Computer-Aided Design 144 (2022) 103179

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

A Deep Learning Driven Active Framework for Segmentation of Large 3D Shape Collections

David George [a], Xianghua Xie [a], Yukun Lai [b], Gary K.L. Tam [a,*]

[a] Department of Computer Science, Swansea University, United Kingdom
[b] School of Computer Science and Informatics, Cardiff University, United Kingdom

ARTICLE INFO

Article history:
Received 1 June 2021
Received in revised form 24 September 2021
Accepted 30 November 2021

Keywords:
Shape segmentation
Active learning
Shape collections
User interaction

ABSTRACT

High-level shape understanding and technique evaluation on large repositories of 3D shapes often benefit from additional information known about the shapes. One example of such information is the semantic segmentation of a shape into functional or meaningful parts. Generating accurate segmentations with meaningful segment boundaries is, however, a costly process, typically requiring large amounts of user time to achieve high-quality results. In this paper we propose an active learning framework for large dataset segmentation, which iteratively provides the user with new predictions by training new models based on already segmented shapes. Our proposed pipeline consists of three components. First, we propose a fast and accurate feature-based deep learning model to provide dataset-wide segmentation predictions. Second, we develop an information theory measure to estimate the prediction quality and for ordering subsequent fast and meaningful shape selection. Our experiments show that such suggestive ordering helps to reduce users' time and effort, produce high-quality predictions, and construct a model that generalizes well. Lastly, we provide interactive segmentation refinement tools, helping the user quickly correct any prediction errors. We show that our framework is more accurate and in general more efficient than the state-of-the-art for large dataset segmentation, while also providing consistent segment boundaries.

© 2021 Elsevier Ltd. All rights reserved.

1. Introduction

Segmented datasets have already been shown incredibly useful for many applications, including shape matching [1], retrieval [2] and modeling [3]. Semantic labels are also useful for shape understanding and abstraction [4], and shape parsing and partial shape recovery [5]. Shape segmentation techniques often benefit the most from such fully labeled datasets. Supervised techniques require ground truth labels to train segmentation classifiers [6], and both supervised and unsupervised techniques need ground truth labels to evaluate their methods [7]. While existing works have shown good efforts and results [8–10], clear ground truth inconsistencies still exist [11]. This means both existing and new techniques could perform better with higher quality ground truth segmentations.

Generating high-quality segmentations for shape datasets is a time-consuming and interaction-heavy task. Smaller datasets, with only small numbers of inconsistencies or errors may be manageable through manual effort [7,12]. Massive datasets would take a great amount of user effort however [13]. Further, these massive datasets typically consist of non-manifold (multiple components, holes, zero thickness, etc.) and low-resolution shapes. These shapes are very difficult to process in segmentation pipelines. Recent works employ point cloud projection [9,14], or further KD-connected point cloud projection [15]. While these are viable techniques, there may be information loss when using point clouds, e.g., connectivity and topology of the shape. Without these, certain reliable features are much harder to compute or are inaccurate when computed (e.g., Shape Diameter Function (SDF) [16], Geodesic Distance). Although connectivity can be re-established (e.g., through K Nearest Neighbors, assuming the resolution of the point cloud is high enough), thin regions of the shape could be wrongly connected, leading to undesirable connections. More recently, there are increasing interests to use mesh-based representations to develop robust CNN techniques [17,18]. For this reason, in our proposed pipeline, we largely focus on input meshes. We further show that by re-meshing these non-manifold 3D models into manifold meshes, our pipeline can handle very large datasets very well.

Previous works that generate ground truth segmentations for large datasets typically focus on active learning approaches, where a user has some control over the system and influences the decisions in some way. [19] first used an unsupervised co-segmentation algorithm, where the user interactively selects pairs

* Corresponding author.
E-mail address: k.l.tam@swansea.ac.uk (G.K.L. Tam).

https://doi.org/10.1016/j.cad.2021.103179
0010-4485/© 2021 Elsevier Ltd. All rights reserved.

of parts between shapes to connect or disconnect. [14] used a supervised algorithm to label a single part at a time. Users are asked to paint two 2D views of a 3D shape. A learning model is trained based on the painted regions and similar shapes (according to global shape descriptors) are evaluated on that model. However, these techniques can only provide a coarse segmentation and output segmentations may have errors. Further, [14] requires one part to be labeled at a time, so datasets with high numbers of parts will take longer and more iterations to label. Here, we developed an active framework which allows full shape segmentation of a shape dataset, to ensure good segmentation quality and it scales well with the number of parts in the dataset.

One of the challenges when developing an active framework for segmentation is minimizing user interactions while maximizing segmentation quality. To balance the quality and speed, we utilize a deep learning model for segmentation predictions. In general, deep learning models can take a long time to train, and typically require a large amount of training data. To resolve these, we propose to use a small Convolutional Neural Network (CNN), using two 2D histogram features as input. The features have been shown useful in previous work [6,8] and fit the CNN paradigm as 2D histograms are like images. Our architecture allows for quick model training and we also adopt an ensemble based learning scheme [20] to help generalize with reduced available training data. In our experiments we compare to other feature-based CNN techniques. We show that our model can perform better than existing fast techniques, with results comparable to the state-of-the-art.

Another difficulty of an active learning framework is the exploration and analysis of model predicted results. It often takes a long time for users to choose the next 3D model to segment, and there are no ground truth data to compare the predictions for ranking. We thus use *entropy*, a measure of uncertainty, to define a ranking measure without needing ground truth segmentations. This ranking measure provides a meaningful ordering of the predicted segment labels in an interactive tabular view. This allows users to see which shapes the deep learning model segmented well or struggled with. Our experiments show that by selecting poorly segmented 3D models with respect to the ranking measure, it reduces both time and interactions required to segment the whole dataset.

Lastly, another problem we observed in existing active frameworks (e.g., [14]) is that they do not allow quick boundary refinement. When there are slight errors in the output segmentation, users will likely discard the results, leading to extra manual effort and longer interaction time. With this observation, we propose an interactive segmentation refinement algorithm that takes the current segmentation and information about the shape (e.g., angle and thickness) to refine the segmentation boundaries. This algorithm can quickly provide high-quality segmentations while greatly reducing interactions and time required to refine a shape.

Our proposed framework has been demonstrated to work well on public datasets (including PSB, COSEG), and also on re-meshed datasets from ShapeNet, which contains thousands of shapes.

Contributions. To summarize, the main contribution of this work is to develop the first deep learning driven active framework for segmentation of large 3D shape collections. The focus is to maintain accurate and meaningful segment boundaries, while keeping human effort and time to a minimum. Our active learning framework consists of several key components:

- First, we show and evaluate a novel deep learning pipeline for shape segmentation which is relatively fast and accurate, and is suitable for active learning purpose.
- Second, we use an information-theoretical metric for ordering the prediction of shape segmentation when ground truth data is not available. The metric is designed for our segmentation tasks. Users can still flexibly choose next shape to annotate through our interface. Our extensive experiments show that the ordering can help reduce total segmentation efforts and time.
- Third, we develop a useful technique for interactive segmentation refinement, which takes into account the segmentation boundaries and thickness of shapes. Our experiments show that it can help users to quickly improve segmentation boundaries, reducing effort and time.

We will also release the source codes of our tools for the community, and provide new and more accurate ground truth segmentation for some existing datasets.[1]

In the following, Section 2 discusses the existing work for segmentation, feature extraction and entropy in geometry processing. In Section 3, we briefly overview our active learning framework. Section 4 discusses the details of the three novel subsystems. We further discuss our framework interface and flow in Section 5 before outlining our experiments and showing their results in Section 6. Finally, in Section 7 we conclude and discuss possible future work.

2. Related work

This work relates to several research areas. We summarize the literature with respect to shape features, shape segmentation, active learning in image analysis, active learning in shape analysis, and use of entropy in graphics processing.

Shape Features and Their Uses. Much of the existing work in shape segmentation is driven by features. These can be defined per face, per vertex, per patch (a cluster of faces), or even per shape. These features are designed for different purposes, and many have been successfully applied in mesh segmentation. Per-face features include, SDF [16] which estimates the thickness of a shape at a given face, Conformal Factor (CF) [21] which computes a position invariant representation of the curvature of non-rigid shapes and Spin Images (SI) which capture the surface information around a face using a 2D histogram. Recent work has also adapted image based features to the 3D domain. One notable example is Shape Context (SC) [22] which is a 3D shape descriptor to encode both curvature and geodesic distance distributions in a 2D histogram [6]. However, there are limitations to how useful a feature can be on certain shapes. Examples are that CF is susceptible on shapes with sharp curvature [11], SDF can fail if the shape has holes and geodesic distance will fail if the shape has multiple components. Therefore feature selection for a new technique is very important, as it can greatly impact the accuracy and speed. For these reasons, we opted to use two features for this work, SC and SI. Recent work has shown both features can be very useful in shape segmentation [6,11,23]. Further they are both 2D histograms, so can be generated at any scale (number of bins) and CNNs should work well to extract useful information.

Unsupervised and Supervised Shape Segmentation. The goal of a shape segmentation algorithm is to partition a single shape into meaningful parts [16,24]. These algorithms typically used a feature which drives the partitioning (see Features section), though other work also used different strategies like fitting of primitive shapes [25]. Recently, unsupervised techniques looked into co-analysis of a set of shapes, using information consistent across the set to improve the final segmentation [7,26–29].

[1] https://cs.swan.ac.uk/~csgarykl/ActiveLabeller/ActiveMeshLabeller.html. Code & Dataset: https://data.mendeley.com/datasets/8zhn66k5xj/1

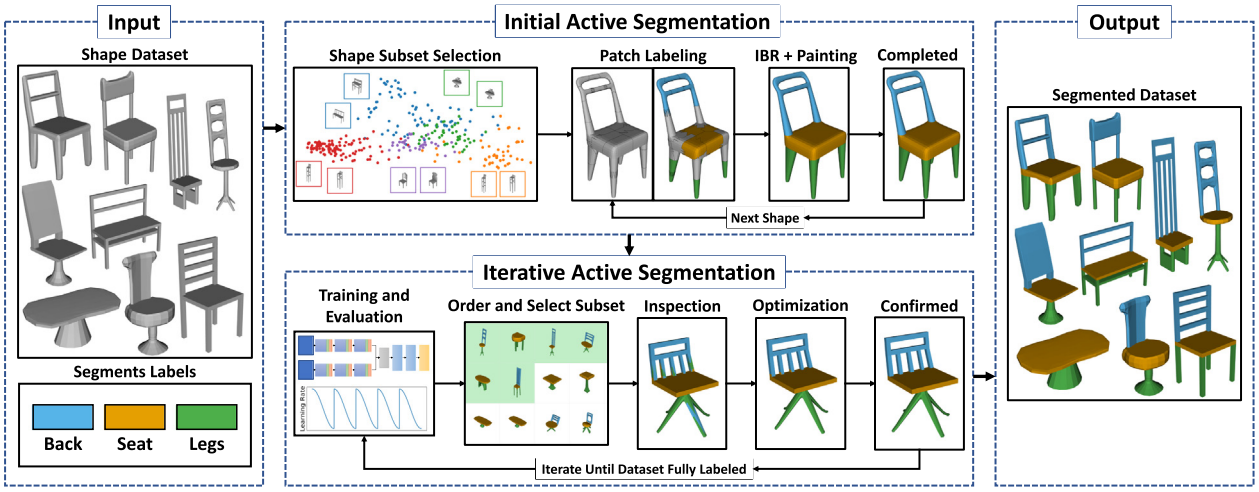


Fig. 1. Proposed Pipeline. After defining the input data (shapes, features and possible segment labels, Sections 4.1, 4.2), users pick some models (Section 4.3 Shape Subset Selection) and use the proposed interface and tools (Section 4.4, Patch Labeling, Painting, and Section 4.5 Interactive Boundary Refinement (IBR)) to annotate ground truth labels. With these ground truths, a fast deep learning model is trained. Graph-cut is applied to refine the predicted labels (Section 4.6 Training and Evaluation). The results are then ordered in an interface (Section 4.7, Order and Select Subset) for users to confirm the ground truth or further select a subset for user-driven inspection and IBR refinement (same interface in Sections 4.4, 4.5). The iterative active segmentation repeats until dataset is fully labeled.

However, these methods struggle with largely varying datasets, especially those with a low number of shapes per set [11]. Further, the segmentation of parts not only relates to the shape geometry, but also the meaning, functionality and designs. All these challenges have led to the recent interests in supervised segmentation techniques.

Supervised segmentation techniques rely on prior knowledge in order to train a model. Typically these methods use large pools of shape features as input and classify them according to segment labels [6]. Subsequent techniques further improve in different ways, such as ranking features to find segment boundaries [30], and training an extreme learning machine [23,31] to classify the labels. However, similar to unsupervised work, these techniques can struggle when datasets are very diverse. To combat this, work using CNNs was proposed [8]. This work arranges a pool of features as an image, and uses an image-based convolution network to predict face labels. However, the simple arrangement leads to unnecessary interference of relationships between features with no correlation, and [11] reduces such interference using 1D convolutions, leading to better results. Recently, several techniques have shown new and interesting shape segmentation methods such as point cloud segmentation [9,32], kd-tree point cloud segmentation [15], projecting image segmentations to shapes [33], hierarchical segmentations [34] and graph CNNs [10].

With the recent surge of new segmentation papers, each focusing on larger datasets, there is a need for high-quality ground truth labels. However, currently available ground truths for widely used segmentation datasets have been shown to contain inconsistent and poor labels for certain shapes within the dataset [11]. This can impact the training performance by introducing inconsistent labels for similar samples. It can also impact evaluation, as inconsistencies incorrectly degrade the performance of a model. Due to this, we emphasize providing accurate, high-quality segmentations in this work.

There is a concurrent work [35] that shares similar spirit as ours which produces high quality part annotations. They employed professionals to carry out the annotation. The fine-grained part dataset further inspires more recent interests in hierarchical shape segmentation [36], and grouping and labeling of semantic parts [37]. Compared to [35], our work proposes an active learning framework that allows fast annotation of large datasets with the help of a machine learning model. The framework can be used for other work to complete annotations of large datasets. To our knowledge, it is the first deep learning driven active learning framework for segmentation of large 3D shape collections that aims at ground truth quality.

Active Image Analysis. Active learning image analysis systems have been widely explored to leverage the human user input to explore large datasets. They focus on using user input to aid the classifiers by annotation (painting, strokes) or drawing bounding boxes. This has the advantage that, the user can see what data the classifier is struggling with and incrementally provides new training data to alleviate this problem, making the classifier more generalized and accurate [38–41]. We utilize this functionality in 3D segmentation by allowing the user to incrementally tune the output labels of our model to make it generalize better, while also incorporating a sorting method to easily rank the outputs of the model, and aid the user in selecting shapes to tune.

Active Shape Analysis. Unlike the image domain, there are few methods using user interactions to aid 3D shape segmentation. One of the earliest techniques proposed in [19] asks the user to select pairs of segments between shapes to denote if they have the same or different segments. This technique is driven by an unsupervised method, so segmentation between shapes can be mismatched, and thus the user input to select the right shapes is crucial. Similarly, [42] asks the user to paint regions of the shapes for segment matching. Both methods tend to focus on smaller sets of shapes and use unsupervised methods to drive the segmentation. Recently, [14] proposed a framework for annotating massive 3D shape datasets. They offer a crowdsourcing application for annotators to label a specified region which is used to train a conditional random field model. Once model predictions are obtained, the user would then be asked to verify the results by selecting all shapes that fail to have adequate annotations. While this technique shows good performance, fine details such as accurate segment boundaries can be difficult to achieve. Further, the user is only asked to verify results, and cannot fine tune 'almost acceptable' segmentations. These 'almost acceptable' segmentations then end up going through another round of model predictions or user labeling. Finally, this approach is a labeling pipeline, where a full pass only provides a single segment for a dataset. Therefore, datasets with many distinct segments require many full passes to achieve a complete segmentation. Our proposed method alleviates all of these problems. By

making use of a fast and robust deep learning model and effective refinement tools, we provide high-quality full segmentations in efficient times.

Entropy uses in Geometry Processing. Entropy is a measure of uncertainty. It can be used to predict the probability of an event given some information. Entropy was first used in 3D geometry processing by [43], where it was used to estimate how much information was contained in a 3D surface. More recently, entropy has also been used for shape simplification [44], shape compression [45] and to estimate the saliency of a 3D shape [46]. In this work, we use entropy to model uncertainty to rank the segmentation prediction without needing ground truth. To our knowledge such ordering and tabular interface for active learning 3D shape segmentation was not explored before.

3. Framework overview

Our active learning framework aims to produce a full segmentation for every shape in a given dataset, whilst minimizing the users' manual efforts. The input to our framework is a collection of manifold 3D shapes of any size S, from a specific category (e.g., aircraft), and a set of pre-defined segment identifiers L (e.g., wings, engines, body, stabilizer). With our framework, users guide the selection of shapes for labeling, interact with the prediction of a deep learning model, and verify the segmentation quality of each shape. The framework will then produce an output of per-face labels for each shape, indicating which face belongs to which segment. The pipeline for our framework is shown in Fig. 1.

The pipeline consists of several components. Each component is inspired by the observations of existing problems, leading to our contribution of a fast and reliable active framework.

The framework is driven by a deep learning model, which predicts the labels for faces given feature descriptors (see Section 4.2 for feature details). As it is a supervised system, initial training data is required. This is obtained by the user manually segmenting several shapes. To aid the user in this task, the system will first suggest a small subset of shapes for the user to label, this is done by clustering global shape descriptors (See Section 4.3). This subset aims to well represent the dataset, to aid model generalization early on.

When manually labeling the dataset, the system offers the user many effective tools to speed up the process while still maintaining the high segmentation quality. These tools include robust over-segmentation and effective painting utilities (Section 4.4) and interactive boundary refinement (Section 4.5).

Once the user confirms the labels for any shape the deep learning model considers them as ground truth for training. So with the initial subset fully labeled, a model will be trained and used to predict results for the remaining shapes in the set (see Section 4.6). These results are displayed in our interactive table, which can be ordered in many useful ways (see Section 4.7). These different ordering methods allow the user to quickly see which shapes are correct and add them to the completed set (to be used for future model training). Alternatively, the table also quickly shows which shapes the model struggles with, and these can then be manually labeled to make the model more generalized and increase overall quality.

The deep learning model is designed both to be quick and give high-quality results, as such, the above steps can be repeated in quick successions to achieve a strong and generalized model. This can be used to quickly and effectively segment the entire dataset, with the user requiring less and less input per iteration (see Fig. 2).

4. Methodology

This section details all of the functions and tools provided by our active learning framework, in order to minimize user input and time, while still keeping high segmentation quality.

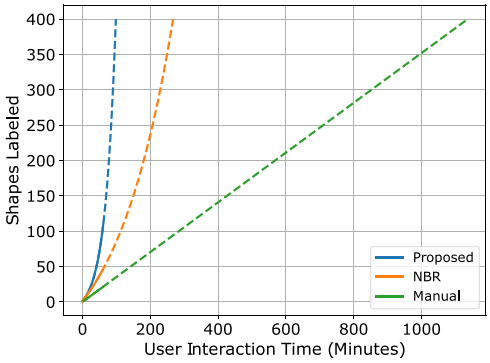


Fig. 2. This graph shows the number of segmented shapes increases as users use the system. The 'proposed' line represents our system with all features enabled, the NBR (No Interactive Boundary Refinement) line represents our system with the boundary refinement feature disabled and the 'manual' represents a system where only painting is enabled. The data making up the solid lines is from our 1-hour user studies, while the dashed lines are interpolated from the user study data. The figure shows that our system considerably speeds up shape segmentation, compared to a manual painting approach. We will provide more evaluations below.

4.1. Input datasets

The input to our framework is a dataset of 3D shapes. As our method makes effective use of both geodesic distances and graph traversal, the input shapes must be manifold. Let S be a dataset consisting of n shapes, $S = \{s_1, s_2, \ldots, s_{n-1}, s_n\}$, where the ith shape $s_i = \{F, V, E\}$ is made up of faces F, vertices V and edges E (the subscript i is omitted when there is no confusion to avoid clutter). In the supplementary materials, we also discuss a re-meshing procedure to handle shapes that are not manifold. It allows us to evaluate on a manifold sub-set of ShapeNet (see Section 6.4).

4.2. Feature extraction

As a pre-processing step we compute several features which help drive the framework. Specifically, we compute 3 face-level features and 1 shape-level feature. **Face-level:** We use Shape Context (SC) [22] and Spin Images (SI) [47] as input for our deep learning architecture, which acts as a dual-branch ConvNet. This independently compresses both features down and then combines them for classification (see Section 4.6). We also utilize the Shape Diameter Function (SDF) [16], which is used to aid our interactive boundary refinement process (see Section 4.5). **Shape-level:** We compute Light-Field Descriptors (LFDs) [48] for each shape in the dataset. Similar to [34], we extract multi-view snapshots of the shapes and then compute the Histogram of Oriented Gradients (HOG) features of those views. These shape-level features are used as an embedding space for shape selection (see Section 4.3). Empirically we find that PCA embedding works nicely and fast for our purpose.

Though these are all pre-defined features, we find them useful for our later deep learning purpose (balance between speed and accuracy, Section 4.6) to support fast active learning. These features are also 2D by design and do not suffer from feature boundary issues [8,11]. More details of feature extraction can be found in supplementary materials.

4.3. Initial shape selection

Our framework is driven by a deep learning architecture. In order to train it we first need some labeled data samples. One solution would be to let the user arbitrarily select some initial

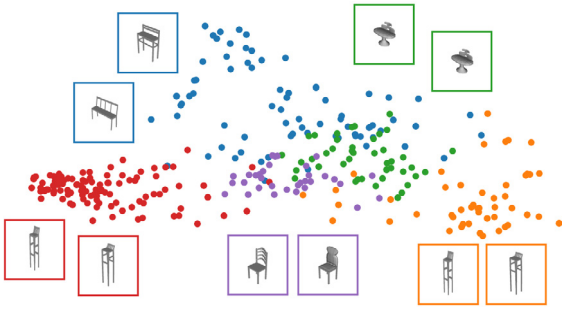


Fig. 3. Shape embedding space from COSEG [7] ChairsLarge dataset using 128-dimension LFD HOG features. Shapes displayed are the two closest shapes to the corresponding cluster centers.

shapes from the dataset and manually label them. While it works to provide labeled training data, there is a chance that this data will not be well distributed throughout the dataset.

We address this by providing an embedded view of the data, which can then be clustered as desired. For a given number of clusters k, we compute k-means on the embedded LFD HOG features for the full dataset. Then, given the cluster labels C^l and cluster centers C^c we compute the n closest shapes to each cluster center. These shapes are displayed to the user so they can select ones they wish to manually segment (Fig. 3).

4.4. Manual segmentation tools

Letting the user segment several shapes in a naive way is one possible way of obtaining an initial training set for the model, such as letting them paint the entire shape from scratch or select segment boundaries with precise clicks. While it works for existing work [14], their work focuses primarily on single part labeling with little regard for segment boundaries. As we focus on shape segmentation with additional care to preserve good segment boundaries, such a way of segmenting shapes manually would be too time-consuming or produce poor results.

As such, our manual segmentation pipeline contains several useful tools to aid the user in quickly and effectively segmenting each shape. Here we outline the manual segmentation pipeline, showing useful tools that help in each step.

Shape Over-segmentation. The first step in the pipeline is to assign segment labels to an over-segmentation of the shape. We provide two options for the over-segmentation, in most cases the shape can be segmented to an almost completed level using random walks segmentation [49]. For the other cases, we also offer a k-means clustering on face centers. It gives uniform patches across the shape and is quick to compute. The outcome of either over-segmentation algorithm is a set of patches across the shape. These patches can quickly be assigned a segment label by the user so that they can move onto the refinement. The user does not have to give all patches a label, as the boundary smoothing algorithm (see Section 4.5) used to transit from this stage to the refinement stage will assign a label to any unlabeled patches.

Segmentation Refinement. At this stage, the shape is fully segmented, however, some modifications may be needed to achieve a good quality segmentation or to make segment boundaries acceptable. In this stage the user is able to 'paint' the shape to change the segment labels assigned to specific faces. There are several useful tools available in this stage:

- **Variable Sized Painting**. When painting a shape, a breadth first search algorithm is used to traverse it and assign the new label to faces it traverses. This algorithm is constrained by an adjustable radius (which is clearly shown to the user during painting). This allows for both large label corrections, or altering very fine details on boundaries.
- **Paint Restrictions**. As a radius based breadth first search is used to paint the shape, it is possible that the bounding sphere that the radius corresponds to will pass over to parts of the shape the user does not wish to paint. To alleviate this, an angle based restriction can be enabled, which will compare the face normal of every traversed face to the normal of the face that was clicked. If the angle is greater than a user changeable threshold then the face will not be painted. As many segment boundaries lie on concave parts of shapes, this can be very useful for quickly refining boundaries.
- **Segment-wide Paint**. If an entire part of the shape is mislabeled (sometimes the result of the deep learning algorithm), it can be quickly re-labeled to another segment by using this feature. All connected faces to the clicked face will be re-assigned the new label.
- **Multiple Shape Views**. Quickly analyzing the quality of a segmentation is essential to minimize the time and effort needed by the user. For this, we allow for multiple views of the shape to be shown at once. This feature is best suited for quickly analyzing the output of the deep learning model, but can also be useful in the early stages of the pipeline.

A final feature, which is the most useful tool in this stage is the interactive boundary refinement, which is covered below in Section 4.5. By making use of all of these tools the user interaction time and effort can be cut down substantially while keeping the segmentation quality high, which is the key focus of this work.

4.5. Interactive boundary refinement

To achieve a high level segmentation with smooth boundaries in an active system, typically, the user would have to spend time in a refinement stage. In this stage, users would typically fine tune the small details of boundaries to achieve the desired result. This task is both tedious and time-consuming. To alleviate this we introduce an interactive boundary refining algorithm.

Our idea is to use multi-label alpha expansion [50] to optimize consistent labels near the boundary areas. Let $\hat{l}_f$ be the currently observed label of a face f. Such label can be obtained from our manual segmentation tools (Section 4.4), machine prediction after graph-cut (Section 4.6), or earlier results of this interactive boundary refinement technique. Our technique is user-driven and can be applied multiple times iteratively.

Let N_f be the set of neighboring faces of f, we can optimize a new set of face labels l_f by solving:

$$\min_{l_f, f \in F} \sum_{f \in F} \xi_D(f, l_f) + \sum_{f \in F, f' \in N_f} \xi_S(f, f'), \tag{1}$$

The data term ξ_D estimates the probability of assigning a label l_f to face f whilst ξ_S is the smoothing term that promotes consistent labels in adjacent faces. In our formulation, we define the data term ξ_D as a weighted geodesic distance term that relates to the closest label boundary. Specifically, we define a set of label boundary edges $E_b^{\hat{l}} \subset E$ where each $e \in E_b^{\hat{l}}$ is shared by a pair of neighboring faces $u, v \in F, u \in N_v$ such that $\hat{l}_u \neq \hat{l}_v$. We define $d_f^{\hat{l}} = Gdist(f, E_b^{\hat{l}})$ as the shortest distance from f to the closest edge in $E_b^{\hat{l}}$, where $Gdist(\cdot, \cdot)$ is the geodesic distance. (We approximate this geodesic distance using the multi-source Dijkstra algorithm and compute the shortest distance between

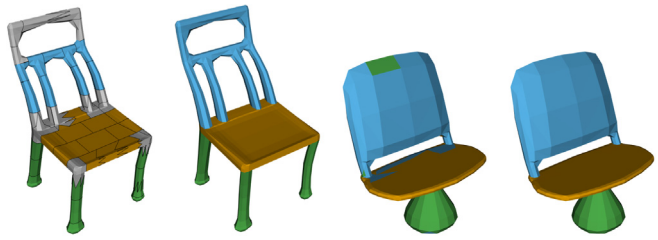


Fig. 4. Resulting segmentation from running our interactive boundary refinement algorithm. The algorithm can take incomplete (gray input patches are considered unlabeled) patch segmentations (a) and return very good refined results (b). The algorithm also performs well when given model predictions (c) as the results shown in (d).

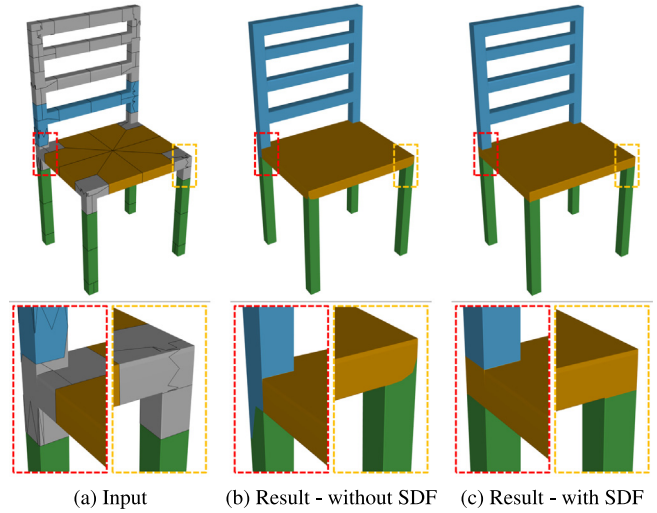


Fig. 5. Comparison of our boundary refinement algorithm with and without SDF in the smoothness term. The segment boundaries (black lines) are poor when SDF is not used ($\omega = 0$) (b), but better when SDF is used ($\omega = 0.2$) (c). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

vertices of edges $E_b^{\hat{l}}$ and vertices of face.) With these, we define $\xi_D(f, l_f) = \exp(-D(f, l_f))$ where:

$$D(f, l_f) = \begin{cases} 1 & \text{if } d_f^{\hat{l}} \geq \sigma \text{ and } l_f = \hat{l}_f \\ 1/2 & \text{if } d_f^{\hat{l}} < \sigma \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The intuition is that if a face is far from a label boundary, we encourage l_f to be the observed label $\hat{l}_f$. If a face f is within σ distance from a label boundary, the probability is set to $\exp(-0.5)$, allowing the labels to be optimized. In all our experiments, we empirically set σ to 0.01 times the bounding diagonal.

The smoothness term ξ_S promotes consistent labels for adjacent faces f and $f' \in N_f$ and is given by:

$$\xi_S(f, f') = \begin{cases} 0 & \text{if } \hat{l}_f = \hat{l}_{f'} \\ \tilde{\xi}_S(f, f') & \text{otherwise.} \end{cases} \tag{3}$$

$$\tilde{\xi}_S(f, f') = (1 - \omega) \cdot \frac{(\pi - \theta_{ff'})}{\pi} + \omega \cdot \frac{\delta_{ff'}}{\max(\delta)} \tag{4}$$

The first term in $\tilde{\xi}_S$ penalizes large curvature where $\theta_{ff'}$ is the dihedral angle between the normals of faces f and f'. The second term penalizes large difference in thickness feature where $\delta_{ff'}$ is the absolute difference between the SDF features [16] of faces f and f'. Both terms are normalized to $[0, 1]$. ω is a weight (between 0 and 1) that trades off the two terms.

In general, penalizing by large curvature provides good results. The optimized labels are continuous and reflect segment boundaries well. This term is useful in both transitioning from patches to painting, and also as a refinement method for the output of the deep learning model (see Fig. 4 where $\omega = 0$).

One drawback of simply using curvature is its inability to detect segment boundaries that do not lie on high curvature regions. One solution to this comes from the observation that segment boundaries also typically lie on regions with a large change in thickness [16]. With this observation in mind, we include the SDF feature in the interactive refinement algorithm.

A comparison of the effect of both terms is shown in Fig. 5. It shows an example shape which would fail using only curvature term, but gives good results with SDF term ($\omega = 0.2$). In our interface, users can freely choose to use either of these terms as both are strong in different ways ($\omega = 0.2$ by default). Users can also apply this interactive refinement algorithm multiple times if desired.

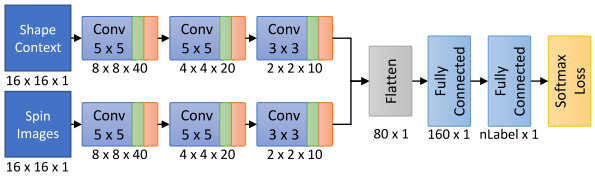


Fig. 6. Architecture of our deep learning model. A Conv block consists of a convolution layer with leaky ReLU [51] activation ($\alpha = 0.2$), followed by a 2×2 max pooling layer with a stride of 2. The numbers underneath each layer represent the output sizes. The architecture separately compresses both input features before combining them to compute predicted class labels.

4.6. Deep learning label predictions

The core of our active learning pipeline is the deep learning model. By providing a small subset of the data, our fast and effective model will predict segmentations for the rest of the dataset, removing the need for manual labeling from scratch.

We have designed our deep learning architecture with *both speed and performance* in mind. The model must be quick to train and evaluate so that the user is not waiting for long periods, but the model must also be accurate to further minimize the user's input and time spent. With this in mind we designed a convolutional neural network to separately compress two features (SC and SI) and non-linearly combine them for label predictions (see Fig. 6 for the architecture).

Previous work has shown that geometric features can be very useful in making a deep learning model both accurate and generalized [8,11], so we opted to use two of the strongest features in our architecture and separately compress them using convolutional and pooling layers. We did this because both features are 2D histograms, so convolving over them in a 2D space is a logical way of compressing the size of the features while maintaining as much of information as possible. Once compressed, the two features are flattened down to a feature vector and concatenated. We then pass this feature vector through a small fully connected network to obtain the final predictions via a softmax layer (Fig. 6).

We chose to train our models using a snapshot ensemble [20] learning scheme. This was because it allows for multiple models to be trained in the same amount of time, increasing the ability for the model to generalize. Empirically, we chose to train our networks using the RMSProp [52] optimizer. In our experiments

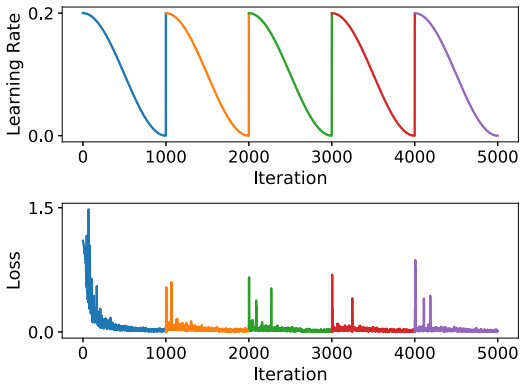


Fig. 7. Learning rate and loss plotted as the model is trained. Different snapshots are shown in different colors. Each time the learning rate resets the model optimizer is forced out of a local minima making the loss spike.

we train each model for $T = 5000$ iterations, and save $M = 5$ snapshots of the model weights. We employ the same learning rate function as proposed by [53]:

$$\alpha(t) = \frac{\alpha_0}{2}\left(\cos\left(\frac{\pi \mod (t-1, \lceil T/M \rceil)}{\lceil T/M \rceil}\right) + 1\right) \tag{5}$$

where α_0 is the initial learning rate (we set $\alpha_0 = 0.01$). This gives a learning rate α for any given $t < T$. The learning rate resets M times so that the model can escape local minima giving a more generalized model [20] (see Fig. 7). As more shapes are labeled and the number of training samples grows, we opted to uniformly sample batches (each with 512 samples) from the entire pool of data. This allows us to fix the number of iterations and still train generalized models. This stops the model from taking an increasing amount of time to train each time new shapes are labeled.

Once an ensemble of trained models has been obtained the remaining shapes in the dataset can be evaluated. The features for all the faces of a shape is passed through each network in the ensemble, and we then extract the label probabilities and average them across all ensembles, giving $\boldsymbol{p}$. The label with the highest probability is considered the segment label for a given face, we call this the predicted segmentation.

Overall, our framework is flexible to the inputs, features and learning architectures. Our design focuses on trading off speed and accuracy so that an active framework is possible where annotators do not need to wait too long. We note here that our active framework is not restricted to our proposed deep learning model. However other choices (e.g. [11] that requires processing many features) are too slow for an active framework.

Graph Cut Refinement. In addition, we also compute a refined segmentation by again making use of multi-label alpha expansion [50] by solving:

$$\min_{l_f, f\in F} \sum_{f\in F} \phi_D(f, l_f) + \lambda \sum_{f\in F, f'\in N_f, l_f \neq l_{f'}} \phi_S(f, f'), \tag{6}$$

where λ is a non-negative constant used to balance the terms and $\phi_D(f, l_f) = \exp(-\boldsymbol{p}_f(l_f))$ penalizes low probability of assigning a label l_f. The second term, $\phi_S = (\pi - \theta_{ff'})/\pi$, penalizes adjacent faces which form concavities, where $\theta_{ff'}$ is the dihedral angle between the face normals of faces f and f', and is normalized to $[0, 1]$. This refinement technique has been used in recent work [7,8,11], but is slightly modified in comparison. The output of the refinement is a segment label per face, we call this the refined segmentation.

(a) High ranking shapes

(b) Low ranking shapes

Fig. 8. Visual comparison of high (a) and low (b) ranking shapes when ranking according to entropy.

Note that graph-cut refinement here is a post-processing step that refines the segmentation whenever there is a machine prediction. The Interactive Boundary Refinement (Section 4.5) is a user-driven refinement process to help obtain ground truth annotation (i.e. after graph-cut refinement to further reduce user interaction (see Fig. 1)). Though both Eqs. (1) and (6) are solved by the multi-label alpha expansion, they have different focuses. The graph-cut refinement (Eq. (6)) refines all segmentation labels globally on *all* faces. The interactive algorithm (Eq. (1)) mainly optimizes labels of the faces *close to* the label boundaries. It can also be activated multiple times with adjustable emphasis on the SDF term by the users.

4.7. Effective table ordering

Each time an evaluation of the model is completed, the user is presented with the results. There are several options to view the results. These options are spread across three menus, where one option per menu is selected at a given time.

Displayed Segmentation. The first menu governs which segmentation is displayed on each model in the table. There are two options to choose from: predicted segmentation and refined segmentation. These are the outputs from the deep learning model and the graph cut refinement, respectively.

Ordering Algorithm. The second menu controls how the table is ordered. There are two options; no order and entropy. Given the probability matrix $\boldsymbol{p}$, of shape S, the entropy score is computed as:

$$E_S = \sum_{f\in S} \frac{\sum_{\boldsymbol{p}_f^l \in \boldsymbol{p}_f} -\boldsymbol{p}_f^l \log(\boldsymbol{p}_f^l)}{n_f^S} \tag{7}$$

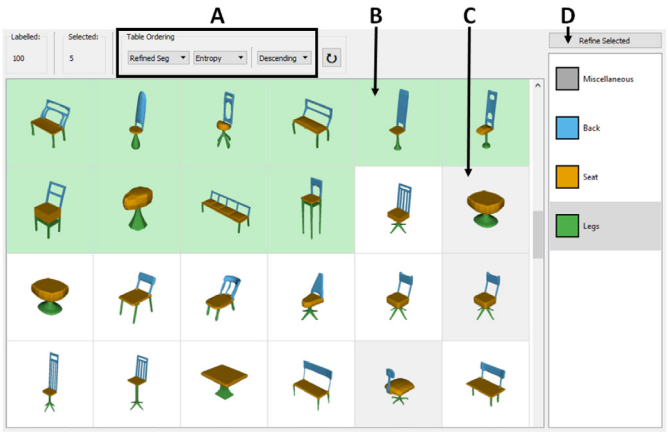


Fig. 9. Table interface of our system, which allows for quick analysis of the entire dataset with an effective ordering method. **A** Table ordering (Section 4.7). **B** Shapes shown in green have full segmentation and have been user verified. These are used to train the deep learning model. **C** The table allows for selection of shapes for manual refinement and verification. **D** Visualize the selected shapes in the annotation interface (Fig. 10). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

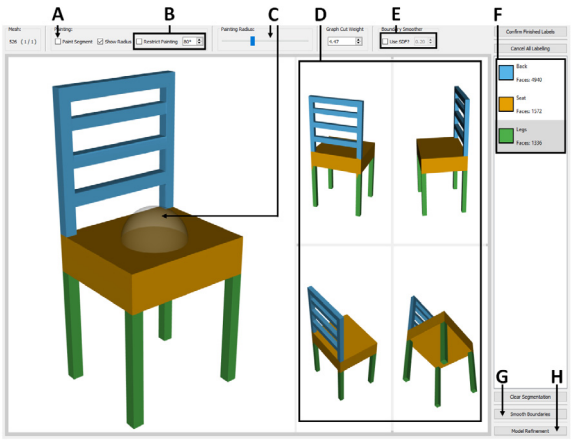


Fig. 10. Annotation interface of our system, where the user can label or refine a subset of shapes. **A** Segment wide paint (Section 4.4). **B** Painting restriction (Section 4.4). **C** Paint radius with visual indicator. **D** Multiple shape views for quick segmentation analysis (Section 4.4). **E** Weight of the SDF influence on the boundary refinement. **F** Segment names, colors and face counts. Selected (gray) segment will be assigned to faces when painting. **G** Boundary refinement (Section 4.5). **H** Model refinement (Section 4.6). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where n_f^S is the number of faces in shape S, $\boldsymbol{p}_f$ are all probabilities for face f, and $\boldsymbol{p}_f^l$ is the probability of face f being assigned label l. As we do not have ground truth labels to evaluate the performance of the remaining unlabeled shapes, we need another measure for ranking the shapes. Entropy is a measure of uncertainty of a probability distribution, therefore it is a natural alternative. For a shape we measure the entropy of each face and then average it across all other faces. This provides a score, which we then use to order the whole dataset.

Order Direction. The final option controls if the data is presented in ascending (worst to best) or descending (best to worst) order. Effective use of these different ordering methods can greatly reduce the time needed to label a dataset. The ordering is based on the model's predictions. Manually refining shapes which the model has trouble segmenting allows the model to generalize more quickly to the rest of the dataset. A visual comparison of entropy ranking is shown in Fig. 8, which allows the user to quickly see shapes which the model is good/bad at segmenting. Note that the ordering tries to provide a consistent view in the table whilst annotators can still choose freely what to annotate next, allowing human analytics to be involved [54].

5. Interface and program flow

We provide a system with many useful tools for interactively segmenting a dataset of shapes. When using the system, the user is presented with two main interfaces, shown in Figs. 9 and 10. These interfaces dynamically change depending on which stage in the program pipeline (Fig. 1) the user is currently at. Here we outline the program flow from each interface to the tools provided as the user progresses through the pipeline.

New Dataset. This is the entry point of the system, where the user will be presented with the table interface showing all shapes in the dataset. At this point, the user will be able to initialize the different segments the dataset will contain. From here, the user needs to select starting shapes to manually segment. They have two options to pick the subset: arbitrarily pick them from the table, or use our initial shape selection tool (Section 4.3).

Coarse Segmentation. Given the subset of selected shapes, the user is now tasked with manual segmentation, this is done in the annotation interface. This is the part of the pipeline which requires the most user time, as such, we provide many options. To quickly assign coarse labeling, each shape is over-segmented, and we provide two options for this (Section 4.4 Shape Over-segmentation), allowing the user to control over how many patches are generated. The user then assigns segment labels to the patches by clicking a segment with a specified label. They do not need to label all patches, and can transit to the next stage by using the interactive boundary refinement algorithm to label the remaining segments and smooth the boundaries.

Segmentation Refinement. In this stage the annotation interface changes to allow for 'painting' (shown in Fig. 10). This is where the segmentation of a shape is completed to the user's satisfaction. The interactive boundary refinement algorithm can be used as often as needed to automatically adjust boundaries, and the user can then fix any small segmentation defects that exist. The user can mark the shape as complete (shown by a green border around the shape) and move on to another shape. Once all shapes in the subset are completed they are then stored as ground truth, and marked in green on the table interface.

Model Training and Evaluation. Once a number of shapes are segmented, the deep learning model can be trained and evaluated. This can be done at any time and allows for the table to be ordered much more effectively. For subsequent retraining, users can segment any number of shapes because our training and prediction technique uses an ensemble of trained models that uniformly sample batches (each with 512 samples) from the entire pool of data (see Section 4.6).

Selecting the next subset. Just like starting with a new dataset, the user can arbitrarily pick shapes from the table, or use our initial shape selection tool (Section 4.3). In addition, the table can now be ordered to rank the shapes according to entropy (Section 4.7). The table can also be used to display the predicted or refined segmentation (Section 4.6). This can be useful, as the user can see shapes that the model has correctly segmented and quickly confirm them. Also, by using the ranking, the user can see shapes that the model could not segment well, and they can then select these as part of the next subset. From this stage the user can either segment the subset from scratch by using **Coarse Segmentation**, or refine the predicted or refined segmentation using **Segmentation Refinement**.

Table 1
5-fold cross validation on the COSEG large datasets [7] using different learning schemes and refinement techniques. (5) denotes only the last 5 snapshots were used for evaluation [20]. **Bold** values denote the highest accuracy for the set and refinement method.

	No Refinement			Graph Cut Refinement			Boundary Refinement		
	Chairs	Vases	Aliens	Chairs	Vases	Aliens	Chairs	Vases	Aliens
	92.19	89.98	90.87	96.76	92.73	94.62	96.66	92.80	94.56
10 Snapshots (5)	91.73	89.96	90.64	96.67	92.34	94.70	96.79	92.78	94.66
5 Snapshots	**92.68**	**90.19**	90.91	**97.18**	**92.75**	**95.02**	**97.03**	**92.81**	**94.78**
3 Snapshots	92.27	90.01	**90.99**	97.10	92.53	94.82	96.93	92.74	94.73
1 Snapshot	91.43	89.14	90.14	96.63	91.57	94.28	96.93	91.91	94.67
Fixed Learning Rate	79.49	87.37	86.22	83.41	90.19	90.79	83.99	90.73	91.50
Decaying Learning Rate	87.34	86.42	88.05	93.93	89.42	92.92	94.45	89.83	93.63

The above program flow iterates and as the user confirms more shape segmentations, the model has access to more training data and better generalizes, reducing the future interaction effort (Fig. 2).

6. Results and discussions

In this section we will evaluate our interactive system. There are several key components that make up our system, here we will provide experiments and results carried out to evaluate the individual components.

6.1. Deep learning model

While any appropriate deep learning model or classification algorithm could be used with our system, this work also proposes a deep learning architecture, which is both fast and effective for 3D shape segmentation. To evaluate our model and design choices we include results from several experiments. These evaluate (i) the use of ensemble based learning, (ii) the performance of the deep learning model and (iii) the performance of the interactive refinement techniques. We use PSB [12] and COSEG [7] and their ground truth in this section.

Firstly, we evaluate the choice of an ensemble based learning scheme. We performed 5-fold cross validation of the 3 large COSEG datasets [7] with varying numbers of snapshots. For comparison, we also performed the same experiments with a fixed learning rate and decaying learning rate. We include these as examples of typical learning rate values for model training. The starting learning rate in all experiments was 0.01. All ensemble experiments used Eq. (5) to update the learning rate. The decaying learning rate experiment reduced the learning rate by a factor of 10 at 50% and again at 75% of the training process. The results are shown in the No Refinement columns of Table 1. As the columns show, using a snapshot learning scheme consistently improves results when compared to fixed and decaying learning rate. There is also a considerable increase in accuracy when only using a single snapshot, which shows that the cosine learning function (Eq. (5)) alone improves the quality of the trained model. In the experiment, each snapshot model is trained for 5000 iterations, regardless of the number of snapshots. We limit 5000 iterations because of speed concern as it is required by our active framework. Specifically, in the "5 Snapshots" model, each snapshot was trained 1000 iterations. In the "10 Snapshots (5)" model where the last 5 snapshots were used, each snapshot was only trained 500 iterations. These give less time for the "10 Snapshots (5)" to converge to local minima in the optimization, and thus the lower performance. Finally, the results show that in the majority of cases "5 snapshots" give the best performance increase. The setting is used in the remaining experiments.

Next we evaluate the accuracy of our deep learning architecture. We devised two sets of experiments; leave-one-out cross validation on the PSB dataset [12] and 5-fold cross validation on the COSEG dataset [7]. We used the recent work from [11] as a comparison as they provide results from several feature driven deep learning architectures, and they also include results for the 2D CNN from [8]. Tables 2 and 3 show the results of the experiments. As our model architecture was designed with speed in mind we compare against existing models that can be trained quickly (Principal Component Analysis (PCA) & Neural Network (NN), 2D CNN). Table 2 shows that our proposed architecture has an increase of 3.5% over PCA & NN and a moderate increase of 1.3% over a 2D CNN [8]. Also when compared to a much bigger network [11], which uses more than 4× more input features and takes considerably longer to train, our proposed method's performance is still within 1% (on average) difference. Similarly, in Table 3 our proposed method again outperforms the 2D CNN, with an increase of 4% when looking at large datasets. These results show that our proposed model offers overall comparable-slightly better accuracy whilst being fast and using less features. It is thus suitable for our proposed active learning framework.

Table 2
Leave-one-out cross validation on the PSB dataset [12]. PCA & NN, 2D CNN [8] and 1D CNN results from [11]. (·) indicates the number of labels.

	PCA & NN	2D CNN	1D CNN	Proposed
Airplane (4)	92.53	94.56	96.52	95.22
Ant (5)	95.15	97.55	98.75	98.75
Armadillo (11)	87.79	90.90	93.74	94.99
Bird (5)	88.20	86.20	91.67	88.64
Chair (4)	95.61	97.07	98.41	97.61
Cup (2)	97.82	98.95	99.73	98.12
Fish (3)	95.31	96.16	96.44	96.43
Fourleg (5)	82.32	81.91	86.74	84.55
Glasses (3)	96.42	96.95	97.09	98.10
Hand (6)	70.49	82.47	89.81	88.21
Human (8)	81.45	88.90	89.81	90.66
Octopus (2)	96.52	98.50	98.63	98.71
Plier (3)	91.53	94.54	95.61	95.32
Table (2)	99.17	99.29	99.55	98.99
Teddy (5)	98.20	98.18	98.49	98.57
Vase (4)	80.24	82.81	85.75	82.87
Average	90.61	92.79	94.80	94.11

The improvements compared to 2DCNN [8] and 1DCNN [11] also show the strength of using 2D features rather than stacking many heterogeneous features as in [8,11]. As mentioned in [11], CNN pooling may be affected by the boundary issues within the stacked heterogeneous feature representations. While [11] alleviates it by using only 1D features, we use only 2D SC and SI features which avoid the issues. These explain the overall good performance.

Finally, we evaluate our refinement techniques, Graph Cut Refinement (end of Section 4.6) and Interactive Boundary Refinement (Section 4.5). As the outputs of our ensemble experiments were reported without any refinement, we pass these outputs through our two refinement algorithms. The Graph Cut Refinement and (Interactive) Boundary Refinement columns of Table 1 show the results of the experiments. As the columns show, both techniques show a considerable increase in accuracy compared

Table 3
5-fold cross validation on the COSEG dataset [7]. 2D CNN [8] and 1D CNN results from [11]. (·) indicates the number of labels.

	2D CNN	1D CNN	Proposed
Candelabra (4)	91.55	93.58	91.35
Chairs (3)	93.48	97.75	94.82
Fourleg (5)	90.75	94.12	92.40
Goblets (3)	92.79	97.80	87.40
Guitars (3)	97.04	98.03	96.23
Irons (3)	80.90	89.89	85.96
Lamps (3)	81.52	86.74	91.44
Vases (4)	89.42	92.47	86.08
Average	89.68	93.80	90.71
VasesLarge (4)	87.57	95.88	92.81
ChairsLarge (3)	92.68	97.71	97.18
AliensLarge (4)	91.93	97.84	95.02
Average	90.73	97.14	95.00

to the un-refined results. The refined results also further support the use of 5 snapshot based ensembles, providing the highest accuracy across all results. Note that the interactive boundary refinement algorithm can be executed iteratively as the data term is based on the position of segment boundaries, which will change between runs. The results shown are after a single run of the interactive boundary refinement, so it is possible for further improvements in the results using an iterative approach.

6.2. Entropy ranking

Given the model predictions, optimally selecting the next set of shapes to refine is key to efficiently labeling a dataset. It might seem logical to select shapes that were predicted well by the model, as annotators can work on them quickly. However, it may be more beneficial, overall, to select the shapes the model predicted poorly, as these are the ones that would make the training set more diverse and help the model generalize.

Our entropy ranking (Section 4.7) allows for effective ordering of the entire dataset based on the predictions of the model. We are interested in evaluating the long term effects that selecting high ranking or low ranking shapes has on the deep model. We devised four experiments to test the entropy ranking strategies: 'Lowest Ranked' first, 'Highest Ranked' first, 'Mixed Ranked', and 'Random'. We run all experiments on the three large COSEG datasets.

Our experiments used five-fold cross validation. Before all experiments, we split the COSEG data into "training set + evaluation set" (80%) and "testing set" (20%) for each fold. Here, we define the "training set" as the set of shapes with known labels. They are used to train the deep model in our experiments. The "evaluation set" is the set of shapes with unknown labels. These are shapes yet to be annotated or confirmed by the users. In our iterative training procedure, we picked shapes from the "evaluation set" (according to one of the entropy ranking strategies/experiments discussed below) and have them annotated. Then we move them to the training set. Note that the "evaluation set" will decrease in size during our training procedure. Thus we do not use the phrase "validation set" to avoid confusion. The "testing set" stays fixed throughout the experiments.

In each of the four experiments, we train our deep model iteratively using 3D shapes suggested by the entropy rankings. We start with 10 shapes in the training set, which were selected using our LFD HOG embedding (Section 4.3). The starting training set of each fold was fixed across all experiments for a dataset for fairness. Given the starting training set, the model is trained and evaluated on both the evaluation and test sets, then the evaluation set is ranked according to entropy. To carry out these experiment consistently, we use the ground truth labels from the COSEG datasets instead of asking users to annotate them. These 10 shapes are then moved to the training set. The shapes that are moved depend on the experiment: 'Lowest Ranked' moves the 10 shapes which had the lowest entropy, 'Highest Ranked' moves the 10 shapes which had the highest entropy, 'Mixed Ranked' moves 5 highest and 5 lowest ranked shapes, and 'Random' moves 10 shapes at random. This process is then repeated until all shapes in the evaluation set are exhausted. The accuracies of both the evaluation and test sets are recorded each time the model is trained and evaluated, with the results shown in Fig. 11. (To plot Fig. 11(a), we stop when there is 10 shapes remaining in the evaluation set.)

The results shown in Fig. 11(a) are the evaluation accuracies for each experiment and each dataset. As shown, choosing the best ranked shapes will give poor long-term results. This is because the highest ranking shapes are typically similar to shapes already in the training set, so adding these will cause the model to over fit and not generalize. Inversely, choosing the worst ranking shapes gives the best long-term results, as they are typically shapes that have large variation to the training set. Adding these shapes will allow the model to generalize better and prevents over fitting. Finally, the Mixed Ranking and Random results perform similarly, as selecting shapes randomly is likely to contain both high and low ranking shapes, similar to evenly selecting high and low ranking shapes. Additionally, in Fig. 11(b), we show the accuracies of the test set as the training set grows. This shows that all methods except selecting the best ranking methods provide a model which generalizes well when introducing new data to the dataset. Similarly, the worst ranking strategy often generalizes faster among the four.

6.3. Usability and user study

To test the usability of our system we conducted an in-lab user study which obtained interaction times, numbers of clicks and accuracies. We selected 11 participants with good computer skills and provided them with instructions and a demo of how to use the system. 10 of the participants were asked to segment the COSEG [7] ChairsLarge dataset for approximately 1 hour. We chose this dataset as it contains 400 shapes and has a well defined ground truth segmentation for evaluation of the results. Half of the participants were given the full system, while the other half had the boundary refinement feature disabled. We did this to monitor the usefulness of the feature and its impact on the resulting segmentations. The final participant was asked to segment 6 of the small COSEG datasets, namely, Candelabra, Chairs, Goblets, Guitars, Irons and Lamps. The aim of this experiment was to record times to achieve certain set accuracies for comparisons to previous works.

The results from the ChairsLarge user studies are shown in Figs. 12 and 13. As the results show, the deep learning model quickly gives good performance when evaluated on the remaining shapes, requiring a training set of only 10% of the dataset to achieve over 80% accuracy (Fig. 12). Additionally, Fig. 13 shows that the required time (a) and interactions (b) to label a shape becomes considerably lower as the model generalizes. Further, there is a significant reduction in labeling times and interactions for the participants who had the boundary refinement feature enabled. This is due to the interactive boundary refinement algorithm providing accurate label outputs.

Comparison to previous work. Our user study also provided data for comparison to the two previous works, Active Co-analysis (ACA) [19] and Scalable Active Framework (SAF) [14]. The participant was asked to use the full system and completely label 6 datasets to ground truth level. The times of these experiments

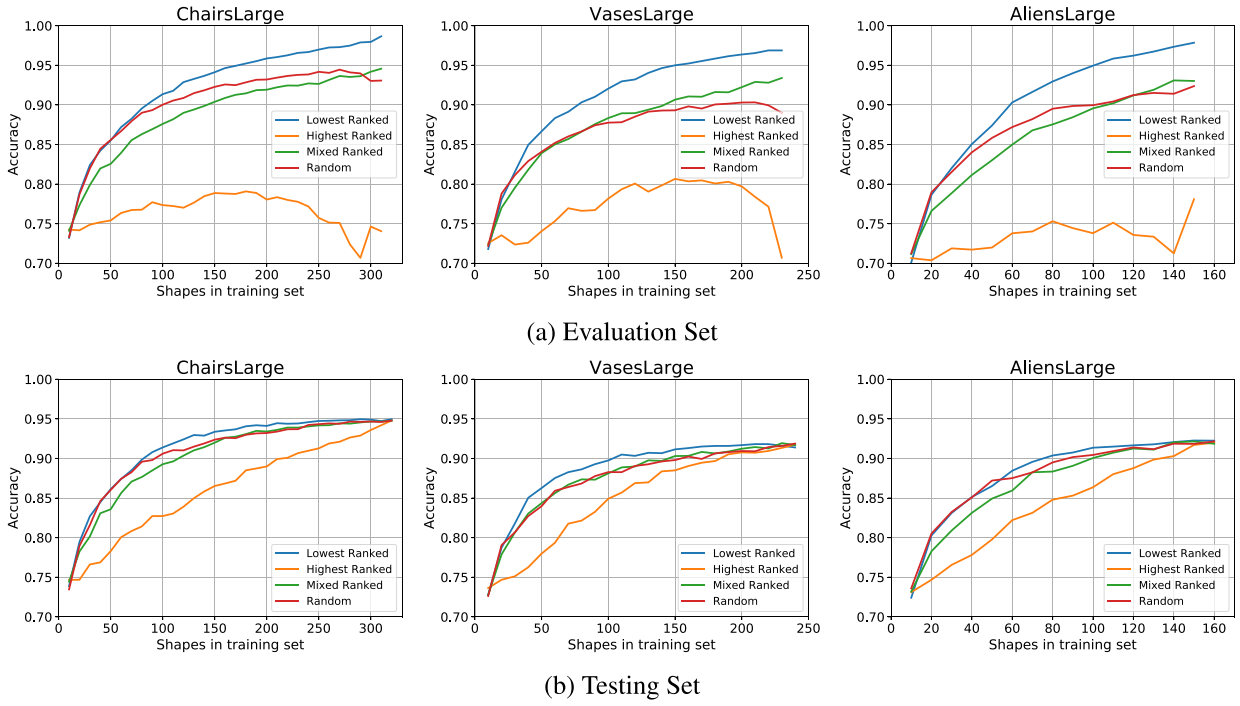


Fig. 11. Results of running our entropy experiments (see text) on the large COSEG datasets [7]. Each graph shows the average accuracy of all runs for different experiments on different datasets. Shapes are moved from the evaluation set to the training set based on the entropy rank and experiment, while the testing set remains constant throughout the experiment. Each experiment consists of a 5-fold cross validation, where the omitted fold is the testing set and the remaining folds are the training and evaluation sets.

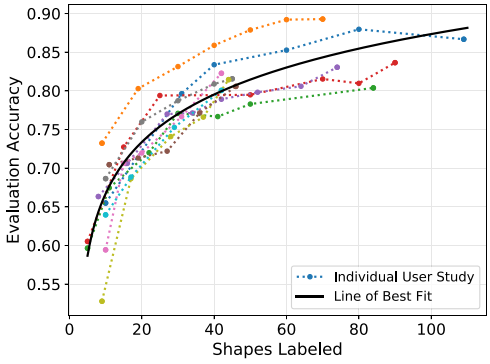


Fig. 12. User study results showing accuracy of the evaluation set as the number of shapes in the training set grows. The evaluation set consists of shapes that have not been confirmed by the user. The graph shows that the model generalizes quickly, requiring less work from the user to achieve ground truth accuracy.

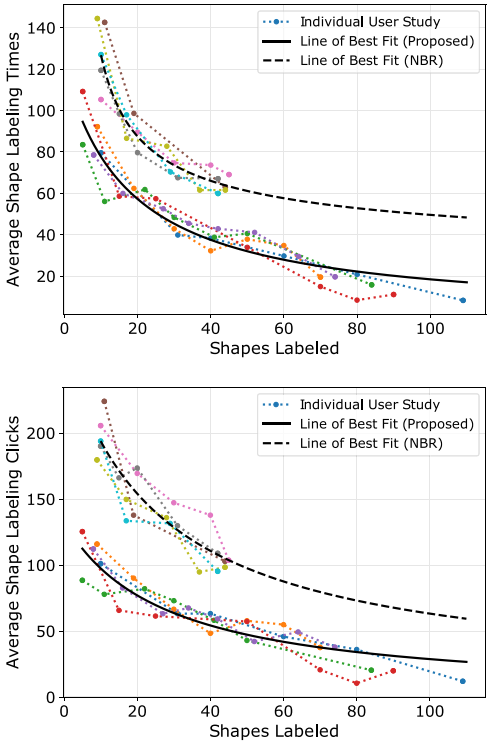


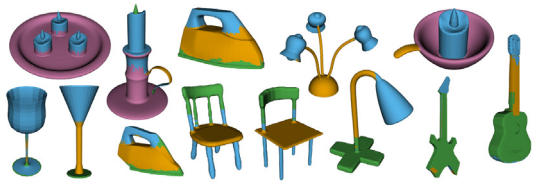
Fig. 13. User study results showing interaction time (in seconds) and number of clicks decreasing as more shapes are added to the training set. The charts also show that users who had the interactive boundary refinement feature disabled (NBR), took much longer and required more clicks to achieve the same segmentations, resulting in significantly fewer shapes segmented in the same time frame.

were recorded as the dataset accuracy passed certain milestones (i.e. 95%, 98%, etc.) and the results are shown in Table 4. As shown, our method is comparable to ACA and slightly slower than SAF at achieving a 95% accuracy. However, as the purpose of this work is providing a tool for efficient ground truth generation, 95% accuracy is not a good enough segmentation. We show this in Fig. 14, where the average accuracy of the set may be 95%, but certain individual shapes show very poor segmentations. For these reasons we also provide timings to achieve higher set accuracies, including a ground truth level (100%). Our method can achieve this level of segmentation as it not only asks the user to verify the results but correct any mistakes with a refinement stage.

Computation Time. Our estimated computation times are based on using our system to label the COSEG ChairsLarge dataset. Our pre-processing stage consists of manifold checking (<0.1s per shape) and feature extraction (∼40s per shape). Then, our

Table 4
Comparison of user interaction times (in minutes) for achieving certain dataset accuracies. We compare our method with the two previous works, ACA [19] and SAF [14] (* denotes estimated times, see original papers). While our method performs similarly to ACA and slightly worse than SAF at 95%, we strive for high-quality segmentations and good boundaries. Furthermore, reporting 95% accuracy is not ground truth level, so we also report times to achieve accuracies up to ground truth level.

	ACA	SAF	Proposed			
(in minutes)	95%	95%	95%	98%	99%	100%
Candelabra	7.00	1.40	5.56	6.34	7.47	8.17
Chairs	10.50*	0.90*	1.30	1.99	2.21	2.88
Goblets	1.20*	0.70*	1.01	1.51	1.51	1.54
Guitars	1.80*	1.90*	2.37	5.58	6.35	9.19
Irons	7.60*	7.20*	2.63	3.27	3.27	3.58
Lamps	0.60*	2.30*	3.14	3.82	4.54	5.26



(a) 95% accuracy

(b) Ground truth accuracy

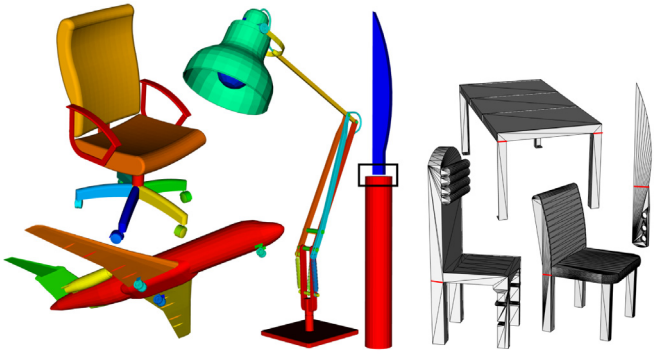
Fig. 14. Visual comparison of segmentation results from sets labeled to 95% accuracy (a) and to ground truth level (b). This shows that a set labeled to 95% accuracy still requires significant work to complete, therefore times to achieve this accuracy are less meaningful.

deep learning model takes ∼90s to train (This time is fixed due to our training scheme) and <0.25s per shape to evaluate and refine. In future we would refine the training and evaluation process to run concurrently with user interactions, greatly reducing the processing time. This scales linearly with the size of the dataset. All timings are reported using a 4-core 4 GHz Intel Core i7, 32 GB of RAM and an Nvidia GTX 1080Ti with 11 GB of VRAM.

6.4. ShapeNet labeling

ShapeNet [13] is a massive online repository of 3D shapes used frequently in shape retrieval and matching techniques. The repository contains thousands of 3D shapes from dozens of shape categories giving shape analysis algorithms the potential to be evaluated on widely diverse datasets. Recently, shape segmentation techniques have also begun to use ShapeNet datasets for benchmarking their proposed algorithms [32,33], this was initiated by the ground truth labels from [14].

However, due to the nature of such a large repository of shapes, many of the shapes are not manifold and consist of a large number of disconnected regions or even polygon soups. Due to this, many techniques have shifted to a point cloud driven algorithm, which sacrifices much of the information that can be obtained from a manifold shape. Furthermore, many mesh driven techniques are limited to point cloud ground truth segmentations (provided by [14]), as extracting a meaningful segmentation on the provided shapes is challenging due to low face counts, poor geometry and miscellaneous parts (see Fig. 15).



(a) Multiple components (b) Low resolution

Fig. 15. Examples from ShapeNet where shapes have multiple components (a), and are low resolution (b). Different components are denoted by different colors and red lines show where segment boundaries would lie. We also show a case where two components have a significant gap between them (black box, (a)). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 5
Re-meshing and labeling statistics for ShapeNet datasets. For each dataset we report the number of shapes (N), number of successfully re-meshed shapes (NR), number of labels (L) and the (user interaction) time to label the dataset in hours (T). Any times shown with (*) are estimated based on labeling the dataset for 1 hour.

	N	NR	Proposed		SAF	
			L	T (H)	L	T (H)
Airplane	4027	4009	6	24.1*	4	22.6*
Bag	83	75	2	0.4	2	0.2*
Cap	56	55	2	0.3	2	0.2*
Earphone	73	60	4	0.3	2	0.2*
Guitar	793	794	3	3.0*	3	2.8*
Knife	426	420	2	1.7*	2	1.5*

To evaluate the usability of our system on very large datasets, we also try to use ShapeNet [13] datasets. However, as many of the shapes are non-manifold we have re-meshed several of the datasets for these experiments. (Our re-meshing procedure can be found in the supplementary materials. Other sophisticated techniques like [55–57] can be used interchangeably). We chose the Airplane and Guitar datasets, and included 4 smaller datasets for a more thorough evaluation of the re-meshing quality.

Our experiments are formulated similarly to our User Studies (Section 6.3), where our system was used for up to one hour of user time for each dataset, and users are asked to aim at ground-truth level. Table 5 shows the timings achieved when labeling the 6 ShapeNet datasets (10K face counts, times shown with (*) are estimated based on 1 hour of user time). The table also shows the number of shapes that were successfully re-meshed and the number of labels the dataset has.

The results show that our method is slightly slower than SAF (based on results estimated in [14], see original paper). Note however that, in the experiment, we label the Airplane and Earphone datasets with an increased number of segments (6 instead of 4 for Airplane, 4 instead of 3 for Earphone). For SAF to achieve this number of segments, much more user time would be required (each new label requires a full pass of the dataset to be processed). Therefore, our running time is likely to be faster than what SAF would achieve with the same segmentations. For all sets, we also emphasize the high-quality segmentation we can achieve (100% accuracy of ours vs 95% accuracy of SAF).

We further run an experiment on the Airplane set and invite an annotator to carry out 800+ annotations (6 labels). The result is shown in Fig. 16. As SAF [14] did not release source code,

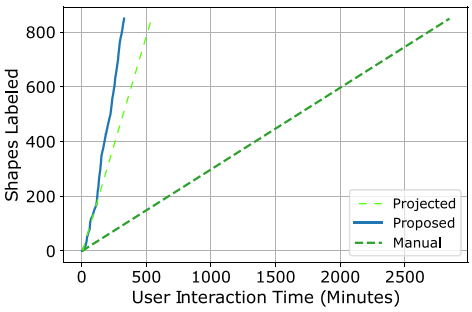


Fig. 16. Timing for annotating 800+ shapes with 6 labels (Airplane Set). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

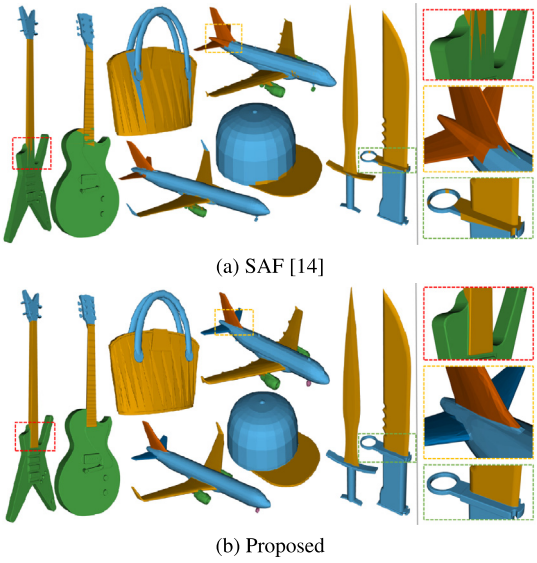


Fig. 17. Visual comparison of segmentation results from ShapeNet datasets. We compare provided segmentations from [14] (a) to segmentations generated by our proposed framework (b). Highlighted regions are shown on the right in a zoomed view. As is shown, our method can provide much more accurate segment boundaries with our refinement step.

we only analyze our results qualitatively. As shown in Figure 2 of [14], the timing required for SAF to annotate 400 shapes follows a superlinear curve against the number of shapes labeled. Our method however can practically achieve good performance for a larger set (800+, i.e., more than double the number of shapes even with more (6) labels). Fig. 16 shows that our timing is roughly linear with a small gradient against the number of shapes labeled. We also show the green dash line which is a projected line using initial data. This shows that the time required for later annotations reduces. Overall, the reduced interaction time is due to a deep model that improves over time. These justify the usefulness of our framework.

We additionally compare the quality of the output segmentation of each system. As Fig. 17 shows, there is a significant difference in segment boundary quality between the two methods. Our method maintains good quality boundaries, while the segmentation from SAF is poor in some regions. While this poor segmentation could be due to point cloud resolution or label projection, Fig. 18 shows that there are also many cases of poor labeling on the point cloud. Also note that SAF's labeling does not partition the surface. There may be overlapping labels or labels that do not jointly cover the shape [14]. Our active learning framework however can achieve ground truth-level quality.

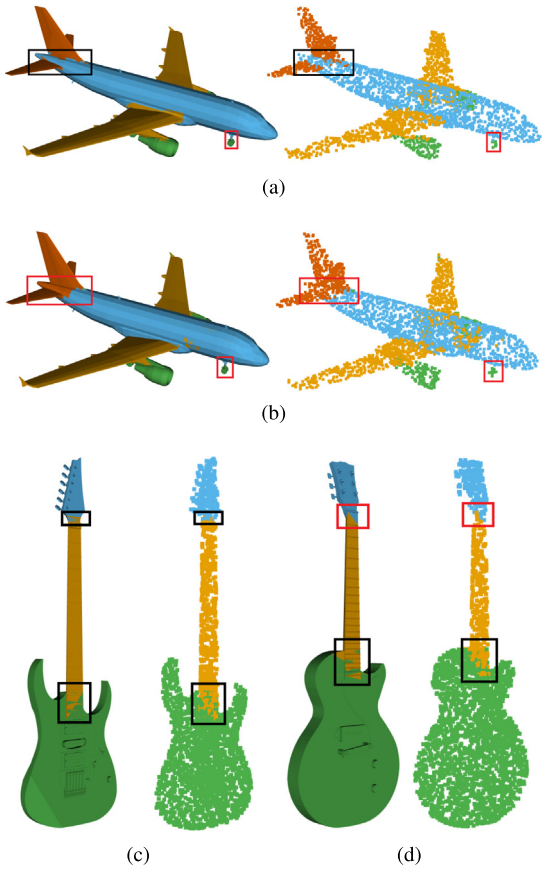


Fig. 18. Comparison between provided ShapeNet labels from SAF [14], when displayed on point clouds or projected onto the original mesh. While there are cases where point cloud resolution impacts the projection (black), there are also many incorrectly labeled sections (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Conclusions

The research development of 3D shape segmentation is often slowed down by the lack of ground truth annotation, and it, in turn, is due to long annotation time and the lack of tools to define good quality shape segmentation boundaries. Researchers often need to pay professionals or rely on crowd-sourcing to obtain data. It would lead to high cost and quality control issue.

In this work, we present an efficient and accurate active learning framework to tackle these issues together. The idea is to obtain ground-truth level annotation with interactive machine co-assistance. To achieve this we combine three core systems: a fast and effective deep learning model, effective shape ordering and selection, and accurate refinement tools. These components are combined to create an iterative interactive active framework, which becomes more effective over time. The framework reduces human interaction efforts whilst achieves a ground truth dataset with 100% accuracy.

We have shown that our active learning framework is not only more accurate than the current state-of-the-art, but also more efficient for datasets with larger number of distinct segments. We also demonstrate that our system can scale with large datasets, allowing for quick and meaningful segmentation of large shape collections.

7.1. Limitations and future work

A trade-off we made with our pipeline was full dataset evaluation. This enables us to provide a powerful shape ordering tool at the cost of processing time. While current dataset sizes do not pose a major time delay for evaluation, as datasets continue to grow, it could soon be an issue. There are several ways we could resolve this while still maintaining effective shape ordering. One way would be to only evaluate on a subset of the data. The selection of the subset would then be key to maintaining effective shape ordering. A solution would be using global shape descriptors to select shapes both similar and dissimilar to shapes in the training set. However, the size of the required subset would still need to be large so that the user has enough diversity when selecting shapes to refine. A better solution would be to train and evaluate in the background. This solution would minimize any user down time while still providing an always up-to-date table. As shapes are confirmed they can be trained on quickly, then shapes can be evaluated (according to shape similarity) and the table can be dynamically updated.

Another trade-off we made was requiring manifold shapes. While re-meshing software is available, and we show a working method in this work, this is still not ideal. The main reasons we require manifold shapes are feature extraction and user painting. Other works have converted the shapes to point clouds, and while this fixes any topology issues, there is still information loss in this process. Another solution would be to introduce artificial edges in the shapes to join the components. While this does not solve all the problems in ShapeNet datasets (such as zero-thickness parts and low resolution), it would allow for features to be extracted and painting between parts.

Our deep model at the moment only works on manifold shapes. It requires our remeshing process to handle shapes with defects and noise. Though our model is fast, it also depends on handcrafted SC and SI features. We believe a more reliable, fast and data-driven method that can directly and flexibly derive deep features from meshes, point-clouds and polygon soups would further improve the active learning framework in the future.

Given the recent interests in hierarchical semantic segmentation, grouping and labeling of shapes, primitive and abstraction, we believe our framework would be extended to obtain valuable ground truth in a cost-efficient way (e.g., by developing a fast learning model and a more intuitive interface) for these new research and unseen datasets in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We would like to thank all participants who help in the annotation experiments. David George was fully-funded by a PhD Studentship from the Engineering and Physical Sciences Research Council, UK (EPSRC DTA: Ep/M506424/1, Ep/L504865/1).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2021.103179.

References

[1] Kleiman Y, van Kaick O, Sorkine-Hornung O, Cohen-Or D. SHED: Shape edit distance for fine-grained shape similarity. ACM Trans Graph 2015;34(6):235:1–235:11.
[2] Shapira L, Shalom S, Shamir A, Cohen-Or D, Zhang H. Contextual part analogies in 3D objects. Int J Comput Vis 2010;89(2–3):309–26.
[3] Chen X, Zhou B, Lu F, Wang L, Bi L, Tan P. Garment modeling with a depth camera. ACM Trans Graph 2015;34(6):203.
[4] Tulsiani S, Su H, Guibas LJ, Efros AA, Malik J. Learning shape abstractions by assembling volumetric primitives. In: Proc. IEEE conf. CVPR, 2017.
[5] Ganapathi-Subramanian V, Diamanti O, Pirk S, Tang C, Niessner M, Guibas L. Parsing geometry using structure-aware shape templates. In: International conference on 3D vision, 2018, p. 672–81.
[6] Kalogerakis E, Hertzmann A, Singh K. Learning 3D mesh segmentation and labeling. ACM Trans Graph 2010;29(3):102:1–102:12.
[7] Sidi O, van Kaick O, Kleiman Y, Zhang H, Cohen-Or D. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In: Proc. ACM SIGGRAPH ASIA, Vol. 30, 6, 2011.
[8] Guo K, Zou D, Chen X. 3D mesh labeling via deep convolutional neural networks. ACM Trans Graph 2015;35(1):3:1–3:12.
[9] Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 2017, arxiv preprint.
[10] Yi L, Su H, Guo X, Guibas L. SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. 2016,
[11] George D, Xie X, Tam GK. 3D mesh segmentation via multi-branch 1D convolutional neural networks. Graph Models 2018;96:1–10.
[12] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation. ACM Trans Graph 2009;28(3):73:1–73:12.
[13] Chang AX, Funkhouser TA, Guibas LJ, Hanrahan P, Huang Q, Li Z, et al. Shapenet: An information-rich 3D model repository. CoRR 2015. arXiv:1512.03012.
[14] Yi L, Kim VG, Ceylan D, Shen I-C, Yan M, Su H, et al. A scalable active framework for region annotation in 3D shape collections. ACM Trans Graph 2016;35:1–12.
[15] Klokov R, Lempitsky VS. Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. 2017, arXiv preprint arXiv:1704.01222.
[16] Shapira L, Shamir A, Cohen-Or D. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis Comput 2008;24(4):249–59.
[17] Hanocka R, Hertz A, Fish N, Giryes R, Fleishman S, Cohen-Or D. MeshCNN: A network with an edge. Proc ACM SIGGRAPH 2019;38(4).
[18] Yuan Y, Lai Y, Yang J, Duan Q, Fu H, Gao L. Mesh variational autoencoders with edge contraction pooling. In: IEEE/CVF conference on computer vision and pattern recognition workshops, 2020, p. 1105–12.
[19] Wang Y, Asafi S, van Kaick O, Zhang H, Cohen-Or D, Chen B. Active co-analysis of a set of shapes. ACM Trans Graph 2012;31(6):165.
[20] Huang G, Li Y, Pleiss G, Liu Z, Hopcroft JE, Weinberger KQ. Snapshot ensembles: Train 1, get M for free. 2017, arXiv preprint arXiv:1704.00109.
[21] Ben-Chen M, Gotsman C. Characterizing shape using conformal factors. In: 3D obj. retrieval. 2008, p. 1–8.
[22] Belongie S, Malik J, Puzicha J. Shape matching and object recognition using shape contexts. IEEE Trans Pattern Anal Mach Intell 2002;24(4):509–22.
[23] Xie Z, Xu K, Shan W, Liu L, Xiong Y, Huang H. Projective feature learning for 3D shapes with multi-view depth images. Comput Graph Forum 2015;34(7):1–11.
[24] Shlafman S, Tal A, Katz S. Metamorphosis of polyhedral surfaces using decomposition.. Comput Graph Forum 2002;21(3):219–28.
[25] Attene M, Falcidieno B, Spagnuolo M. Hierarchical mesh segmentation based on fitting primitives. Vis Comput 2006;22(3):181–93.
[26] Hu R, Fan L, Liu L. Co-segmentation of 3D shapes via subspace clustering. Comput Graph Forum 2012;31(5):1703–13.
[27] Meng M, Xia J, Luo J, He Y. Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization. Comput Aided Des 2013;45(2):312–20.
[28] Wu Z, Wang Y, Shou R, Chen B, Liu X. Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering. Comput Graph 2013;37(6):628–37.
[29] Shu Z, Qi C, Xin S, Hu C, Wang L, Zhang Y, et al. Unsupervised 3D shape segmentation and co-segmentation via deep learning. Comput Aided Geom Design 2016;43:39–52.
[30] Benhabiles H, Lavoué G, Vandeborre J-P, Daoudi M. Learning boundary edges for 3D-mesh segmentation. Comput Graph Forum 2011;30(8):2170–82.
[31] Xie Z, Xu K, Liu L, Xiong Y. 3D shape segmentation and labeling via extreme learning machine. Comput Graph Forum 2014;33(5):85–95.
[32] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3D classification and segmentation. 2016, arxiv preprint.

[33] Kalogerakis E, Averkiou M, Maji S, Chaudhuri S. 3D shape segmentation with projective convolutional networks. In: Proc. IEEE conf. CVPR, 2017.
[34] Yi L, Guibas L, Hertzmann A, Kim VG, Su H, Yumer E. Learning hierarchical shape segmentation and labeling from online repositories. In: Proc. ACM SIGGRAPH, 2017.
[35] Mo K, Zhu S, Chang AX, Yi L, Tripathi S, Guibas LJ, et al. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In: Proc. IEEE Conf. CVPR, 2019.
[36] Yu F, Liu K, Zhang Y, Zhu C, Xu K. PartNet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In: Proc. IEEE conf. CVPR, 2019.
[37] Wang X, Zhou B, Fang H, Chen X, Zhao Q, Xu K. Learning to group and label fine-grained shape components. ACM Trans Graph 2018;37(6).
[38] Vijayanarasimhan S, Grauman K. Multi-level active prediction of useful image annotations for recognition. In: Neural info. process. sys. 2009, p. 1705–12.
[39] Branson S, Perona P, Belongie S. Strong supervision from weak annotation: Interactive training of deformable part models. In: Proc. int. conf. on comput. vis., 2011, pp. 1832–9.
[40] Vezhnevets A, Buhmann JM, Ferrari V. Active learning for semantic segmentation with expected change. In: Proc. IEEE conf. CVPR, 2012, p. 3162–9.
[41] Branson S, Van Horn G, Wah C, Perona P, Belongie S. The ignorant led by the blind: A hybrid human–machine vision system for fine-grained categorization. Int J Comput Vis 2014;108(1–2):3–29.
[42] Wu Z, Shou R, Wang Y, Liu X. Interactive shape co-segmentation via label propagation. Comput Graph 2014;38:248–54.
[43] Page DL, Koschan AF, Sukumar SR, Roui-Abidi B, Abidi MA. Shape analysis algorithm based on information theory. In: Proc. int. cont. on image process, 2003, p. 229-32.
[44] Xing LP, Hui KC. Entropy-based mesh simplification. Comput Aided Des and Appl 2010;7(6):911–8.
[45] Lee D-Y, Sull S, Kim C-S. Progressive 3D mesh compression using MOG-based Bayesian entropy coding and gradual prediction. Vis Comput 2014;30(10):1077–91.
[46] Limper M, Kuijper A, Fellner DW. Mesh saliency analysis via local curvature entropy. In: Proc. conf. of the Euro. assoc. for comput. graphics, 2016, p. 13–6.
[47] Johnson AE, Hebert M. Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Trans Pattern Anal Mach Intell 1999;21(5):433–49.
[48] Chen D-Y, Tian X-P, Shen Y-T, Ouhyoung M. On visual similarity based 3D model retrieval. Comput Graph Forum 2003;22(3):223–32.
[49] Lai Y-K, Hu S-M, Martin RR, Rosin PL. Rapid and effective segmentation of 3D models using random walks. Comput Aided Geom Design 2009;26(6):665–79.
[50] Boykov Y, Veksler O, Zabih R. Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Mach Intell 2001;23(11):1222–39.
[51] Xu B, Wang N, Chen T, Li M. Empirical evaluation of rectified activations in convolutional network. In: ICML deep learning workshop, 2015.
[52] Tieleman T, Hinton G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. 2012, COURSERA: Neural Networks for Machine Learning.
[53] Loshchilov I, Hutter F. SGDR: stochastic gradient descent with restarts. 2016, arXiv preprint arXiv:1608.03983.
[54] Tam GKL, Kothari V, Chen M. An analysis of machine- and human-analytics in classification. IEEE Trans Vis Comput Graph 2017;23(1):71–80.
[55] Huang J, Su H, Guibas L. Robust watertight manifold surface generation method for ShapeNet models. 2018, arXiv preprint arXiv:1802.01698.
[56] Huang J, Zhou Y, Guibas L. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. 2020, arXiv preprint arXiv:2005.11621.
[57] Hu Y, Schneider T, Wang B, Zorin D, Panozzo D. Fast tetrahedral meshing in the wild. ACM Trans Graph 2020;39(4).