now discussed all network layers used in our CNN architecture, and their functions.

- **Convolutional layers** simulate the organisation of humans' visual cortex, and the neurons responses of local receptive field. They consist of filters, which are convolved over the input to produce new feature maps as outputs, one for each filter.
- **Batch normalization layers** typically follow conv layers to normalise the output. It allows much higher learning rates, and makes the network less sensitive to the initialisation [46].
- **Leaky ReLU layers** A Rectified Linear Unit (ReLU) layer simulates the firing of a neuron by means of an activation function. We use the leaky ReLU variation [47] instead of a regular ReLU as having a small negative gradient will stop cases where all inputs are negative and the activation produces zero. We set the slope to be 0.2 in our experiments.
- **Max pooling layers** Pooling layers are used to downsample the output features of the previous layer to better manage the high feature size, these typically performs after the conv layers.
- **Depth concatenation layers** To merge feature maps from different branches into a single feature map, we use a concatenation layer to concatenate via the depth dimension. This is pioneered in [48] to provide a mechanism for separate learning of features and later merging for classification.
- **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.
- **Dropout layers** are included in to regularise the network to reduce overfitting [49]. It works by randomly selecting neurons to be ignored during this pass of the training. This is done by ignoring the weights assigned to them during the forward pass and not updating their weights on the back pass. We set 50% of neurons to be randomly ignored in our experiments.
- **Softmax layers** are activation functions typically used for classification. The function computes the exponential of the input and divide them by the sum of all exponential values, giving prediction probabilities as output. Coupled with a loss function, they penalise differences between the predicted and true labels, and

update the network parameters in back propagation.

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

The rationale behind our architecture design stems from several research questions:

Q1 Can we reduce the unnecessary inference between unrelated features and improve performance?

Q2 How can we make good use of multi-scale features in a deep learning architecture?

Q3 How can we train such a network in a practical time frame given the increased features and branches?

One possible answer to eliminate inference between unrelated features (Q1) is a fully connected NN. However, such a network would lead to impractical training times (due to the increased feature sizes and number of scales). Our compromise is to use a 1D network instead. Though it does not fully resolve the issue in [22], it avoids guessing the parameter for image resizing, and alleviates unnecessary inference between number of unrelated features from (at most) 5 to 2 per each filter. Further, because both face-level [18, 22] and patch-level features [28, 30] have been shown useful alone, we hypothesise that features from different scales can be trained and analysed independently (Q2). Inspired by GoogLeNet [48], we separate and train features of each scale in individual branch, formulate our architecture as a multi-branch network and concatenate them through the depth channel. This also reduces training time (Q3). Fully connected layers are used to get the final predictions. We introduce a second fully connected layer due to the increased amount of data from all 3 branches. Finally the addition of batch normalization layers and a dropout layer allows the network to better generalize and reduce overfitting. We opted to use 3 branches as it shows highest accuracy with the quickest training time empirically (Q3). An evaluation on the use of 1-4 branches can be found in Section 4.

Training. First, each feature vector $\boldsymbol{X}^k$ is separately passed through a conv layer to extract some low-level features. Sixteen 15×1 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 leaky ReLU layer, it is then max-pooled to reduce the size in half (400 components, 16 channels). This process is repeated with a conv layer with thirty-two 11×1 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 then passed through a fully connected layer with 172 neurons, and then through a dropout layer with a 50% dropout. Finally, a fully connected layer with the number of neurons equal to the number of classes in the set is used, with a softmax activation function. The output of the softmax layer can then be used to compute the loss (we use categorical cross entropy) in order to back-propagate through the network to update parameters.

Similar to a NN, there are two learning passes. The feed-forward pass produce a label prediction, and the back-propagation updates parameters to reduce the prediction error. These passes are repeated for a set number of iterations (set to 50, using a learning rate in the log-space between -2 and -4 and a momentum of 0.9). The label probabilities that are produced after training are subsequently used for graph-cut post-refinement. Results are reported in **1D CNN** column, Table 1.

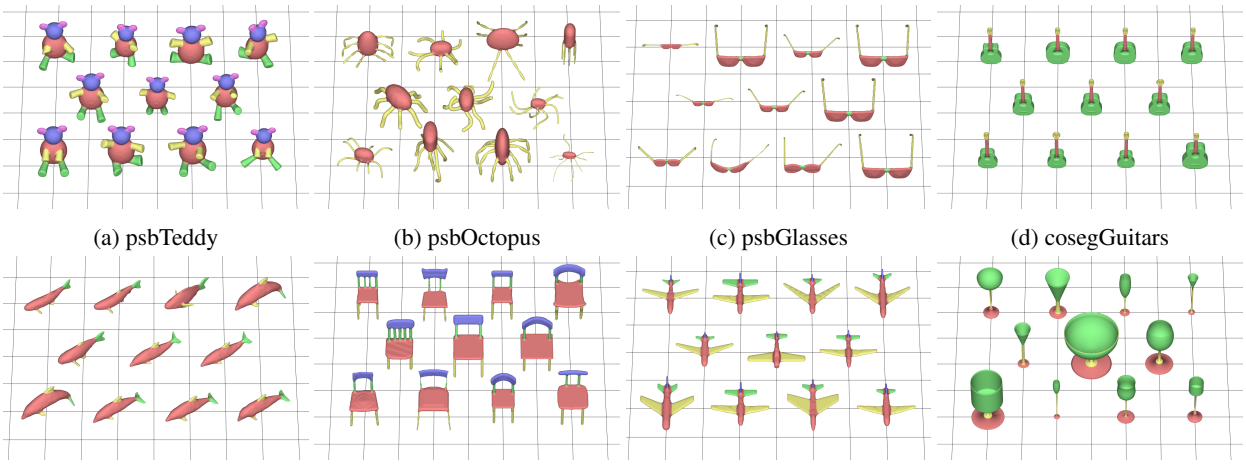
3.5. Graph-Cut Refinement

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 [38] 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:

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

where λ is a non-negative constant used to balance the influence of the two terms and $\xi_D(u, l_u) = -\log(\boldsymbol{p}_u(l_u))$ is a data term that penalises low probability of assigning a label l_u. The second term ξ_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



(a) psbTeddy (b) psbOctopus (c) psbGlasses (d) cosegGuitars

(e) psbFish (f) psbChair (g) psbAirplane (h) cosegGoblets

Figure 6: Visualisation of some results of our 1D CNN technique on the PSB and Coseg datasets, with an accuracy of over 95%.

between two features is high, and is given by:

$$\xi_S(u, v, l_u, l_v) = \begin{cases} 0, & \text{if } l_u = l_v \\ -\log(\theta_{uv}/\pi)\varphi_{uv}, & \text{otherwise} \end{cases}$$

where the cost is based on the dihedral angle (θ_{uv}) and edge length (φ_{uv}) between faces u and v. This formulation has been applied commonly in [22, 15, 28].

Here, we propose to modify ξ_S:

$$\xi_S(u, v, l_u, l_v, f_u, f_v) = -\log(\theta_{uv}/\pi) - \omega\|f_u - f_v\|_2$$

by replacing the distance term φ_{uv} with a geometric feature term $\omega\|f_u - f_v\|_2$. It promotes similar classification label if the Euclidean distance between features f_u, f_v of face u and v is small. A constant (ω), 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 (Section 4).

4. Experiments and Results

In this section, we outline the experiments and accuracy measure used to evaluate each deep learning technique on mesh segmentation. Then we discuss the results and put forward our observations.

All experiments were conducted on the PSB [14] and Coseg [15] datasets, which are widely used datasets for evaluating mesh segmentation techniques [18, 22]. The PSB dataset contains 19 sets with 20 meshes per set. Similar to [30], we omit three sets (Bust, Bearing and Mech) from our results in Table 1, because these sets are either inconsistently labelled or contain meshes with too much variance within the set (Further discussion is provided at the end of this section). The Coseg dataset has 8 sets with between 12 and 44 meshes per set. The dataset also includes 3 large sets with 200 to 400 meshes per set. We ran 3 different types of experiments:

- Leave-one-out cross validation - a single mesh is removed from the training set and used exclusively for testing. This is repeated for all meshes in the set.
- 5-fold cross validation - the set is split into 5 equal (or close to equal) subsets. In each run, a single subset is left out of the training set and used for testing. This is repeated for all 5 folds. The training & testing splits used will be publicly available.
- Fixed training/testing split - we run experiment using the training/testing splits defined in [21] for each set.

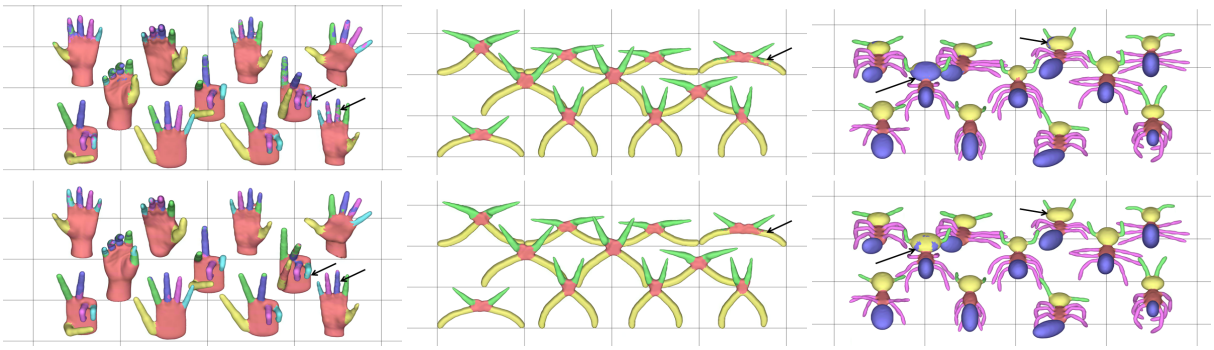
The PSB dataset was used in all experiments. The Coseg dataset was only used in the 5-fold and fixed training/testing split experiments [21] as running leave-one-out cross validation is a lengthy process.

For all experiments we use the accuracy measure:

$$Accuracy(l, gt) = \frac{\sum_{t \in T} a_t \delta(l_t = gt_i)}{\sum_{t \in T} a_i}$$

where a_t, l_t and gt_t are respectively the area, the predicted label and the ground truth label of triangle t. $\delta(l_i = gt_i)$ is assigned to 1, if the predicted label is the same as the ground truth; otherwise 0. This is similar to [18, 22, 15]

Finally, as pointed out by Kalogerakis et al [21], there is no publicly available implementation for Guo et al's



(a) Hand (b) Plier (c) Ant

Figure 7: Visual comparison of our 1D CNN (bottom row) and TOG15 [22] (top row). Our method performs better on certain meshes (see arrows on the figures)

architecture [22]. Similarly, we use our own faithful reimplementation of Guo et al's architecture, closely following the details in the paper. We tried Matlab (MatConvNet) and Python (TensorFlow) implementations, and both show similar results. We reported Python's results as they are marginally better. Both reimplementations are internally validated by three independent researchers from two research teams at Swansea University. Both source codes are available for external validation. Though the reported accuracy of [22] is lower than that reported in the original paper, the reproduced accuracy is still higher than the results reported in [21]. We believe that our reimplementation is faithful.

Leave-one-out Cross Validation. Experimental results for Leave-one-out cross validation 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 and column **TOG15** shows the results using [22]. Some visual results of our 1D CNN technique are shown in Figure 6.

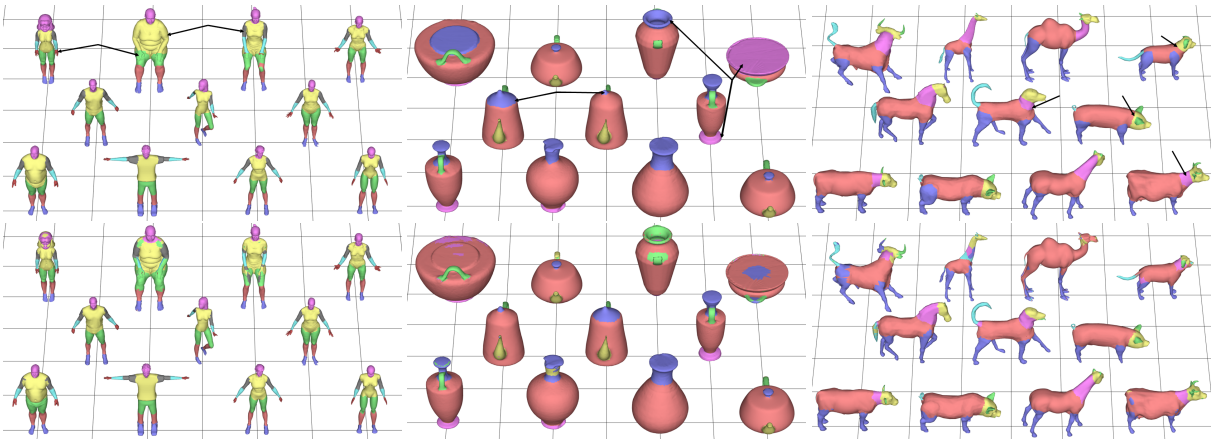
A direct comparison of our 1D CNN and [22] shows that our method achieves higher accuracies on all of the sets. The majority of the sets see a large improvement in accuracy, especially some of the harder sets (Armadillo, Hand, Vase). Five sets (Fish, Glasses, Octopus, Table, Teddy), which have very high accuracies (> 96%) in [22], also show slight improvement. Figure 6 and 7 show some visual comparisons of the results.

We further investigate the sets with poorer results (< 90%) and observe several problems.

- First, the Human set (Figure 8, column (a)) is badly and non-consistently labelled in general, and there is insufficient support to train a proper model (see arrows). This is challenging for any machine learning

	PCA & NN	AE & RF	AE & NN	ToG15 [22]	1D CNN
Airplane	92.97	92.62	92.53	94.56	**96.52**
Ant	95.15	95.17	95.15	97.55	**98.75**
Armadillo	88.21	88.43	87.79	90.90	**93.74**
Bird	85.14	88.93	88.20	86.20	**91.67**
Chair	95.55	95.69	95.61	97.07	**98.41**
Cup	95.09	97.95	97.82	98.95	**99.73**
Fish	94.41	96.21	95.31	96.16	**96.44**
Fourleg	83.61	83.99	82.32	81.91	**86.74**
Glasses	94.22	96.57	96.42	96.95	**97.09**
Hand	78.33	73.76	70.49	82.47	**89.81**
Human	87.03	86.69	81.45	88.90	**89.81**
Octopus	96.93	96.99	96.52	98.50	**98.63**
Plier	93.75	92.59	91.53	94.54	**95.61**
Table	99.22	99.18	99.17	99.29	**99.55**
Teddy	98.07	98.24	98.20	98.18	**98.49**
Vase	79.73	82.07	80.24	82.81	**85.75**
Average	91.09	91.57	90.61	92.79	**94.80**

Table 1: Experimental results for leave-one-out cross validation on the PSB dataset [14]. **Bold**: highest accuracy.



(a) Human (b) Vase (c) Fourleg

Figure 8: Visualisation of sets where our method achieved sub 90% accuracy. Top row shows ground truth, bottom row shows our 1D CNN results. (a) shows where poor ground truth (arrows) resulted in loss in accuracy. (b) shows where inconsistent ground truths and large variations (arrows) resulted in poor results. (c) shows where outliers in the set (back row left 3) and inconsistencies in the ground truths (arrows) caused a loss in accuracy

technique, making all techniques fail to achieve high accuracies (> 90%).

- Second, the Vase set (Figure 8, column (b)) contains meshes that are significantly different from the rest. As indicated by arrows, there is a mesh (back row, second from right) that 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). Also there are two meshes that are almost identical (middle row), but the label is very different (e.g. blue segment).
- Third, the Fourleg set (Figure 8, column (c)) contains three meshes that are very different from all other meshes in the set (back row, left most 3). Label inconsistencies are also present where the majority of meshes contain a neck segment (purple), but some meshes do not (see arrows).
- Finally, each of the fingers in the Hand set (Figure 7) are considered separate segments in the ground truth. It is very hard to achieve good results using features alone for such set. We believe the result can be improved with a correspondence matching technique.

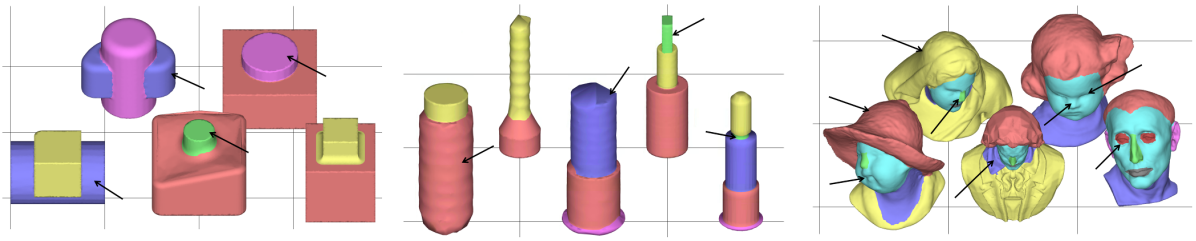
For completeness, the results for the sets we omitted from Table 1 are as follows. Accuracies of 88.67%, 70.06% and 88.53% [22], and 89.69%, 61.97% and 88.14% (our 1D CNN) were achieved for the Bearing, Bust and Mech sets respectively. Following [30], these sets were omitted due to ground truth inconsistencies (Figure 9) and lack of sufficient training data. These are reflected in the lower accuracies of both methods.

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 that have consistent and well-defined labels (Fish and Teddy), AE & RF performs better than the 2D CNN technique. This is interesting as these NN models consists of 2-3 layers, and require much shorter training time than CNN techniques.

Finally we 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. It suggests that there is complementary improvement overall.

Our conclusion is that, compared to [22], if there is a sufficiently large number of good meshes with consistent labels across the set, the new features and 1D CNN architecture can improve performance. Our technique also does not require parameter tuning for features reshaping or sampling to 2D images.

5 Fold Cross Validation. The experimental results in Table 2 show a comparison of running our 1D CNN architecture with different numbers of branches on the PSB dataset. As shown in the table (Columns **1B**, **2B**, **3B**, **4B**), performance steadily increases as the number



(a) Mech (b) Bearing (c) Bust

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.

	1D CNN						ToG15
	1B	**2B**	**3B**	**4B**	**1B (600)**	**3B (600)**	**[22]**
Airplane	94.93	95.57	95.65	95.60	94.24	94.31	93.43
Ant	98.24	98.75	98.82	98.75	97.15	97.23	96.91
Armadillo	93.08	93.29	93.47	93.56	91.02	92.17	87.05
Bird	90.86	91.14	91.34	91.82	90.69	90.80	90.00
Chair	97.72	98.03	98.14	98.39	96.57	97.10	96.43
Cup	99.62	99.65	99.69	99.65	99.45	99.65	99.13
Fish	96.47	96.69	96.75	96.82	96.39	96.68	95.99
Fourleg	87.10	87.78	88.23	88.43	86.38	87.50	84.92
Glasses	96.68	96.72	96.89	96.94	96.25	96.61	96.31
Hand	88.86	89.33	89.66	89.64	87.40	88.45	80.31
Human	87.50	88.81	89.02	88.85	87.34	88.49	82.51
Octopus	98.54	98.67	98.71	98.75	98.51	98.61	98.39
Plier	95.41	95.36	95.52	95.59	95.29	95.34	95.23
Table	99.61	99.62	99.62	99.62	99.59	99.62	99.08
Teddy	98.39	98.37	98.35	98.40	96.67	97.06	95.90
Vase	84.43	86.35	87.10	86.11	84.06	84.66	81.08
Average	94.22	94.63	**94.81**	**94.81**	93.57	94.02	92.04

Table 2: 5-fold cross validation labeling accuracies for the PSB dataset [14]. Results of our 1D CNN with differing number of branches are shown (**1B**, **2B**, **3B**, **4B**), as well as using the same features as ToG15 [22] (**1B (600)**, **3B (600)**)

of branches increase, up until it plateaus at 3-4 branches. A direct comparison to the result of 2D CNN (**ToG15 [22]**) shows that our method outperforms the 2D architecture for all sets, even using a single branch network. It also supports empirically our choice of using a 3-branch network as it is faster to train whilst reaching similar performance as compared to that of 4-branch.

Table 2 also shows that, when using the same set of features in [22], whether it is one (Column **1B (600)**) or three branches (Column **3B (600)**), our architecture can still outperform, with the latter giving better results. These results show that the use of 1D data and filters is useful, and that the multi-branch architecture (with multi-scale features) and the addition of new features (including our proposed more robust conformal factors) all separately contribute to the improvement.

We further show experimental results using the Coseg dataset [15] in Table 3 and Figure 6. We see a large improvement over the 2D CNN [22] for the smaller datasets (over 4% on average, Table 3 left), and notice a larger improvement when the datasets have hundreds of meshes (over 6% on average, Table 3 right). This shows that our model can generalize well when sufficient training data is provided, even if there are large variations in the meshes in the sets (e.g. VasesLarge, AliensLarge). This supports our earlier conclusion that, given a large set of well labelled meshes, our method can be effectively trained for good performance, and can handle largely varying meshes in the set.

Fixed training/testing splits. A final set of experiments use the training/testing splits defined in the concurrent work [21], and use the PSB and Coseg datasets. Table 4 shows the results of these experiments for 2D CNN [22], Projective CNN [21], and our 1D CNN. (The results for 2D CNN [22] and Projective CNN [21] are copied from [21] for direct comparison). It shows that our 1D CNN technique clearly outperforms the existing 2D CNN architecture [22], and also performs comparably with the concurrent work Projective CNN architecture [21]. Note however that these experiments do not fully evaluate the method for each set because not all meshes are used at least once for testing. We include these results for completeness.

5. Conclusion

In this paper, we have shown a novel way of using CNNs on the geometric feature space 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 alleviate unnecessary inference of unrelated features. It also avoids the problem of parameter tuning for reshaping and re-sampling of features, and achieves better performance. Our novel technique clearly out-performs existing work [22] in terms of accuracy and can support more features and a more complex and deeper network. We have further shown a novel way of computing more consistent and robust conformal factor which is less sensitive to small areas of large curvature.

We additionally performed a comprehensive and comparative study of several deep learning techniques for mesh segmentation. We showed that simpler network architectures (e.g. AEs, NNs and RFs) can still perform reasonably well using the same set of geometric features when compared to more complex CNN models. Their training time is also significantly shorter. This suggests that if only a reasonable (not perfect) segmentation is required for downstream application, AE, NN and RF would be a good choice.

We have also shown some labelling problems in the PSB dataset which is commonly used as a segmentation benchmark. For example, there is an insufficient number of meshes in certain sets to cover a large variation of shape and topology and some of the segmentation boundaries in the ground truth labels are not well-defined or consistent (e.g. Human, Bearing, Bust).

Finally, we release the data and code of all techniques discussed in this paper, helping the research of supervised mesh segmentation in the community.

SmallSet	1D CNN	ToG15[22]
Candelabra	93.58	91.55
Chairs	97.75	93.48
Fourleg	94.12	90.75
Goblets	97.80	92.79
Guitars	98.03	97.04
Irons	89.89	80.90
Lamps	86.74	81.52
Vases	92.47	89.42
Average	**93.80**	89.68

LargeSet	1D CNN	ToG15[22]
Vases	95.88	87.57
Chairs	97.71	92.68
Aliens	97.84	91.93
Average	**97.14**	90.73

Table 3: 5-fold cross validation labelling accuracies for the Coseg dataset [15].

	Testing Meshes	1D CNN	ToG15 [22]	ShapePFCN [21]
psbAirplane	8	95.92	91.60	93.00
psbAnt	8	98.72	97.60	98.60
psbArmadillo	8	93.31	85.00	92.80
psbBird	8	91.04	83.10	92.30
psbChair	8	97.67	96.70	98.50
psbCup	8	94.45	92.10	93.80
psbFish	8	96.48	94.50	96.00
psbFourleg	8	87.71	82.40	85.00
psbGlasses	8	96.31	95.30	96.60
psbHand	8	91.70	73.80	84.80
psbHuman	8	90.58	85.60	94.50
psbOctopus	8	98.48	97.40	98.30
psbPlier	8	95.81	95.20	95.50
psbTable	8	99.57	98.50	99.50
psbTeddy	8	88.27	97.30	97.70
psbVase	8	81.94	77.80	86.80
psbAverage	-	93.62	90.24	93.98
cosegCandelabra	16	94.39	85.90	95.40
cosegChairs	8	96.02	93.80	96.10
cosegFourleg	8	93.64	88.20	90.40
cosegGoblets	6	99.46	86.10	97.20
cosegGuitars	35	98.43	97.70	98.00
cosegIrons	6	84.75	79.70	88.00
cosegLamps	8	84.04	78.00	93.00
cosegVases	16	87.55	84.40	84.80
cosegAverage	-	92.29	86.73	92.86
Total Average	-	93.18	89.07	93.61

Table 4: Fixed training/testing split results for PSB [14] and Coseg [15] datasets. Training/testing splits are the same as [21], all sets use 12 training meshes (except cosegGoblets which uses 6)

Acknowledgements

David George is fully-funded by a PhD Studentship (DTG) from the UK Engineering and Physical Sciences Research Council.

References

[1] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, H. Zhang, Contextual part analogies in 3d objects, Int. J. Comput. Vis. 89 (2-3) (2010) 309–326.

[2] Y. Kleiman, O. van Kaick, O. Sorkine-Hornung, D. Cohen-Or, Shed: Shape edit distance for fine-grained shape similarity, ACM Trans. on Graphics 34 (6) (2015) 235:1–235:11.

[3] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, H.-Y. Shum, Mesh editing with poisson-based gradient field manipulation, ACM Trans. on Graphics 23 (3) (2004) 644–651.

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

[5] Y. Yang, W. Xu, X. Guo, K. Zhou, B. Guo, Boundary-aware multidomain subspace deformation, IEEE Trans. Vis. & Comput. Graphics 19 (10) (2013) 1633–1645.

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

[7] L. Shapira, A. Shamir, D. Cohen-Or, Consistent mesh partitioning and skeletonisation using the shape diameter function, Vis. Comput. 24 (4) (2008) 249–259.

[8] O. V. Kaick, N. Fish, Y. Kleiman, S. Asafi, D. Cohen-OR, Shape segmentation by approximate convexity analysis, ACM Trans. on Graphics 34 (1) (2014) 4:1–4:11.

[9] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, et al., Discrete differential-geometry operators for triangulated 2-manifolds, Vis. & Math. 3 (2) (2002) 52–58.

[10] S. Shlafman, A. Tal, S. Katz, Metamorphosis of polyhedral surfaces using decomposition., Comput. Graphics Forum 21 (3) (2002) 219–228.

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

[12] A. Golovinskiy, T. Funkhouser, Randomized cuts for 3D mesh analysis, ACM Trans. on Graphics 27 (5) (2008) 145:1–145:12.

[13] A. Shamir, A survey on mesh segmentation techniques, Comput. Graphics Forum 27 (6) (2008) 1539–1556.

[14] X. Chen, A. Golovinskiy, T. Funkhouser, A benchmark for 3D mesh segmentation, ACM Trans. on Graphics 28 (3) (2009) 73:1–73:12.

[15] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, D. Cohen-Or, Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering, in: Proc. ACM SIGGRAPH ASIA, Vol. 30, 2011.

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

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

[18] E. Kalogerakis, A. Hertzmann, K. Singh, Learning 3D mesh segmentation and labeling, ACM Trans. on Graphics 29 (3) (2010) 102:1–102:12.

[19] Z. Xie, K. Xu, L. Liu, Y. Xiong, 3d shape segmentation and labeling via extreme learning machine, Comput. Graphics Forum 33 (5) (2014) 85–95.

[20] Z. Xie, K. Xu, W. Shan, L. Liu, Y. Xiong, H. Huang, Projective feature learning for 3d shapes with multi-view depth images, Comput. Graphics Forum 34 (6) (2015) to appear.

[21] E. Kalogerakis, M. Averkiou, S. Maji, S. Chaudhuri, 3D shape segmentation with projective convolutional networks, in: Proc. IEEE Conf. CVPR, 2017.

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

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

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

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

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

[27] R. Hu, L. Fan, L. Liu, Co-segmentation of 3d shapes via subspace clustering, Comput. Graphics Forum 31 (5) (2012) 1703–1713.

[28] M. Meng, J. Xia, J. Luo, Y. He, Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization, Comput. Aided Des. 45 (2) (2013) 312–320.

[29] Z. Wu, Y. Wang, R. Shou, B. Chen, X. Liu, Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering, Comput. & Graphics 37 (6) (2013) 628–637.

[30] Z. Shu, C. Qi, S. Xin, C. Hu, L. Wang, Y. Zhang, L. Liu, Unsupervised 3d shape segmentation and co-segmentation via deep learning, Comput. Aided Geom. Des. 43 (2016) 39–52.

[31] P. Theologou, I. Pratikakis, T. Theoharis, A comprehensive overview of methodologies and performance evaluation frameworks in 3d mesh segmentation, Comput. Vis. and Image Understanding 135 (2015) 49–82.

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

[33] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, arXiv preprint.

[34] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, arXiv preprint.

[35] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, X. Tong, O-cnn: Octree-based convolutional neural networks for 3d shape analysis, ACM Trans. on Graphics 36 (4).

[36] L. Yi, H. Su, X. Guo, L. Guibas, Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation, arXiv preprint.

[37] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, Y. Lipman, Convolutional neural networks on surfaces via seamless toric covers, in: Proc. ACM SIGGRAPH, 2017.

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

[39] R. Liu, H. Zhang, A. Shamir, D. Cohen-Or, A part-aware surface metric for shape analysis, Comput. Graphics Forum 28 (2) (2009) 397–406.

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

[41] A. E. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3d scenes, IEEE Trans. Pat. Anal. & Mach. Intell. 21 (5) (1999) 433–449.

[42] J. Sun, M. Ovsjanikov, L. Guibas, A concise and provably informative multi-scale signature based on heat diffusion, Comput. Graphics Forum 28 (5) (2009) 1383–1392.
[43] M. M. Bronstein, I. Kokkinos, Scale-invariant heat kernel signatures for non-rigid shape recognition, in: Proc. IEEE Conf. CVPR, 2010, pp. 1704–1711.
[44] G. Taubin, A signal processing approach to fair surface design, in: Proc. ACM SIGGRAPH, ACM, 1995, pp. 351–358.
[45] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F. E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2016) 11–26.
[46] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proc. Int. Conf. Mach. Learn., 2015, pp. 448–456.
[47] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, ICML Deep Learning Workshop.
[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proc. IEEE Conf. CVPR, 2015, pp. 1–9.
[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958.