SUPERVISED MESH SEGMENTATION FOR 3D OBJECTS WITH GRAPH CONVOLUTIONAL NEURAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMİR KAAN PEREK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2019

Approval of the thesis:

SUPERVISED MESH SEGMENTATION FOR 3D OBJECTS WITH GRAPH CONVOLUTIONAL NEURAL NETWORKS

submitted by **EMİR KAAN PEREK** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Yusuf Sahillioğlu
Supervisor, **Computer Engineering, METU** _____

Assoc. Prof. Dr. Sinan Kalkan
Co-supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Pınar Duygulu Şahin
Computer Engineering, Hacettepe University _____

Assoc. Prof. Dr. Yusuf Sahillioğlu
Computer Engineering, METU _____

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Emir Kaan Perek

Signature :

ABSTRACT

SUPERVISED MESH SEGMENTATION FOR 3D OBJECTS WITH GRAPH CONVOLUTIONAL NEURAL NETWORKS

Perek, Emir Kaan
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Yusuf Sahillioğlu
Co-Supervisor : Assoc. Prof. Dr. Sinan Kalkan

August 2019, 54 pages

Mesh segmentation is a fundamental application that is primarily used for understanding and analyzing 3D shapes in a broad range of areas in Computer Science. With the increasing trend of deep learning, there have been many learning-based solutions to the mesh segmentation problem based on the classification of the individual mesh polygons.

In this thesis, we cast mesh segmentation as a supervised graph labeling problem by using Graph Convolutional Neural Networks (GCNN). We treat a mesh object as a graph to be labeled and develop a segmentation model that takes an entire structure of the object as input and returns its segmentation. While similar models focus on labeling each polygon of the 3D objects separately, our model is capable of labeling all polygons in a single run thanks to GCNN. Moreover, being able to use connectivity information in the graph provides an opportunity for a drastic decrease in the required features used as input to the model, compared to the previous studies. We train and test our model for segmenting human shapes, one of the challenging shapes to segment. We report competitive results compared to other state-of-art supervised

segmentation techniques by using noticeably less input features.

Keywords: mesh segmentation, deep learning, graph convolutional neural networks.

ÖZ

GRAFİKSEL EVRİŞİMLİ SİNİR AĞLARI KULLANARAK DENETİMLİ 3 BOYUTLU NESNE BÖLÜTLEMESİ

Perek, Emir Kaan
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Yusuf Sahillioğlu
Ortak Tez Yöneticisi : Doç. Dr. Sinan Kalkan

Ağustos 2019 , 54 sayfa

Nesne bölütlemesi 3B şekillerin anlamlandırılması ve analizi başta olmak üzere Bilgisayar Bilimlerinin birçok farklı alanında kullanılan temel bir uygulamadır. Derin öğrenme uygulamalarının yaygınlaşmasıyla nesne bölütlemesi için tekil çokgenlerin sınıflandırılmasına dayanan çözümler halihazırda üretilmiştir.

Bu tezde biz, nesne bölütlemesi problemini Grafiksel Evrişimli Sinir Ağlarını kullanarak bir denetimli grafik etiketleme problemi olarak ele alıyoruz. Bir objeye etiketlenmesi gereken bir grafik yapısı olarak bakıp, bu objenin tüm yapısını girdi olarak alan ve bölütlenmiş halini çıktı olarak veren bir bölütleme modeli geliştiriyoruz. Benzer modeller 3B obje üzerindeki her bir çokgeni ayrı ayrı etiketleme üzerine kuruluyken, bizim modelimiz objenin tüm çokgenleri için tek bir turda etiketleme yeteneğini Grafiksel Evrişimli Sinir Ağları sayesinde edinebilmektedir. Ayrıca, grafik yapısındaki bileşenlerin bağlantısallığını kullanabilmek önceki çalışmalara nazaran girdi modelleri için gereken özellik sayısını ciddi derecede azaltabilmektedir. Biz modelimizi segmentasyon işlemi zorlu şekil gruplarından biri olan insan şekilleri üzerinde eğitip test ettik ve diğer denetimli bölütleme teknikleriyle rekabet edebilecek düzey-

deki sonuçları onlardan çok daha az girdi özelliği kullanarak elde edebildik.

Anahtar Kelimeler: nesne bölütlemesi, derin öğrenme, grafiksel evrişimli sinir ağları

To my family

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my thesis advisors Assoc. Prof. Dr. Yusuf Sahillioğlu and Assoc. Prof. Dr. Sinan Kalkan for their advice, guidance, and continuous support.

I would like to thank the family of Ankawiser; Erkut Alakuş, Onur Tosun, Önder İlke Sever, and Mehmet Gökhan Aygün who are my friends, my colleagues, and also my brothers for their backing and toleration to me during this period.

I would like to thank my trip-team; Baran Kemer, Buğra Tan, and Önder İlke Sever for their motivational events and acts.

Last but not least, I would like to thank my lovely family for their endless support and confidence to me.

Finally, this thesis work is supported by the Scientic and Technological Research Council of Turkey (TÜBİTAK) under the project EEEAG-115E471.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
DNN	Deep Neural Network
CNN	Convolutional Neural Network
GCNN	Graph Convolutional Neural Network
GCL	Graph Convolutional Layer

CHAPTER 1

INTRODUCTION

Object segmentation is the process of decomposing virtual objects into meaningful and smaller pieces. In this thesis, objects are assumed to be 3D surfaces discretized as mesh structures comprised of vertices, edges, and polygonal faces. Hence, we deal with the mesh segmentation problem. Furthermore, we restrict our attention to the segmentation of humans, a popular object type that appears quite frequently in computer graphics applications.

Mesh segmentation is a crucial and fundamental problem, which has been studied extensively, since it is critical for understanding and processing 3D objects for further applications such as animation, texture applying, modeling, and skeleton extraction, to name a few.

1.1 Motivation and Problem Definition

Unlike a regular data segmentation, mesh segmentation should be handled as a different case because of its structure, which includes a network inside. It is also required to be visually consistent and semantically correct.

There are fundamental solutions to the segmentation problem such as watershed segmentation [12] and statistical modeling based methods like Conditional Random Field [13, 7]. Most of the existing solutions, however, suffer from not being able to inject the inherent graph structure into the process. They overcome this problem by increasing the number of features used in segmentation process which results in high dimensional and complex data models.

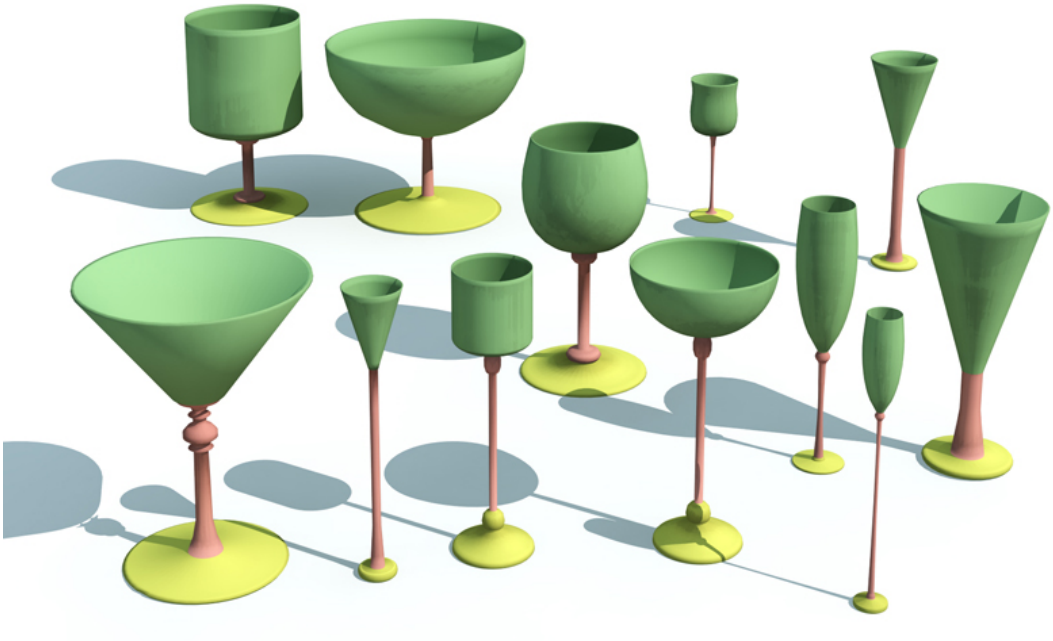


Figure 1.1: Visualization of a mesh segmentation result (Figure Source: [1]).

There are also prominent techniques that are quite commonly used for mesh segmentation. Shape Diameter Function [14] is one of the most popular one among them; however, it does not use the graph structure of the mesh, so the results of this algorithm suffer from the adjacency related problems which results in a fragmentary segmentation. Region Growing [15] and Random Walking [16] algorithms are other solutions to the problem but they require a logical set of seed points to start and the selection of these seed points brings another problem to the table which needs to be solved separately. We avoid seeding and explicitly consider the graph structure.

1.2 Proposed Methods and Contributions

In this thesis, we propose a data-driven mesh segmentation technique which uses in its core the Graph Convolutional Neural Network, the specialized version of the Deep Neural Networks for graph-structured data. Our method presents a generic approach for mesh segmentation and represents a solution independent of the polygon counts of the input meshes. Our method has a training phase in order to create a capable

network model and after the training phase, segmentation could be done straightforwardly and in real-time by using the capabilities of our trained model without the need of any input parameter like a seed point, segment count, and so on.

Our contributions in this thesis are as follows:

- In contrast to the most of the mesh segmentation techniques, we propose a method that requires just a few core features of the mesh objects and extends them using the additional feature extraction power of Graph Convolutional Neural Networks. Our decision making system also takes into account the neighborhood information which brings more accuracy to the results.
- Unlike all the segmentation models based on learning, our model is the first one taking the entire mesh object as an input instead of a single polygon of the mesh object. In other words, we feed the input mesh to our network in its entirety and only once, whereas the existing learning-based methods feed each mesh polygon to the network. This provides us an object-based learning, instead of a polygon-based learning.
- We show that it is possible to create a model that learns the pose of an object and this model can transfer the information of the pose to the newly encountered objects, which in turn yields a pose-invariant segmentation solution.

1.3 The Outline of the Thesis

The remaining of this thesis is organized as follows:

In Section 2; mesh segmentation techniques and their approaches are examined. Also, background information about Deep Neural Networks and Graph Convolutional Neural Networks are given and their usages are explained.

In Section 3; our proposed method for mesh segmentation is presented in detail. Architectural structure of the segmentation model and used loss functions to accomplish learning process are described.

In Section 4; experimental results of our segmentation method are given quantita-

tively through figures. In particular, we compared our method with other methods and we also evaluated our segmentation performance with the Princeton Segmentation Evaluation Benchmark.

In Section 5; conclusion of the method is given. Also, future work of our method have been discussed.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Mesh Segmentation

Mesh segmentation has always been an active research area in computer graphics, because it is one of the key parts of shape analysis. There is a wide variety of methods that have been proposed to accomplish this task. All of the methods try to solve an optimization problem with different approaches. There are several surveys [17, 18, 19, 2] on mesh segmentation techniques. They categorize these methods according to their solution approach, discuss their performance and compare with each other in detail. We will use the categorization of Rodrigues et al. [2] to give brief information about the related work (Figure 2.1).

2.1.1 Volume-based segmentation

This type of algorithms gives 3D volumes as their output. They are sub-categorized as Exact Convex Decomposition, Approximate Convex Decomposition, Volumetric Meshes, and Space Partitioning.

Exact Convex Decomposition In this case, 3D mesh objects are decomposed into exact convex sub-volumes. Chazelle [20] has proposed an algorithm for partitioning a polygon into a number of smaller convex sub-volumes in 1984 and this study has been an origin point in this field. On the following studies [21, 22], Chazelle's work has been improved but this family of algorithms did not take an essential role in mesh segmentation techniques due to the performance issues and lack of ability to consistently give meaningful results.

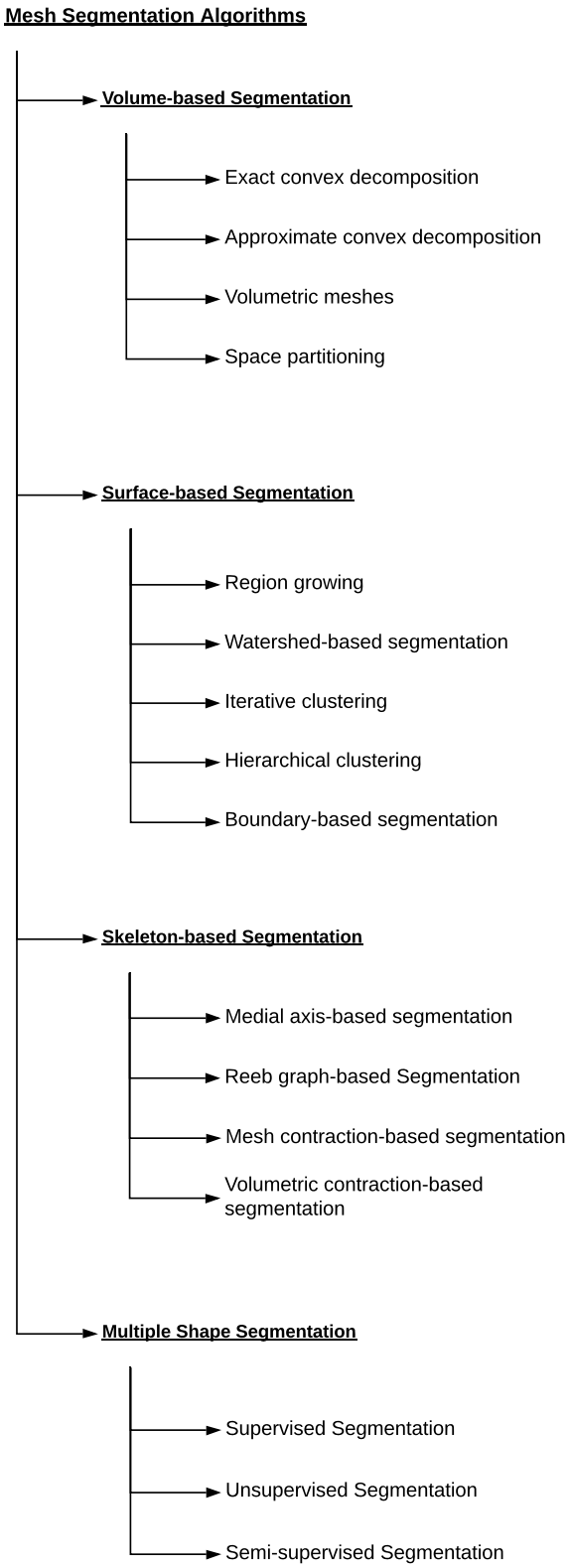


Figure 2.1: Categorization of mesh segmentation algorithms in [2].

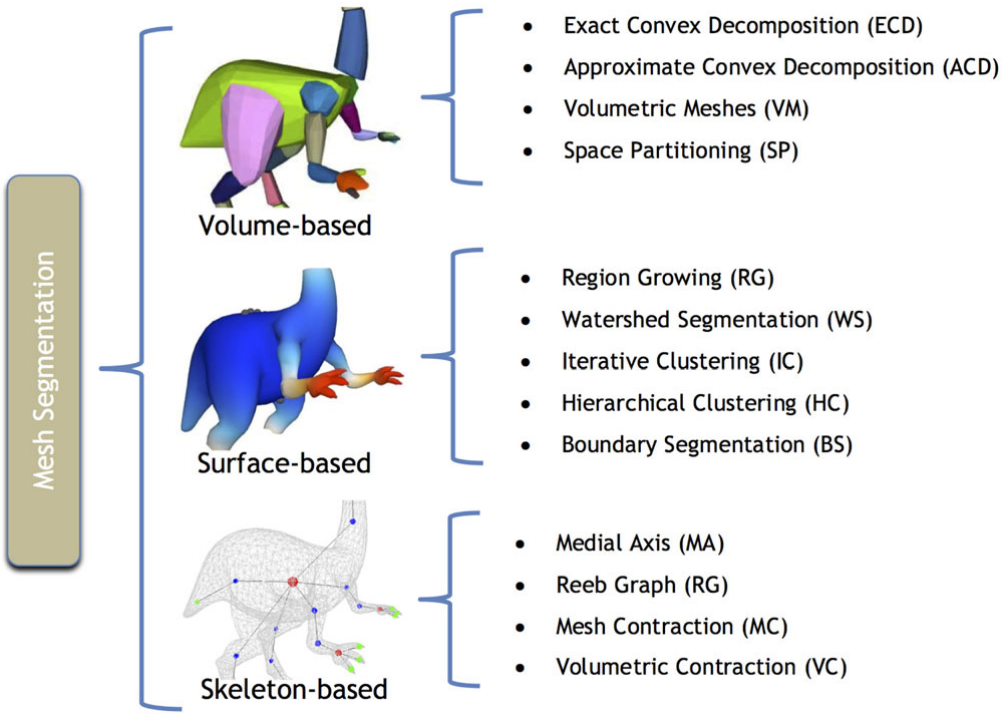


Figure 2.2: Taxonomy of algorithms according to their solution approaches (Figure Source: [2]).

Approximate Convex Decomposition In this case, decomposition results are the approximate volumetric convex pieces instead of the exact convex hulls. Lien and Amato's work [23] is known as the first study in this group. Their work is built upon the measure of concavity of 3D objects. Later on, Kreavoy et al. [24] have developed on a method which is based on the convexity instead of the concavity. Liu et al. [25] have proposed a new method based on Morse theory and stated convex decomposition problem as an integer linear programming problem.

Volumetric Meshes In this case, decomposition results are the set of primitive volumetric elements such as voxels, tetrahedra, hexahedra, and so forth. Attene et al. [3] have proposed an algorithm for this purpose by recursively enclosing the object with convex polyhedra. Xian et al. [26] have come up with a three step solution to this problem by combining surface mesh segmentation techniques with the graph cut algorithm.

Space Partitioning In this last case, decomposition is done by subdividing the space

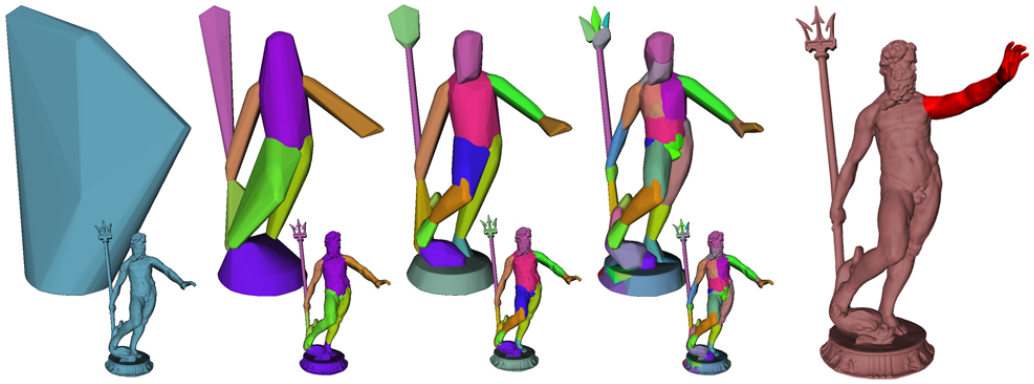


Figure 2.3: Recursive process of polyhedral enclosing presented in [3] (Figure Source: [3]).

into regions using the properties of the mesh objects. Simari et al. [27] have introduced a multi-objective optimization problem for space partitioning using k-means clustering and Voronoi partitioning in its core.

2.1.2 Surface-based segmentation

This type of algorithms gives 2D regions as their output. In this family of algorithms, surface of the 3D objects is used for producing the segmentation results. Consequently, the features of the surfaces like dihedral angle, curvature, geodesic distance and the constraints are the principal concern in these algorithms. Similar to [2], we classify surface-based segmentation algorithms into 5 different sub-categories as such Region Growing, Watershed Segmentation, Iterative Clustering, Hierarchical Clustering and Boundary Segmentation.

Watershed Segmentation Mangan and Whitaker [12] have described a method to apply watershed image segmentation into 3D surfaces. They have created patches in surfaces according to the curvatures of the surfaces.

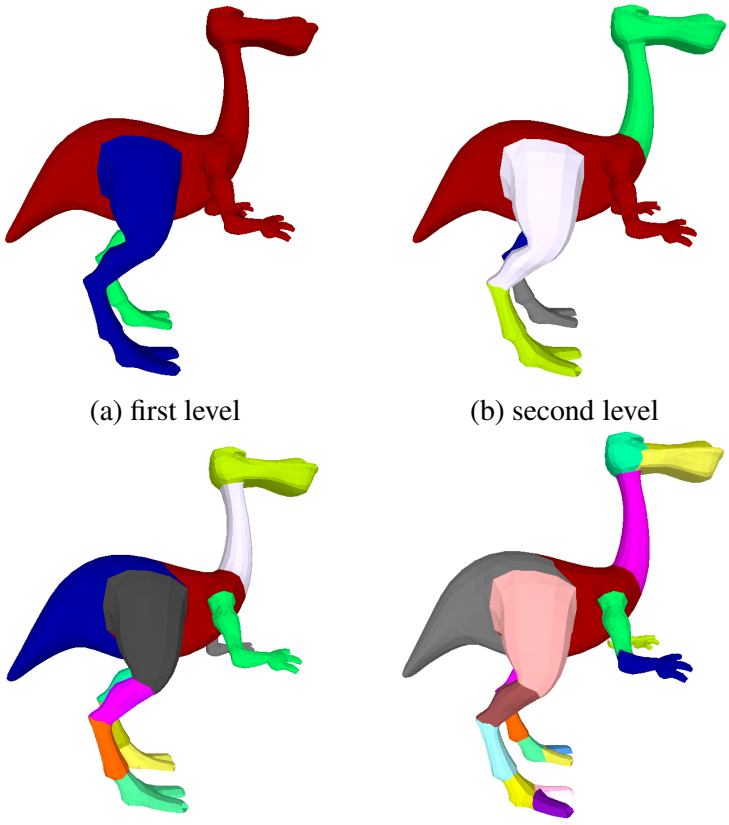
Iterative Clustering Shlafman et al. [28] have used convexity, curvature and proximity for the surface decomposition task. Liu and Zhang [29] have used spectral clustering for the first time in the mesh segmentation problem by defining affinity matrices with the help of geometric properties of the regions. Simari et al. [30] have employed

symmetry detection and a new representation of the meshes, called folding tree, for the segmentation. Their approach was quite novel but was not sufficiently successful on non-symmetric mesh inputs. Huang et al. [31] have proposed a novel algorithm to decompose a deformable mesh object into semantically correct parts using modal analysis and requiring just a single pose of the object. Algorithm was, however, dependent on the definition of the deformation energy function and the attributes of the modal analysis.

Hierarchical Clustering Katz and Tal [4] have geodesic distances and convexity in their study to decompose object meshes into segmented parts. Attene et al. [32] have described a hierarchical method for face clustering relying on fitting primitives belonging to an arbitrary set. Their method was basically defining each triangle as a different cluster and merging adjacent clusters by their geometric closeness. Lai et al. [16] have proposed a random walk algorithm for mesh segmentation which computes the probability of belonging to each selected seed point for all the triangles over the mesh. This approach was quite novel but brought other problems to solve like how to select efficient seed points and how to describe closeness of triangles. Shapira et al. [14] have created a new attribute named 'shape diameter' and proposed a novel segmentation and skeleton extraction technique depending on it.

Region Growing Katz et al. [15] have used feature point and core extraction to create mesh segmentation in region growing manner and proposed a novel algorithm for pose-invariant segmentation. Later on, researchers have started to create statistical models for mesh segmentation and tried to optimize these models as a solution. To this end, Lavoué and Wolf [13] have used an algorithm which is based Markov Random Fields, graphical probabilistic models, in order to achieve a global optimal solution using only local interactions of the mesh structure thanks to the Markov property of the random fields by combining both the geometry and the attributes into this model.

Boundary Segmentation Golovinskiy and Funkhouser [5] have used random cuts over the meshes by creating a partitioning function that tries to compute how likely each edge is to lie on these random cuts and aimed to find the most consistent cuts according to this function. Lin et al. [33] have used the concept called 'part salience',



(a) first level (b) second level

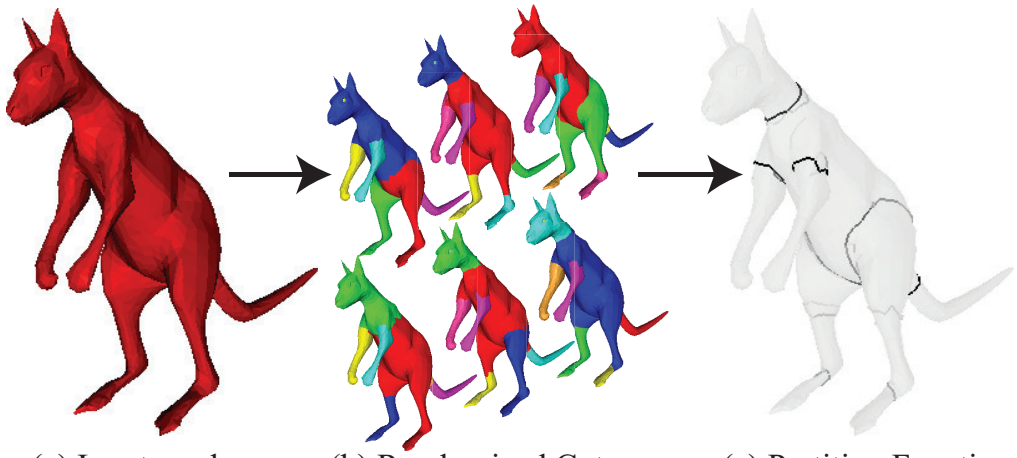
(c)third level (d) fourth level

Figure 2.4: Hierarchical clustering using k-means and graph cut introduced in [4] (Figure Source: [4]).

which was initiated in cognitive psychology, that depends on three factors: the protrusion, the boundary strength, and the relative size of the part. They have created the computational approximation of this concept for mesh segmentation purpose.

2.1.3 Skeleton-based segmentation

This type of algorithms give 1D skeleton as their output which corresponds to structural shape of the given 3D mesh input. Skeleton-extraction has an important role in research fields and applications like animation, virtual navigation and segmentation. There are approaches to extract skeleton from mesh objects using medial axis, reeb



(a) Input mesh (b) Randomized Cuts (c) Partition Function

Figure 2.5: Boundary-based segmentation used in [5] (Figure Source: [5]).

graph and geometric contraction.

Medial axis In this case, main purpose is to find the sufficient medial axis points of the mesh to create skeleton built upon these points. Several methods have been proposed for the approximation of skeleton using medial axis. One of them is Siddiqi et al. [34] which generates the approximation of medial axis using a 3D voxelization technique. Another algorithm is proposed by Mortara et al. [35] which is based on extraction of model structure for human-shaped bodies.

Reeb graph Reeb graph is a 1D graph representation whose node points are the critical points of the subject. Aleotti and Caselli [6] have introduced an algorithm for Reeb graph-based segmentation for robot grasping.

Geometric contraction In this case, 3D object is continuously deflated until the result becomes a skeleton. Au et al. [36] have proposed a novel and successful contraction technique for the purpose of skeleton extraction. Lovato et al. [37] have developed another approach from voxel coding and active contours.

Skeleton is a 1D structure that captures the topology and geometry of an object in a simplified manner. An extracted skeleton that reflects its topology accurately provides valuable information in order to create segmented parts of an object. Reniers and Telea [38] have proposed shape segmentation technique using the skeleton of an

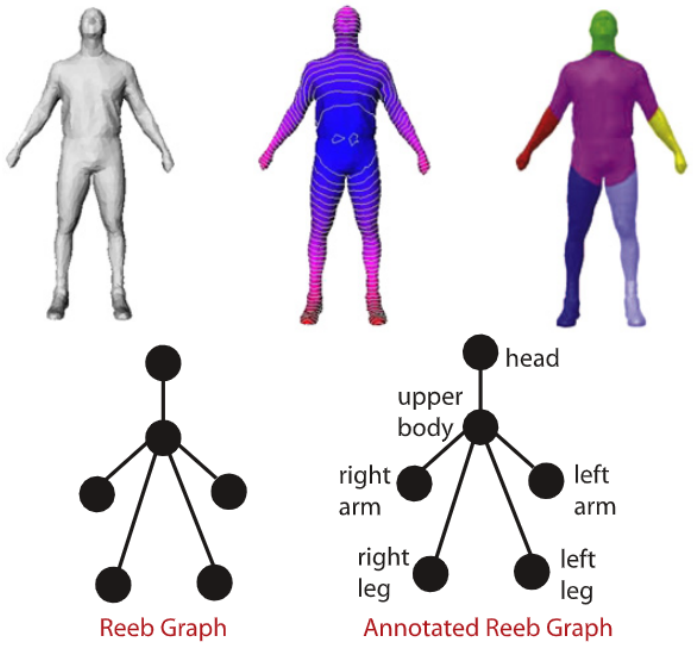


Figure 2.6: Reeb graph based segmentation used in [6]. From left to right: original mesh, the level-sets of the integral geodesic function, generated reeb graph, annotated reeb graph, segmentation result (Figure Source: [6]).

object. Li et al. [39] have also introduced a skeleton-based approach for the process of mesh decomposing.

2.1.4 Multiple shape segmentation

A shape in isolation may not possess sufficient information to carry out a perfect segmentation process. With the increasing availability of 3D models, researchers have started using shape collections to improve their results in various geometry processing tasks. Shape segmentation task is also affected by this trend which involves the query shape as well as a collection of annotated or unlabeled related shapes that are primarily used to train a model. Our proposed method in this thesis also lands in this family of algorithms. These methods can also be subdivided as supervised, unsupervised and semi-supervised according to their method of extracting the required segmentation aspects.

Supervised Segmentation solutions are also affected with the rising trend of machine

learning. Kalogerakis et al. [7] have formulated a conditional random field model for the purpose of the segmentation and optimized it with the JointBoost classifier [40]. This study has been regarded as a state-of-art work and brought a new perspective to the mesh segmentation research. However, their model has required about six hundred features of the 3D model for the optimization task.

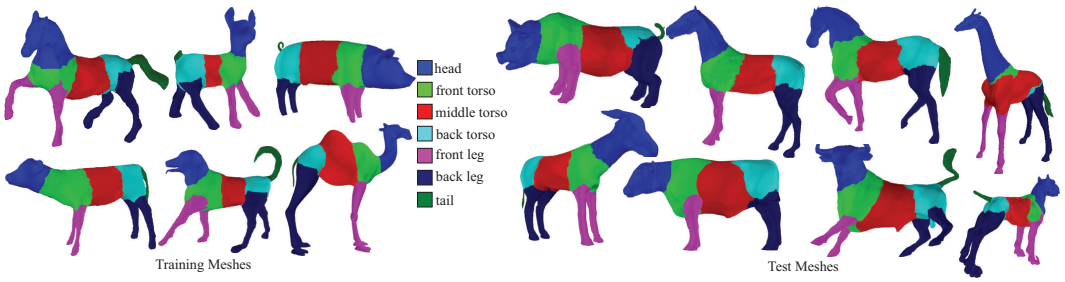


Figure 2.7: Result of the supervised mesh segmentation algorithm introduced in Kalogerakis et al. [7] (Figure Source: [7]).

Xie et al. [41] used a quite similar technique with [7] except they used extreme learning machine [42, 43] architecture instead of the JointBoost classifier in [40]. Using extreme learning machine has provided observable improvement in the training performance. Nonetheless, the performance of the segmentation has not improved. Makadia and Yumer [44] have proposed an algorithm that utilizes a structured SVM model [45] by using sparsely labeled objects as training data.

In 2015, Guo et al. [46] have introduced a novel and rather successful technique by adapting CNN to the supervised mesh labeling problem. They managed to get remarkable results on the accuracy of the labeling process. Later on, George et al. [47] and Wang et al. [48] also proposed CNN based solutions for the supervised mesh segmentation problem. However, all these CNN based solutions have suffered from converting an irregular form of graph structure into a regular grid structure, and they developed their approach for this conversion process. Also, they used hundreds of shape features for each face of a 3D object as the input of their network. We also use a supervised approach in our method, so we fall in this sub-category of the mesh segmentation algorithms.

Projective approaches are also used for the segmentation task. Wang et al. [49] have supplied one of the methods that handles the segmentation process by defining 3D

mesh object as a collection of 2D image and transferring 2D image segmentation into 3D mesh object without using any machine learning techniques. Similarly, Kalogerakis et al. [8] have used convolutional neural networks for the projective segmentation. They have transferred the mesh structure into 2D projective images and they run CNNs on these images. Finally, they have transferred the results into the 3D mesh structure.

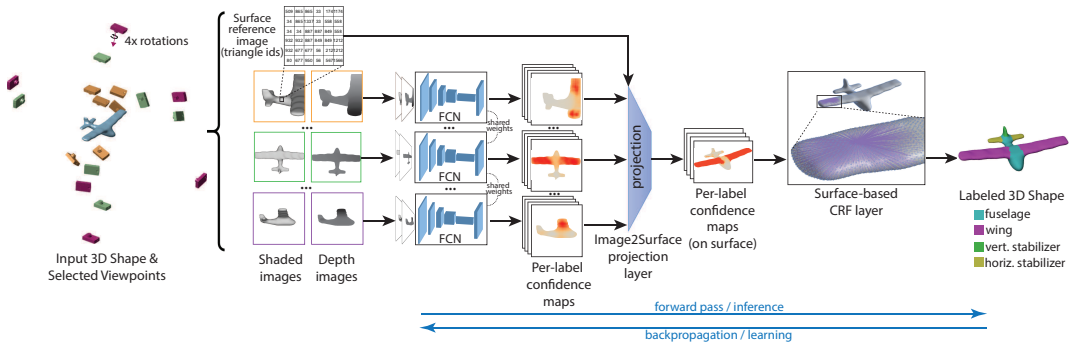


Figure 2.8: Representation of the pipeline and architecture proposed in [8] (Figure Source: [8]).

Semi-supervised In the algorithm proposed by Lv et al. [50], additional energy term of unlabeled data is added to the conditional random fields, which is responsible for labeling. Also, they have modified the virtual evidence boosting (VEB) [51] to succeed the optimization of semi-supervised learning problem. This semi-supervised study have taken the advantage of the unlabeled data which is useless for supervised methods. Shu et al. [52] have proposed a scribble based segmentation with weakly-supervised learning and acquired comparable segmentation results with other state-of-art techniques.

Unsupervised Kreavoy et al. [53] have created an interchangeable component modeling for the objects which relies on the idea that the same classes might share the same segmentation model. Sidi et al. [54] have performed an analysis in the descriptor space and using the analysis they have created a spectral clustering for the segmentation purpose. Their approach used shape descriptors which were independent of the pose and the location of an object.

2.2 Deep Neural Networks (DNN)

The term 'Deep Neural Network' (DNN), which is also referred to as basis of 'deep learning', is derived from the study on artificial neural networks [55]. DNNs inspired by the working mechanism of the brain. DNNs have been applied to great variety of areas such as natural language processing, linguistics, pattern recognition, recommendation systems, classification models, and so forth. DNNs are able to learn high-level features with more complexity than standard neural networks. Accordingly, the usage of DNNs for complicated tasks result in quite successful and impressive outcomes.

2.2.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are a common type of the DNNs. CNNs composed of multiple convolution layers and each of the layers provide higher level of abstraction and crucial information to the consecutive layers. Hubel and Wiesel [56] introduced the concept of receptive fields in 1962 and this research inspired to the architecture of CNNs. CNNs are widely used in areas such as image and video recognition [57, 58, 59, 60], speech recognition [61, 62, 63, 64], robotics [65, 66], automatic game plays [67, 68, 69], and so forth.

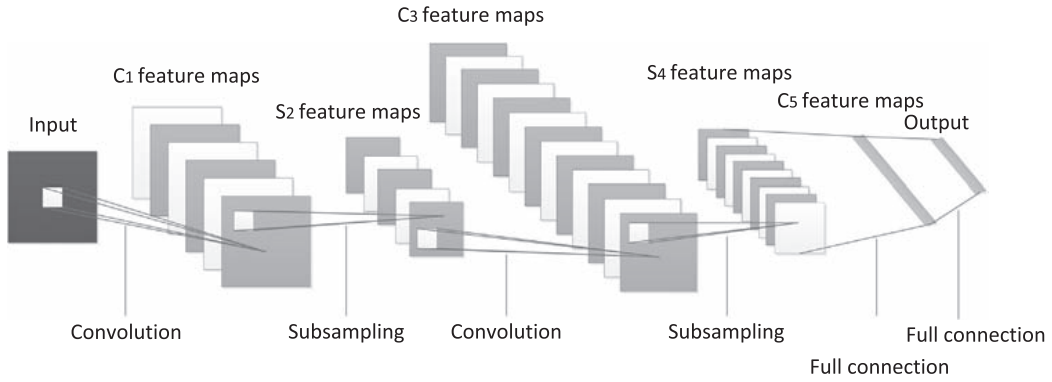


Figure 2.9: Schematic architecture of CNNs (Figure Source: [9]).

2.2.2 Graph Convolutional Neural Networks (GCNN)

Applying deep learning on graph-structured data is a complicated task, because of the complex and non-grid structure of it. Graph Convolutional Neural Networks (GCNN) are the generalization of convolutional neural networks from Euclidean domain to graph-structured data. Because the structure of the graph data is not like a regular grid, it is not possible to apply classical convolutional filters on graphs.

There exists specialized architectures of CNNs to handle specific cases with graph-structured data. In the study of Duvenaud et al. [70], CNN architecture that operates directly on graphs by using standard molecular feature extraction methods based on circular fingerprints is introduced. Li et al. [71] have proposed a solution to sequence based models like Long short-term memory networks (LSTM) using graph-structured data. Since all these methods are developed for the specific cases, they are not presenting a generic solution to the broad range of problem types. There are two methods to build generic convolutional filters; spatial construction and spectral construction.

Spatial construction is basically provides localization of convolutional filters through fixed size of kernel and filtering is completed by sliding those filters. However, sliding filters on a graph-structured data is not a straightforward application as 2D image data or 1D audio data. Therefore, this approach has few challenges inside. First challenge is defining receptive fields and neighbourhoods on a non-uniform arbitrary graph data, since there does not exist any mathematical definition to efficiently convert graph data as pointed in [72]. Second challenge is the ordering of the nodes in the graph. For 2D images and 1D audio, it is quite obvious to create the order of input data in order to consume. Nonetheless, for a graph structure, there might be created many different ordering of the nodes. Niepert et al. [73] and Vialatte et al. [74] proposed different methods by using spatial approach to generalize CNNs on irregular graph structure.

Spectral construction is the representation of the parametrized convolutional filters as Laplacian operator in spectral domain by using Graph Fourier Transform. Bruna et al. [72] and Henaff et al. [75] have proposed the methods for applying spectral filtering in the CNNs. However, using spectral filtering had drawbacks in two main

topic; localization of the filters and high cost of computation. Defferrard et al. [10] have introduced a technique to overcome these drawbacks and their technique turned into state-of-art method in the Graph Convolutional Networks. They approximated smooth filters in spectral domain using Chebyshev polynomials. Their architecture was tested on regular domains like MNIST and gave nearly same performance with basic CNNs.

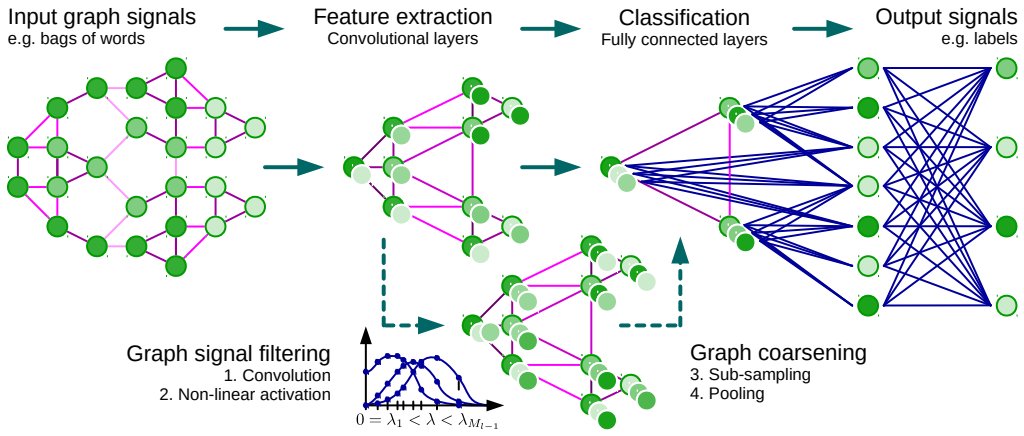


Figure 2.10: Schematic architecture of GCN proposed in [10] (Figure Source: [10]).

Later on, Kipf and Welling [76] have taken the performance of the GCN one step further. They simplified the convolutional filtering operation with generalized and differentiable form of Weisfeiler-Lehman algorithm [77] and proposed a semi-supervised learning mechanism for graph-structured data. Their method is currently state-of-art method in this field and has wide area of usage. In our network model, we also use the benefits of this approach.

CHAPTER 3

SEGMENTATION MODEL

In this chapter, we explain the model architecture and the steps of our segmentation process. Firstly, we will show the preparation process of the mesh objects in order to use them in our GCNN structure. Next, architecture of the network and training of the network will be explained in detail. Finally, application of the result received from network to the original mesh object, post-processing, will be demonstrated.

3.1 Ground-Truth Labeled Training Data

First of all, we need to have a labeled set of 3D objects since our method uses supervised approach. These labeled objects will be split to use as training, validation and test sets in order to train our segmentation network. Since, our segmentation model takes whole 3D model as input (related works take faces of 3D model as input), it is a challenging process to create an effective dataset for our purpose. Variety of data augmentation techniques can also be used to enrich dataset. For instance, we also created 90° rotated versions of each individual model to train our model rotation independently.

3.2 Pre-Process Stage

In order to use convolutional neural networks, we need to have a specified data structure. Size of the input features and the output features must be determined beforehand. In our neural network, 3D mesh objects will be our input models. In real world, 3D mesh objects could have any size of triangles. The question is how can we standardize

Figure 3.1: Example of ground-truth labeled data set.

these objects so we can use all of them in our network. For images, there are highly efficient and simple upsampling and downsampling methods to change image resolution. However, it is not that easy to apply upsampling or downsampling for graphs. At this point, geometry processing comes to the assistance. Our graph structure represents 3D mesh objects and there are several adequate 3D object upsampling and downsampling methods, which are called subdivision and simplification in geometry processing literature. Before converting our 3D objects into graph structure, we can apply simplification or subdivision methods to extract uniform graph structures.

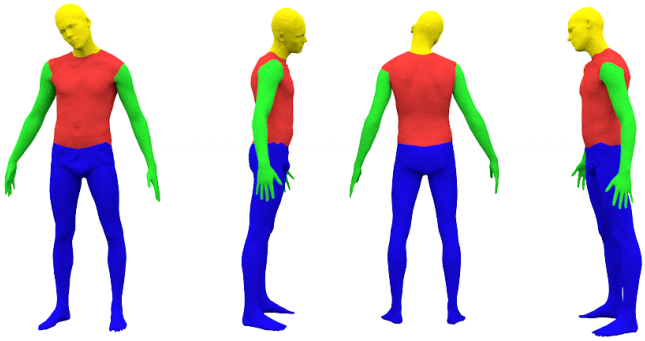


Figure 3.2: Example of data augmentation with 90°, 180° and 270° rotation.

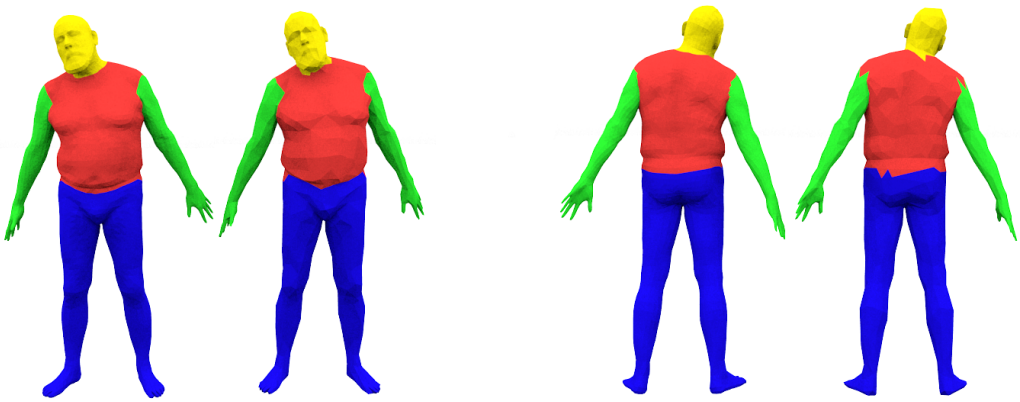


Figure 3.3: Visual result of data pre-processing. Original mesh (13776 faces, on the left side of poses) and simplified mesh (2000 faces, on the right side of poses). Labeling information and details are kept in a good way.

3.3 Architecture of the Model

3.3.1 Input Features and Output

We have used 2 most basic characteristics of polygons as input features. They are center position of the polygon and unit normal vector of the polygon. Each feature includes 3 numeric values which are X, Y and Z coordinates, which results total of 6 input features in our situation. Output of the network will be the final scores of each segmentation tag in our segmentation model. For instance, if we have 4 segmentation tag like head, torso, arms and legs, our network will produce 4 different scores of each polygon belongs to input model. Applying Softmax function to these scores will convert nominal scores to probability distribution of each segmentation tag.

3.3.2 Loss Functions

Our goal in this model is taking the feature vector of model which has size NxF, where N is the polygon count of pre-processed 3D model and F is the size of input features which is 6 in our situation, and getting the probability distribution vector P whose size is NxL, where L is the size of predefined set of possible labels, such as 'head', 'torso', 'arms' and 'legs'. By taking the index of maximum values in

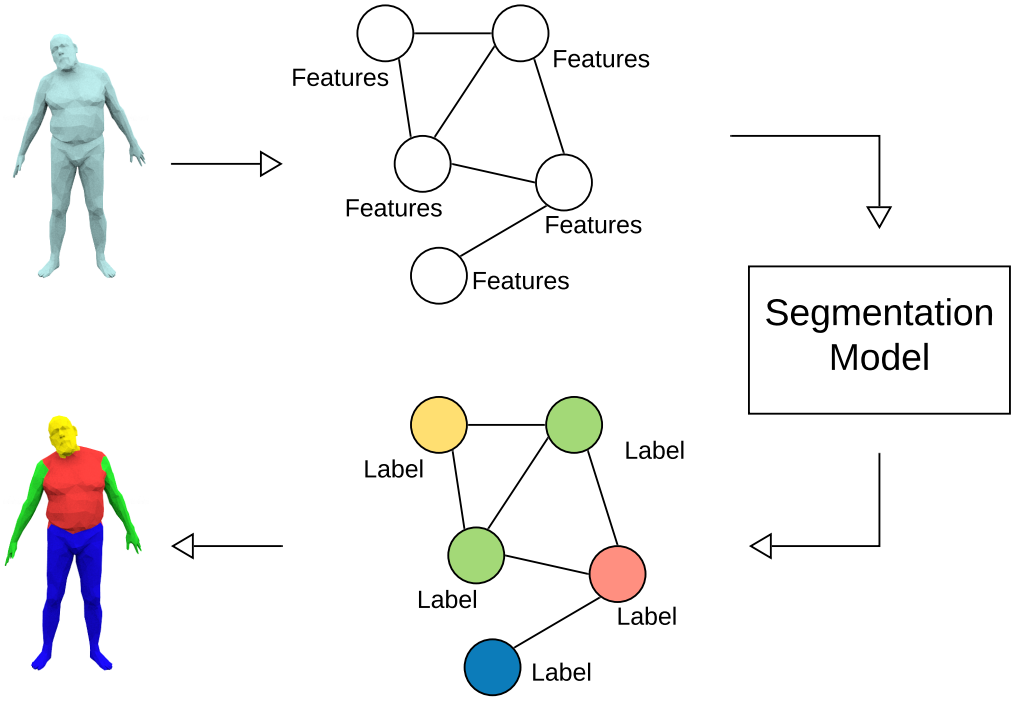


Figure 3.4: Pipeline for the segmentation process of a single object.

probability distribution vector for each row, we will get indicated labels for each face in our input 3D object.

There are 2 loss functions used in the training process of our segmentation model.

First one evaluates the performance of classifier:

$$L_1 = \left(-\sum_{p_i}^{N} \log(p_i) \right) / N, \tag{31}$$

where p_i is the predicted probability for correct label. This function can also be expressed as the average of Negative Likelihood Loss for each face in the object.

Second one evaluates the smoothness of classifier:

$$L_2 = \left(\sum_{i=1}^{N} \sum_{a_j}^{3} \sum_{k=1}^{L} |p_{i_k} - p_{j_k}| \right) / N, \tag{32}$$

where p_{i_k} is the predicted probability for k^{th} label of i^{th} face. This function basically penalizes the differences between predictions of the neighbourhood faces. In mesh segmentation problem, adjacent faces are most likely belong to the same class except

the faces on the border edges. If our network makes the same prediction for every face in the model, this function achieves the optimal point. The main purpose behind the necessity in this formula is prevent the fluctuation between adjacent faces, which leads to less noise in label prediction.

Overall loss function to use in our network is:

$$L_{overall} = \alpha L_1 + \beta L_2, \tag{33}$$

where α and β denotes scalar parameters which will be tuned as a hyper-parameter for training phase. If α value is higher, network will attach importance to accuracy more than smoothness and it is quite possible to observe noises on segmentation results. Otherwise, if β value dominates $L_{overall}$, network will care about smoothness more than accuracy, which will result in losing edges of boundaries on mesh object. Therefore, it is crucial to find balance between α and β in order to have efficient segmentation results.

3.3.3 Network Layers

There are 3 different subsequent networks in our segmentation model each of them are trained to accomplish different priorities. We named them as; Prediction Network, Denoise Network and Transmit Network.

3.3.3.1 Prediction Sub-Network

First sub-network of segmentation network is 'Prediction Sub-Network', main task of this sub-network is making predictions as accurate as possible. For this purpose, influence of Accuracy Loss (Eqn. 31) is higher than the Smoothness Loss (Eqn. 32) in the training of this sub-network. Major part of the recognition will be accomplished in this part of the model. This sub-network includes both Graph Convolutional Layers and Linear Layers.

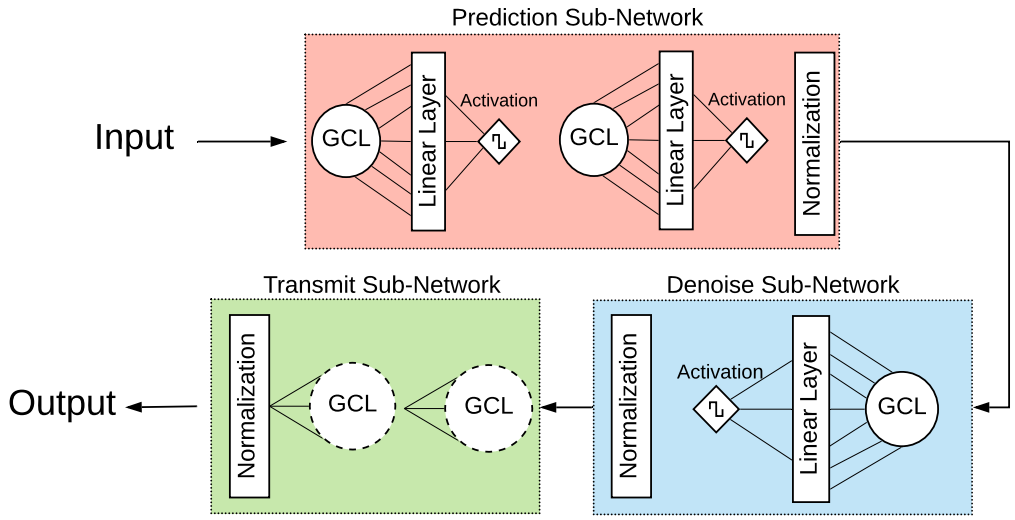


Figure 3.5: Illustration of the segmentation network.

3.3.3.2 Denoise Sub-Network

Second sub-network is 'Denoise Sub-Network'. Previous network generates first round of the scores with satisfying accuracy. For 3D segmentation purpose having good recognition rate is a major sign of successful segmentation but not always sufficient by itself. Mispredicted parts in 3D mesh object might affect consistency of segmentation adversely. For this reason, results of the first sub-network have been processed in this network. The difference from the first network is that Smoothness Loss has higher impact than the Accuracy Loss in this network. As a result of this approach, without losing the affect of Accuracy Loss, each polygon in our 3D object becomes more similar to its neighbors. This sub-network includes both Graph Convolutional Layers and Linear Layers.

3.3.3.3 Transmit Sub-Network

Last sub-network of our segmentation model is 'Transmit Sub-Network'. This sub-network does not have the training phase and has a quite simple job to perform which is basically replace your scores with the average of your neighbours. In other words, this module is the minimal version of the Denoise Sub-Network that targets a few

noisy polygon left in the middle of the correct polygons. According to results, we observed a slight improvement in accuracy and slight decrease in noise with this empirical approach, so we decided to add this sub-network to our model as the last module of the pipeline. This sub-network includes only Graph Convolutional Layers with all of the weights are constant, which means definition of loss function and training process are not necessary in this module.

3.4 Training Model

After defining network architecture, we need to complete training process for in order to get meaningful segmentation result.

Overall algorithm for training is summarized in Algorithm 1.

Algorithm consists of general steps used most commonly in DNNs. However, there exists a handicap for this training process. Taking derivative of GCLs during backward process requires passing through the adjacency matrix of graph structure for every single GCL. Using more GCLs caused more slower training process. Another performance issue to mention is that since GCLs require feature passing mechanism specific to each graph structure. It is not possible to merge multiple training data as a single input where we can easily success for uniform data structures like images. For this reason, concurrent network passing could not be applied for training data. For every single input in training data, we had to complete all loss calculations to handle new one. Under these circumstances, training time becomes disproportionately higher compared to common CNNs.

Algorithm 1: Training Segmentation Network

Input : Labeled & Pre-Processed Training Data Set, T; Labeled & Pre-Processed Validation Data Set, V; Number of Epochs, m; Batch Size, b; Learning Rate, l; L_1 factor, α; L_2 factor, β

Output: Learned Weights, W

initializate W
for m *epochs* **do**
    BatchSet $\leftarrow$ create subsets of T with size B
    foreach $batch$ *in BatchSet* **do**
        $L \leftarrow$ calculate overall loss (Eqn. 33) for $batch$ with L_1 (Eqn. 31) factor α and L_2 (Eqn. 32) factor β
        $grads \leftarrow$ Calculate L Backward
        Update W with gradient $grads$ and learning rate l by using Adam Optimization Algorithm [78]
    end
    $L_{val} \leftarrow$ calculate overall loss for V with L_1 factor α and L_2 factor β
    if L_{val} *do not decrease for few epochs* **then**
        Update l
    end
end
return W

3.5 Post-Process Stage

In pre-process stage, we changed structure of our 3D mesh model to make it usable in our network. In this stage, we will transfer the labeling information of our pre-processed structure, retrieved from our network, to the original one. We have used pretty straightforward approach to transfer this information. Since our upsampling/-downsampling methods do not change positions of the object parts, we may directly

transfer label information to original object structure by their positions.

Algorithm 2: Transferring Pre-Processed Object Labeling

Input : Labeled Pre-Processed 3D Object, P; Original 3D Object, O

Output: Labeled Original 3D Object

```
foreach face f in O do
    c ← closest face to f in P ;
    Set label of c to f ;
end
return O
```

In this stage, there might be added several computer graphics techniques such as point clustering, contextual feature labeling to increase overall performance of the segmentation. Since we focus on handling segmentation task through Deep Neural Networks, we did not prefer to use any extra computer graphics technique in this step.

CHAPTER 4

EXPERIMENTS AND EVALUATION

In this chapter, firstly, we demonstrate the dataset used for our experiments, our tools and methods for pre-processing and post-processing applications. Then, the training process and architecture details explained in detail. Lastly, results and evaluation of our proposed method examined.

4.1 Dataset

Since our approach is a supervised method, we need to acquire a qualified and correspondingly labeled 3D object dataset that serves better for our segmentation purpose. Despite there are plenty of widely used 3D shape datasets like ShapeNet and ModelNet, they are not directly usable in our network for the reason that they are not correspondingly labeled. Thus, we needed to find alternative datasets for our experiments, which preferably correspondingly aligned. The reason for searching a correspondingly aligned dataset is that if the object group is aligned, we might transfer the single object's ground-truth segmentation information to the rest of the group easily. Which means we would complete manual segmentation once for each one of the datasets.

In all our experiments, we used %80 of dataset as training set, %10 of dataset as validation set and %10 of dataset as test set.

4.1.1 FAUST Dataset

FAUST dataset [11] includes 100 triangulated, high-resolution mesh of 10 different subjects. Each one of these subjects was scanned in 10 different poses and all of them

are aligned.

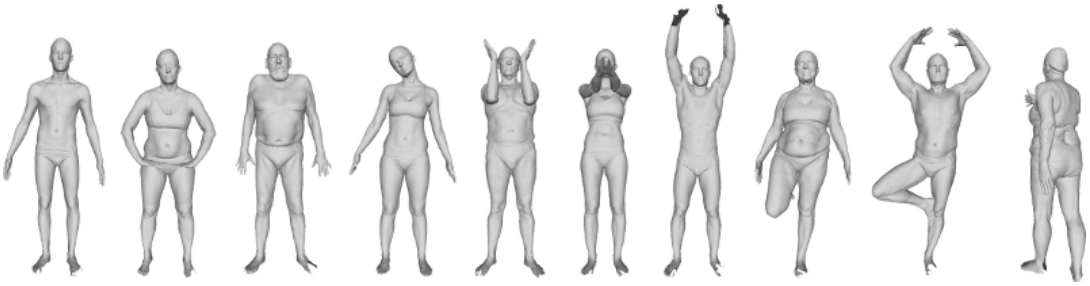


Figure 4.1: Overview of the 10 poses in the FAUST dataset (Figure Source: [11])

We used Blender [79] software to visualize models, apply face labeling and transfer face labeling to other models in the dataset. Blender is an open-source 3D creation suite which has great number of features and Python API.

For each one of the 100 models in this dataset, we labeled each face as one of the four classes, which represents head, torso, arms and legs, and also created 90°, 180° and 270° rotated versions on z-axis in the scope of data augmentation (See Figure 3.2).

As a result, we created 400 ready to use consistent and labeled 3D objects from FAUST dataset with a little effort. If the models were not aligned correspondingly, it would make labeling process more difficult and time-consuming.

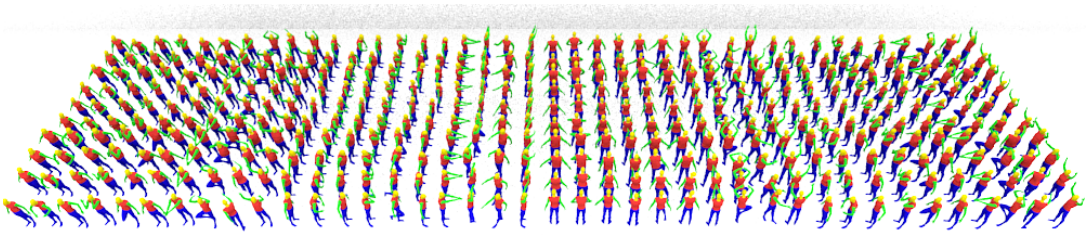


Figure 4.2: Overview of 400 labeled and augmented FAUST models (Augmented version of Figure 3.1).

4.2 Pre-Processing and Data Extraction

After having a set of ground-truth labeled 3D objects, we have applied pre-process application as stated in Section 3.2. In this stage, we decided to normalize face count

of objects to 2000. Every object in FAUST dataset has 13776 faces originally. We applied mesh simplification process to all of the objects in our dataset. For mesh simplification process, we used modification support of Blender software. The ability to modify 3D objects with Python interface made it easy to adjust mesh structure and visualize results directly.

Next step was converting our 3D objects into numeric input features in order to consume it in our neural network. Determined features are the position and the unit normal vector of the face. We used NumPy [80] library to store values in the matrix format. Since our dataset objects are closed manifold and consist of triangulated polygons, each face has exactly 3 adjacent faces. So, every face in our dataset transformed into 6 input features, 1 label index and 3 adjacent indices. As a result, we had input feature matrix with size $400 \times 2000 \times 6$, label index matrix with size 400×2000 and adjacency index matrix with size $400 \times 2000 \times 3$.

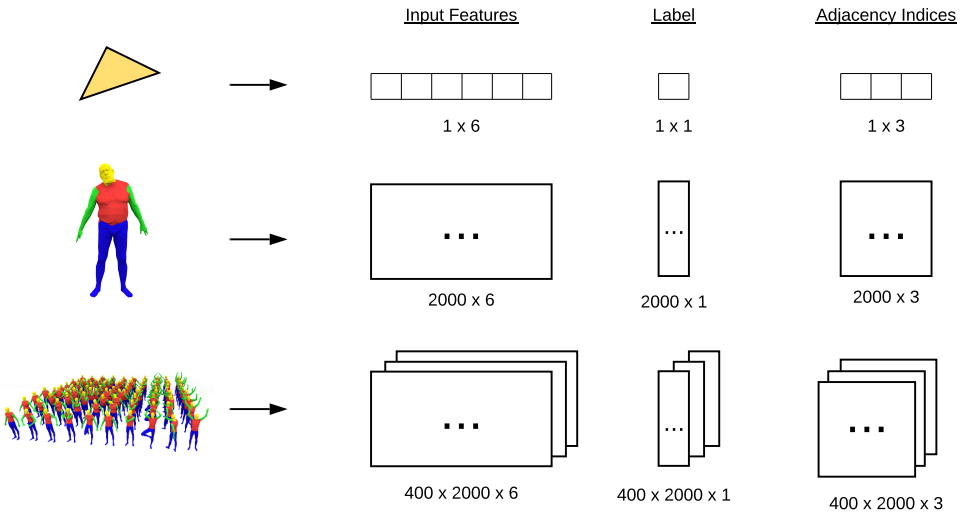


Figure 4.3: Illustration of 3D visual object to nominal value conversion.

4.3 Experiments and Architecture Details

We trained our models using PyTorch [81] and Deep Graph Library (DGL) [82]. In all of our experiments we trained weights from scratch. We used NVIDIA GeForce 1080Ti GPU for our training processes.

There are several hyper-parameters and architectural decisions in our model that we need to optimize. We have 2 different networks (Prediction Sub-Network and Denoise Sub-Network) to train independently. Since our Prediction Sub-Network is the most important part of the segmentation model. We completed most of our experiments on it.

4.3.1 Activation Function

Activation function is one of the architectural decisions that used in all layers of the networks. We tried 4 different activation functions to find the best one fits our problem. All parameters and conditions are equal in this experiment except activation functions.

Table 4.1: Training details for activation function experiment. (G) sign indicates the layer is graph convolutional layer, which means before passing to the next layer feature passing operation applied. 'Dropout' indicates dropout rate between layers. 'Activation' indicates activation function used between layers (not specified for this case). 'B.Size' indicates batch size used during training process. 'Opt.' indicates optimizer method used in training process. 'Epoch' indicates epoch size for training. α and β indicates loss function constants in Eqn. 33. 'LR' is the initial learning rate used in training. 'LR-P' and 'LR-F' indicates scheduler patience and scheduler factor of learning rate, respectively (After 30 epoch LR multiplied by 0.5 for this case).

Layers									
$6(G) \times 128(G) \times 512(G) \times 256(G) \times 128(G) \times 4$									
Dropout	Activation	B.Size	Opt.	Epoch	α	β	LR	LR-P	LR-F
0.3	-	10	Adam	100	1.0	0.3	0.001	30	0.5

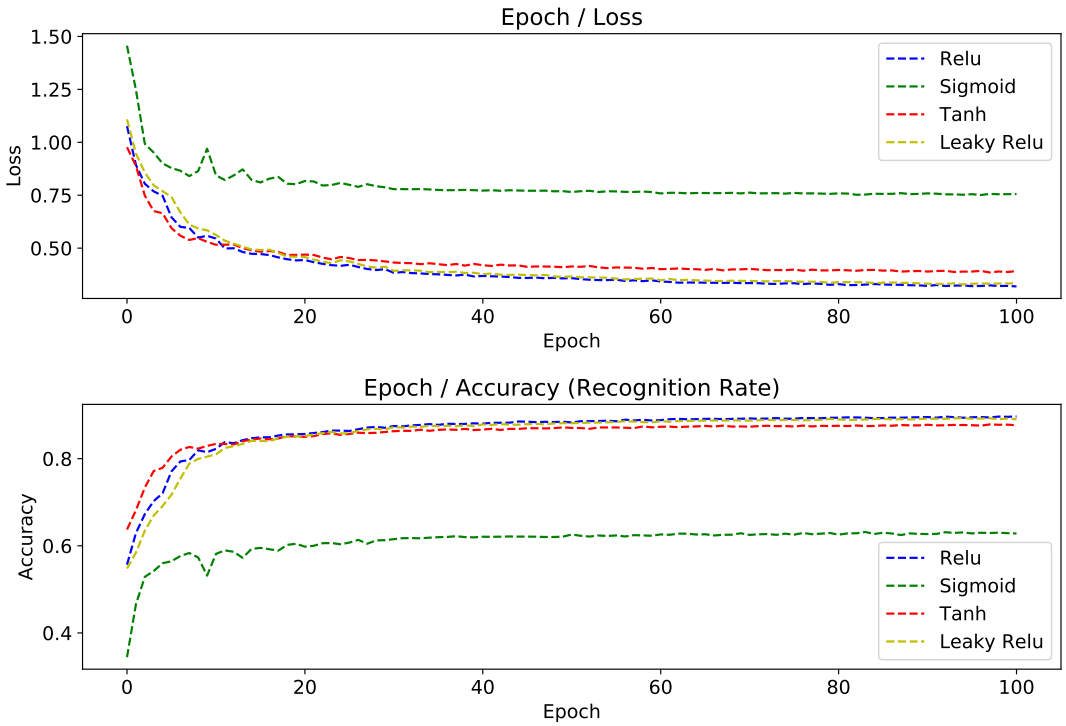


Figure 4.4: Loss and accuracy plot of different activation functions in Prediction Sub-Network.

After 100 epochs of training, results are Relu > Leaky Relu > Tanh > Sigmoid. Relu activation function seems the most appropriate one for our problem.

4.3.2 Dropout Rate

Dropout is an important regularization technique which mainly used in order to prevent overfitting in networks. We trained our network with different dropout rates to check the appropriateness of dropout in our problem and dataset.

Table 4.2: Training details for dropout rate experiment.

Layers									
$6(G) \times 128(G) \times 512(G) \times 256(G) \times 128(G) \times 4$									
Dropout	Activation	B.Size	Opt.	Epoch	α	β	LR	LR-P	LR-F
-	Relu	10	Adam	100	1.0	0.3	0.001	30	0.5

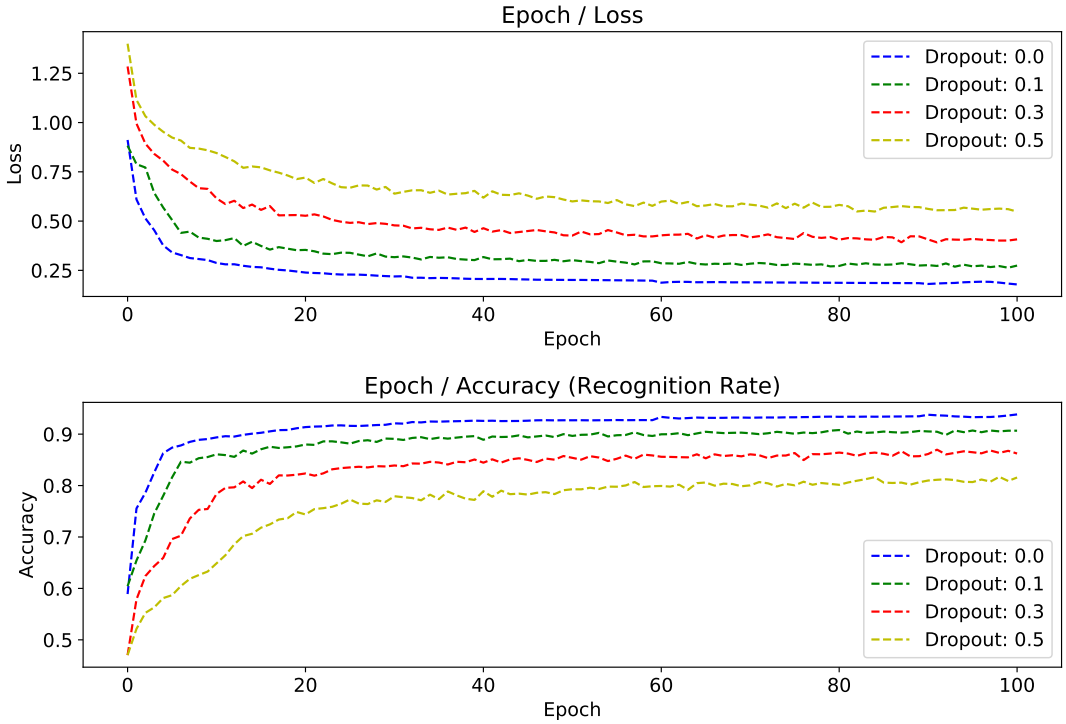


Figure 4.5: Loss and accuracy plot of different dropout rates in Prediction Sub-Network.

After 100 epochs of training with different dropout rates, not using dropout seems as the best case for our network. However, we needed to check if this might cause an overfitting problem and compared our validation set accuracy and test set accuracy.

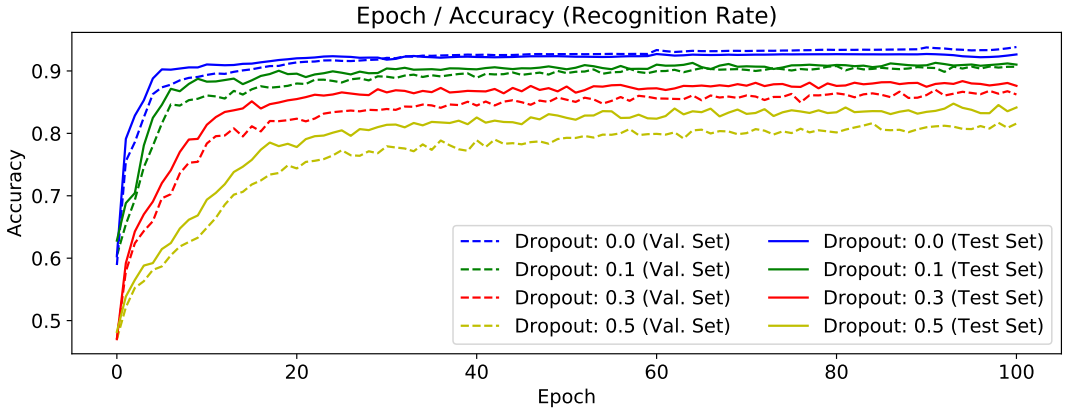


Figure 4.6: Accuracy plot of different dropout rates for validation set and test set.

Using no dropout resulted in slightly higher (around %1) validation accuracy, however, test set accuracy still the best one. There does not exist any sign of overfitting with no dropout. Dropout rate around 0.1 seems the best value in context of fitting, however final result is a little bit worse than no dropout. Also, higher amount of dropout rates indicates the sign of underfitting with higher test set accuracy.

4.3.3 Layer Architecture

We have run experiments to find the best layer architecture to fit our model. Increasing the number of GCLs caused huge increase in training time up to 12 hours.

Our experimented layer architectures are as follows;

Table 4.3: Different layer architecture details and their names for the experiment.

NET1	$6(G) \times 64(G) \times 128(G) \times 64 \times 4$
NET2	$6(G) \times 24(G) \times 64(G) \times 128(G) \times 256(G) \times 128(G) \times 64(G) \times 24(G) \times 4$
NET3	$6(G) \times 128(G) \times 1024(G) \times 128(G) \times 64 \times 4$
NET4	$6(G) \times 64(G) \times 128(G) \times 512 \times 64 \times 4$
NET5	$6(G) \times 64(G) \times 128(G) \times 256(G) \times 128(G) \times 4$
NET6	$6(G) \times 64(G) \times 128(G) \times 256(G) \times 512(G) \times 64 \times 4$
NET7	$6(G) \times 128(G) \times 512(G) \times 256 \times 128 \times 4$
NET8	$6(G) \times 128(G) \times 512(G) \times 256(G) \times 128(G) \times 4$

Table 4.4: Remaining parameters for layer architecture experiment.

Dropout	Activation	B.Size	Opt.	Epoch	α	β	LR	LR-P	LR-F
0.0	Relu	10	Adam	100	1.0	0.3	0.001	30	0.5

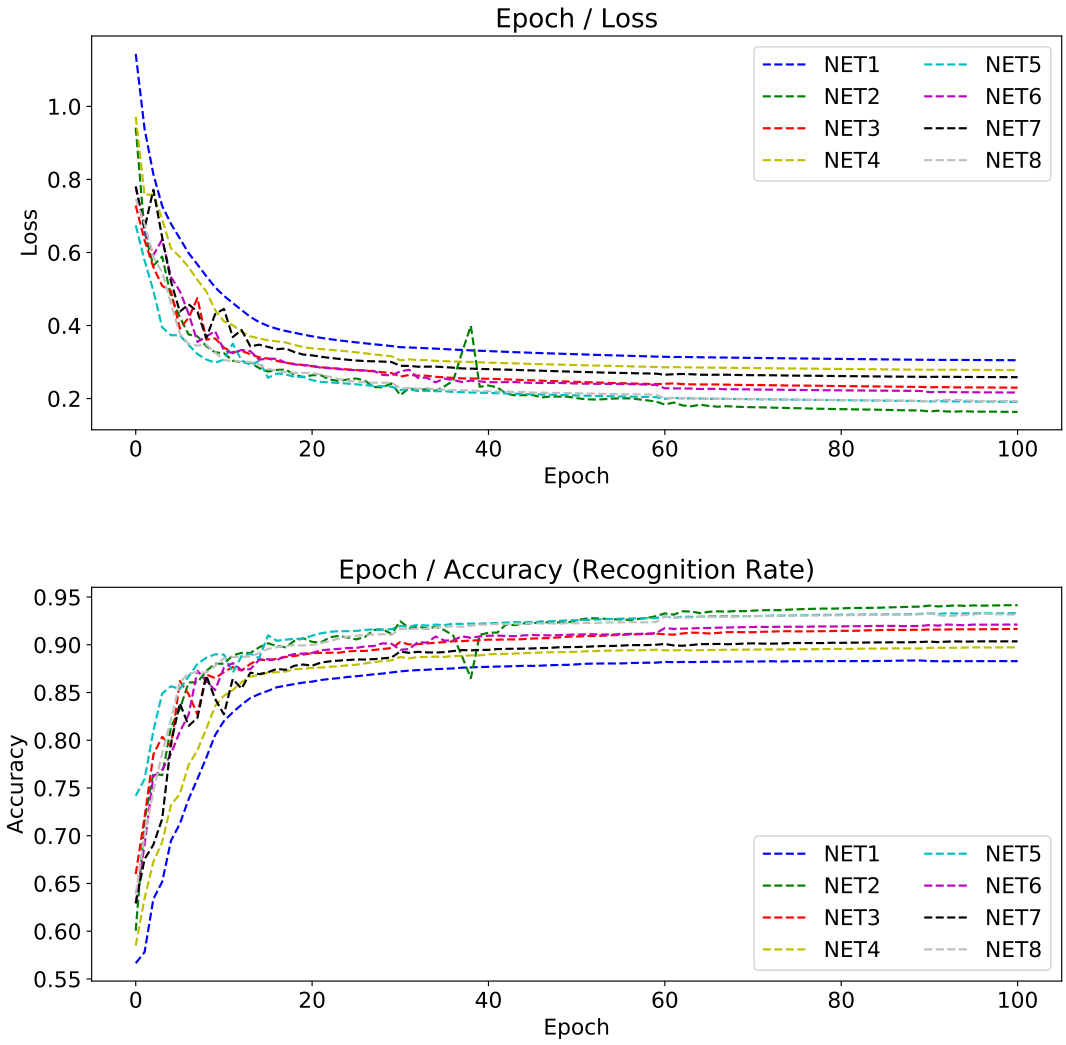


Figure 4.7: Loss and accuracy plot of different layer architectures in Prediction Sub-Network.

Table 4.5: Recognition rate values of test set for different layer architectures.

NET1	NET2	NET3	NET4	NET5	NET6	NET7	NET8
0.883	**0.941**	0.916	0.897	0.933	0.921	0.904	0.932

NET2 appears as the most successful configuration for our model. It could be seen from Table 4.3 that NET7 and NET8 have the same number of hidden layer configuration, however, last 2 layers of NET8 is GCL, so we can see the affect of using GCLs in our model.

4.3.4 α and β Constants

α and β are the stabilizer values of the loss function (Eqn. 33). If our case was making labeling only, we probably would not need a smoothness term in our loss function. However, in segmentation problem, noises are not desirable at all. For Prediction Sub-Network α value must be relatively higher than β value, since accuracy is the primary concern of this first network. For Denoise Sub-Network concern is smoothness, oppositely.

We have made several experiments for the best accuracy that we can obtain with the following setup;

Table 4.6: Training details for α and β experiment.

Layers									
$6(G) \times 24(G) \times 64(G) \times 128(G) \times 256(G) \times 128(G) \times 64(G) \times 24(G) \times 4$									
Dropout	Activation	B.Size	Opt.	Epoch	α	β	LR	LR-P	LR-F
0.0	Relu	10	Adam	100	1.0	-	0.001	30	0.5

For standardization α value is set to 1.0 and β values are changed. Since the scale of the loss function changes with these values, final loss value is not a meaningful indicator in this experiment.

Table 4.7: Recognition rate values of test set for different β values.

β	0.0	0.1	0.2	0.3	0.4	0.5
R.Rate	0.9369	0.9435	0.9428	0.9385	**0.9476**	0.9376
β	0.6	0.7	0.8	0.9	1.0	1.5
R.Rate	0.9455	0.942	0.9348	0.9402	0.9425	0.9346

Up to a specific point, β value improved the recognition rate of the network, since smoothness is a positive factor for the correctness of the segmentation. However, after that point, accuracy started to decrease. Higher β value dominates the loss function and network starts to give more importance on the smoothness than the accuracy.

This leads to lose boundary edges of the object. Since torso part of the human shape has the longest boundary edge and more volume in the whole body, torso part starts to suppress other parts of the object. On the other hand, lower β value leads to the apparent labeling fluctuation, mostly on the boundary edges, since the network does not give much importance on the labeling of the neighbourhoods. This balance between α and β directly affects the error rate of the network and they are the most crucial hyper-parameters for the overall performance of the network.

With $\alpha = 1.0$ and $\beta = 0.4$ values, accuracy is the highest according to our experiments.

4.3.5 Denoise Sub-Network Training

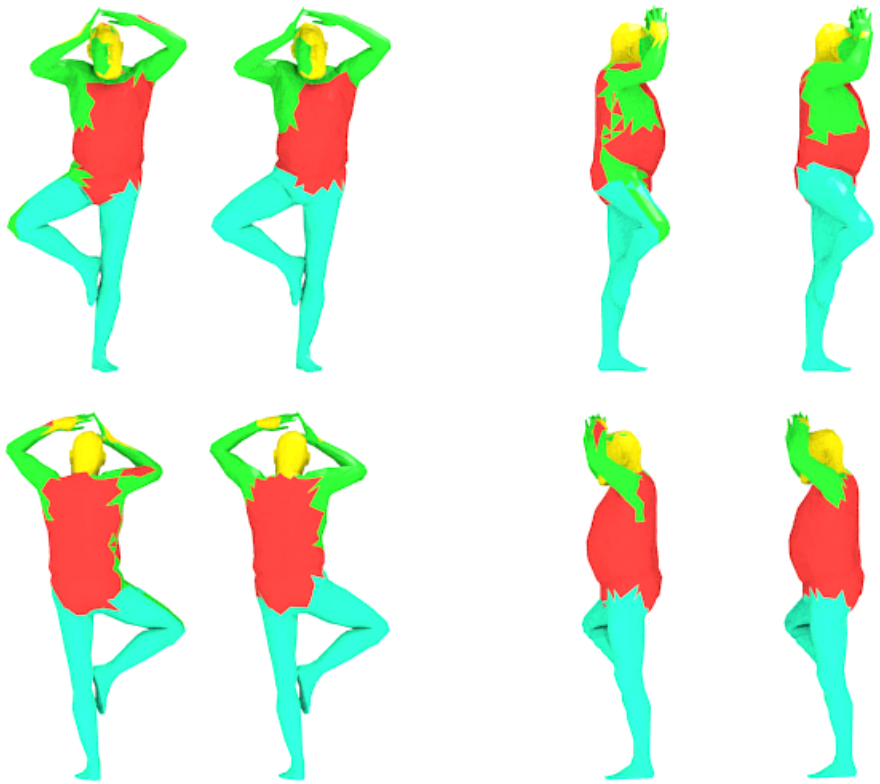


Figure 4.8: Effect of Denoise Sub-Network on a sample object from different views. On each pair, left object represents the input segmentation of Denoise Sub-Network, right object represents the output segmentation of Denoise Sub-Network.

This network is a quite similar network to Prediction Sub-Network, except the balance between α and β. However, this network does not provide a visible increase in recognition rate of the final labeling. It only eliminates the noisy labels in the final prediction for the sake of segmentation purpose. That's why recognition rate values are not a good indicator in the training phase of this network.

Parameters are mostly inherited from the previous experiments. Sample architecture of the Denoise Sub-Network;

Table 4.8: Parameters for Denoise Sub-Network.

Layers									
$4(G) \times 24(G) \times 64(G) \times 128(G) \times 256(G) \times 128(G) \times 24(G) \times 4$									
Dropout	Activation	B.Size	Opt.	Epoch	α	β	LR	LR-P	LR-F
0.0	Relu	10	Adam	100	1.0	1.0	0.001	30	0.5

For the models that labeled successfully in the first network, denoise application is quite useless and does not provide a visible outcome. So, we need to visualize not successfully recognized models to see the difference. The 3D object in Figure 4.8 is the least successfully recognized model in the test set. Effect of denoise application could easily be seen on unsuccessful objects. Overall accuracy performance of the network is not improved but the noise on the object have been mostly overcome.

After training phase of this network, average test set accuracy improved from 0.9476 to 0.9531.

4.4 Evaluation

In this section of thesis, we evaluated our method with the previous supervised 3D mesh segmentation methods for human shapes. We made our experiments on the FAUST dataset which has 100 unique human model, however, previous methods made experiments on PSB [83] dataset which has 20 models on different categories.

For evaluation, leave-one-out cross validation technique has been used. 19 meshes

are selected for training and 1 mesh is used for testing. This process is completed 20 times for each single human model and the average accuracy is calculated. As in Kalogerakis et al. [7], the accuracy is computed as:

$$Acc = \sum_{t \in T} a_t g_t(l_t) / \sum_{t \in T} a_t, \tag{41}$$

where a_t is the area of the triangle t and l_t is the predicted label. $g_t(l_t)$ denotes the l_tth component of g_t, which equals to 1 if the prediction is correct.

Table 4.9: Evaluation table with different methods. First column represents the paper. Second column is the final accuracy on the human objects. Third column is the used feature counts for the methods. Fourth column is the checkbox for the method using Graph-Cut optimization or not. Last column is the main method used in the paper.

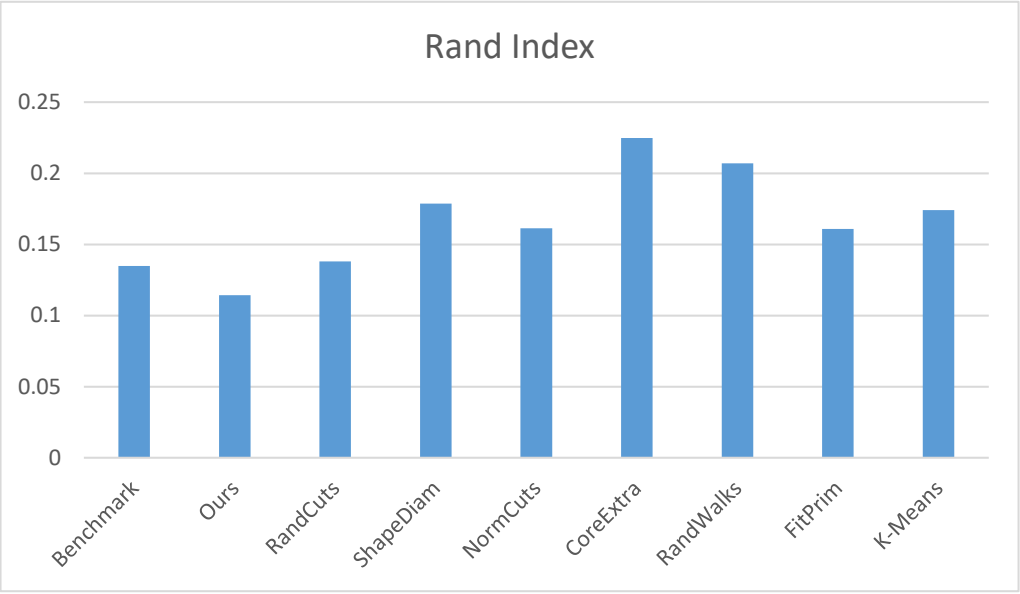
	Accuracy (SB19)	F.C.	GC Opt.	Method
Kalogerakis et al. [7] [2010]	$\mathbf{93.60}\%$	810	✓	CRF Opt.
Xie et al. [41] [2014]	88.61%	619	✓	Extreme LM
Guo et al. [46] [2015]	88.90%	600	✓	CNN
George et al. [47] [2018]	89.81%	800	✓	CNN
Ours	88.13%	**6**	✗	Graph-CNN
Ours	91.49%	593	✗	Graph-CNN

Firstly, we trained our network with 6 features that we used in our experiments, and we achieved a successful accuracy in labeling. Later on, we trained a bigger network with 593 unary features that are shared by Kalogerakis et al. [7], and our accuracy value outperformed all other techniques using only unary features with the lowest feature count. The only method we could not surpass is also used pairwise features of the faces as a second energy term in their network. All other methods used only unary features of the faces and our method is the best one among them. We also did not apply the Graph Cut post-optimization method to our results, which increases the accuracy around a few percents; instead, we used the straight output of our network. The method of Wang et al. [48] is not added to the evaluation table, since they did not share their accuracy on human shapes specifically.

Princeton Segmentation Benchmark

We also evaluated our segmentation performance with PSB [83]. PSB evaluates the performance of the segmentation with 4 different metrics. Rand Index metric measures the likelihood that a pair of faces are either in the same segment in two segmentations, or in different segments in both segmentations. Cut Discrepancy metric sums the distances from points along the cuts in the computed segmentation to the closest cuts in the ground truth segmentation, and vice-versa. Consistency error metric tries to account for nested, hierarchical similarities and differences in segmentations. Last metric hamming distance measures the overall region-based difference between two segmentation results.

For each metric in the benchmark lower value represents better performance. Results are given in Figure 4.9 and Figure 4.10. 'Benchmark' indicates the human-segmented baseline for benchmark and there are 7 different segmentation algorithms beside our model for comparison.



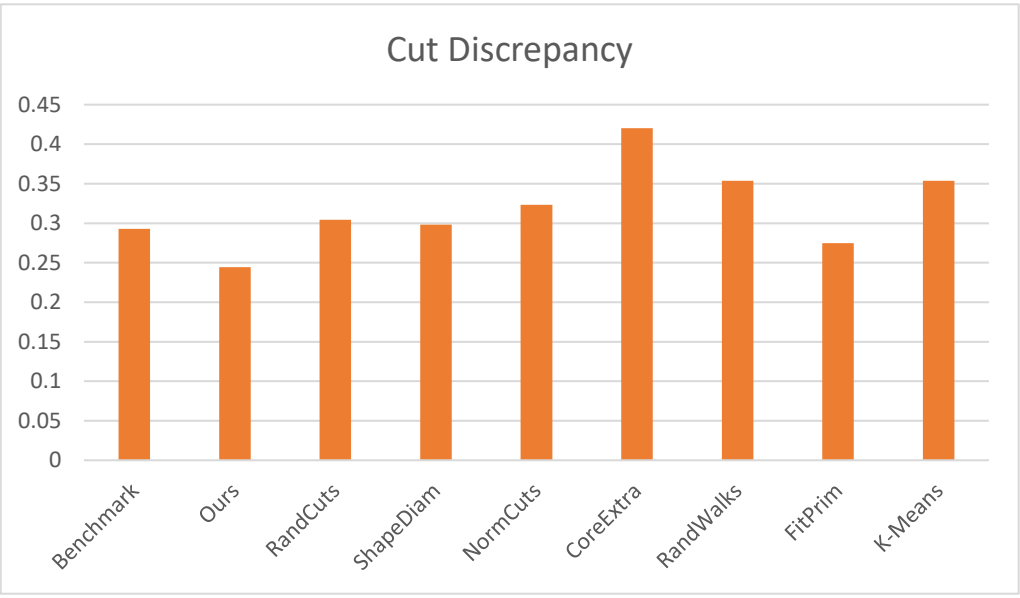
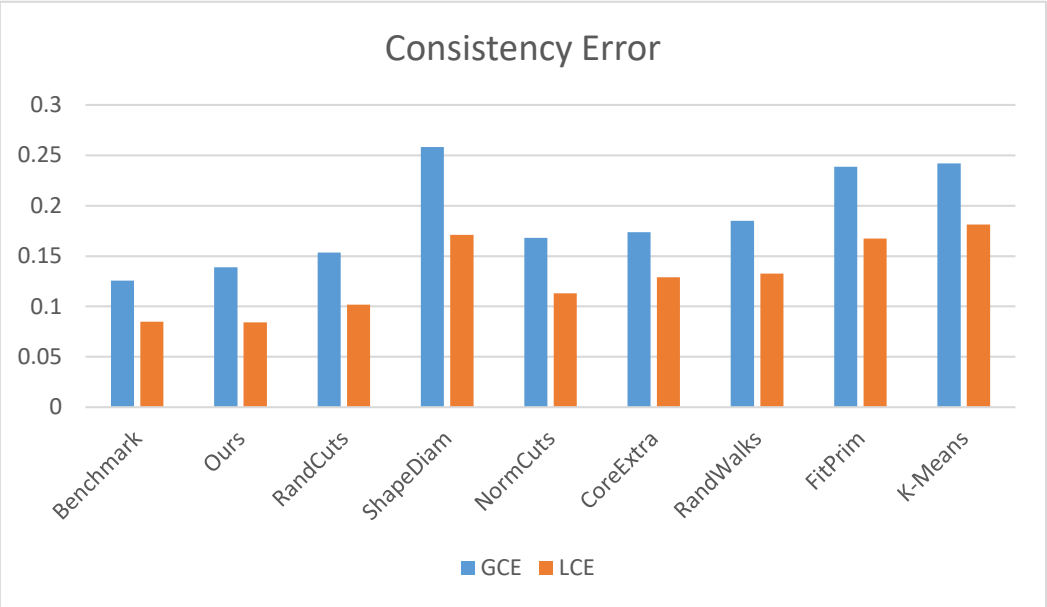


Figure 4.9: Rand Index and Cut Discrepancy metrics.



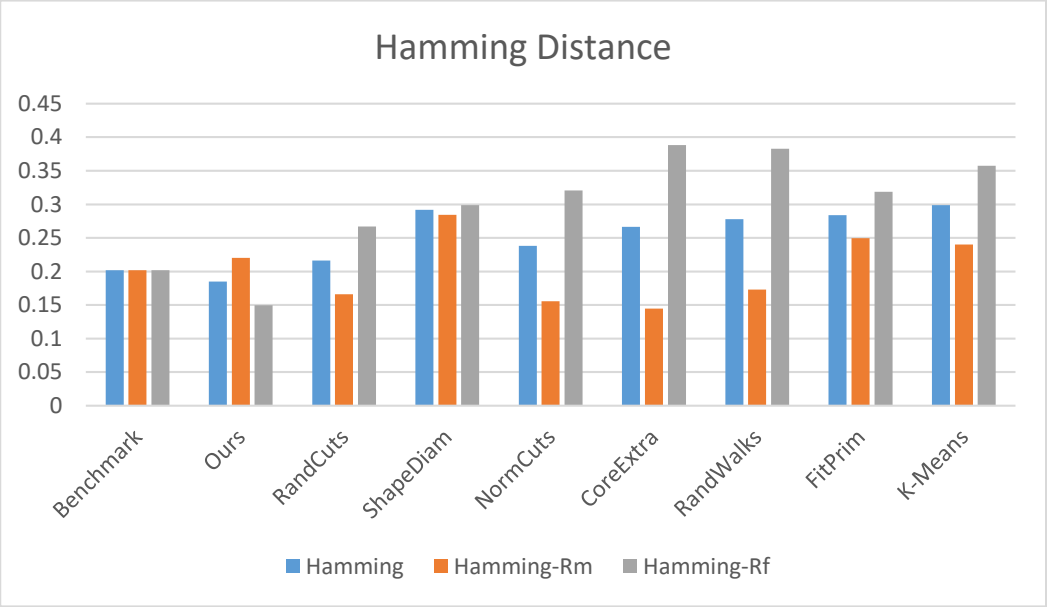


Figure 4.10: Consistency Error and Hamming Distance metrics.

4.5 Visual Results

We visualized the segmentation result of 6 objects from different view angles in the PSB dataset in Figure 4.11. Our network is highly successful to recognize body parts and there is not any visible defect in the results.

Figure 4.11: Visual results of sample objects.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this thesis, we studied the segmentation of 3D objects using Graph Convolutional Neural Networks with a supervised learning-based approach. We mainly focused on human shapes in this thesis since it is one of the most challenging and popular shape types for segmentation.

We neither handled each face in the object separately nor converted the irregular graph-based structure into a uniform grid-based structure like the previous learning-based approaches. We have developed a GCNN model that works directly on the graph structure of 3D objects, and this brought the ability to acquire competitive segmentation with significantly fewer features. We also tested our model with a large number of features like the previous works, and our results outperformed most of the state-of-the-art methods. Unlike previous approaches that used different post-optimization algorithms, we also developed a different network model for noise and boundary optimization by adjusting the smoothness power of loss function. This way, our method falls into the pure learning-based segmentation category.

5.1 Future Work

In the future, we plan to train the models on different datasets and different shape types. COSEG and PSB datasets have several combinations, and we need to check the capability of our model for each category in these datasets. We also plan to cross-check several features to detect the most significant ones for the best labeling accuracy with the fewer features possible. Moreover, the performance of the networks that are trained on a particular type and tested on a similar but different type (teddy bear and

human; plane and bird) could also be measured to check the generalization capacity of the segmentation model.

REFERENCES

[1] Y. Wang, S. Asafi, O. Van Kaick, H. Zhang, D. Cohen-Or, and B. Chen, "Active co-analysis of a set of shapes," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 165, 2012.

[2] R. S. Rodrigues, J. F. Morgado, and A. J. Gomes, "Part-based mesh segmentation: A survey," in *Computer Graphics Forum*, vol. 37, pp. 235–274, Wiley Online Library, 2018.

[3] M. Attene, M. Mortara, M. Spagnuolo, and B. Falcidieno, "Hierarchical convex approximation of 3d shapes for fast region selection," in *Computer graphics forum*, vol. 27, pp. 1323–1332, Wiley Online Library, 2008.

[4] S. Katz and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts," in *ACM Transactions on Graphics (TOG)*, vol. 22, pp. 954–961, ACM, 2003.

[5] A. Golovinskiy and T. Funkhouser, "Randomized cuts for 3d mesh analysis," in *ACM transactions on graphics (TOG)*, vol. 27, p. 145, ACM, 2008.

[6] J. Aleotti and S. Caselli, "A 3d shape segmentation approach for robot grasping by parts," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 358–366, 2012.

[7] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3d mesh segmentation and labeling," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 102, 2010.

[8] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3d shape segmentation with projective convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3779–3788, 2017.

[9] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, pp. 3844–3852, 2016.

[11] F. Bogo, J. Romero, M. Loper, and M. J. Black, "FAUST: Dataset and evaluation for 3D mesh registration," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, (Piscataway, NJ, USA), IEEE, June 2014.

[12] A. P. Mangan and R. T. Whitaker, "Partitioning 3d surface meshes using watershed segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308–321, 1999.

[13] G. Lavoué and C. Wolf, "Markov random fields for improving 3d mesh analysis and segmentation.," in *3DOR*, pp. 25–32, 2008.

[14] L. Shapira, A. Shamir, and D. Cohen-Or, "Consistent mesh partitioning and skeletonisation using the shape diameter function," *The Visual Computer*, vol. 24, no. 4, p. 249, 2008.

[15] S. Katz, G. Leifman, and A. Tal, "Mesh segmentation using feature point and core extraction," *The Visual Computer*, vol. 21, no. 8-10, pp. 649–658, 2005.

[16] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, "Fast mesh segmentation using random walks," in *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pp. 183–191, ACM, 2008.

[17] A. Shamir, "A survey on mesh segmentation techniques," in *Computer graphics forum*, vol. 27, pp. 1539–1556, Wiley Online Library, 2008.

[18] P. Theologou, I. Pratikakis, and T. Theoharis, "A comprehensive overview of methodologies and performance evaluation frameworks in 3d mesh segmentation," *Computer Vision and Image Understanding*, vol. 135, pp. 49–82, 2015.

[19] M. Rashad, M. Khamiss, and M. MOUSA, "A review on mesh segmentation techniques," 2017.

[20] B. Chazelle, "Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm," *SIAM Journal on Computing*, vol. 13, no. 3, pp. 488–507, 1984.

[21] B. Jüttler, M. Kapl, and Q. Pan, "Isogeometric segmentation. part i: Decomposing contractible solids without non-convex edges," 2013.

[22] D.-M. Nguyen, M. Pauley, and B. Jüttler, "Isogeometric segmentation. part ii: On the segmentability of contractible solids with non-convex edges," *Graphical Models*, vol. 76, no. 5, pp. 426–439, 2014.

[23] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polygons," *Computational Geometry*, vol. 35, no. 1-2, pp. 100–123, 2006.

[24] V. K. D. J. A. Sheffer, "Shuffler: Modeling with interchangeable parts," *Visual Computer journal*, 2007.

[25] H. Liu, W. Liu, and L. J. Latecki, "Convex shape decomposition," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 97–104, IEEE, 2010.

[26] C. Xian, S. Gao, and T. Zhang, "An approach to automated decomposition of volumetric mesh," *Computers & Graphics*, vol. 35, no. 3, pp. 461–470, 2011.

[27] P. Simari, D. Nowrouzezahrai, E. Kalogerakis, and K. Singh, "Multi-objective shape segmentation and labeling," in *Computer Graphics Forum*, vol. 28, pp. 1415–1425, Wiley Online Library, 2009.

[28] S. Shlafman, A. Tal, and S. Katz, "Metamorphosis of polyhedral surfaces using decomposition," in *Computer graphics forum*, vol. 21, pp. 219–228, Wiley Online Library, 2002.

[29] R. Liu and H. Zhang, "Segmentation of 3d meshes through spectral clustering," in *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pp. 298–305, IEEE, 2004.

[30] P. Simari, E. Kalogerakis, and K. Singh, "Folding meshes: Hierarchical mesh segmentation based on planar symmetry," in *Symposium on geometry processing*, vol. 256, pp. 111–119, 2006.

[31] Q.-X. Huang, M. Wicke, B. Adams, and L. Guibas, "Shape decomposition using modal analysis," in *Computer Graphics Forum*, vol. 28, pp. 407–416, Wiley Online Library, 2009.

[32] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical mesh segmentation based on fitting primitives," *The Visual Computer*, vol. 22, no. 3, pp. 181–193, 2006.

[33] H.-Y. S. Lin, H.-Y. M. Liao, and J.-C. Lin, "Visual salience-guided mesh decomposition," *IEEE Transactions on Multimedia*, vol. 9, no. 1, pp. 46–57, 2006.

[34] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker, "Hamilton-jacobi skeletons," *International Journal of Computer Vision*, vol. 48, no. 3, pp. 215–231, 2002.

[35] M. Mortara, G. Patané, and M. Spagnuolo, "From geometric to semantic human body models," *Computers & Graphics*, vol. 30, no. 2, pp. 185–196, 2006.

[36] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee, "Skeleton extraction by mesh contraction," in *ACM transactions on graphics (TOG)*, vol. 27, p. 44, ACM, 2008.

[37] C. Lovato, U. Castellani, and A. Giachetti, "Automatic segmentation of scanned human body using curve skeleton analysis," in *International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications*, pp. 34–45, Springer, 2009.

[38] D. Reniers and A. Telea, "Skeleton-based hierarchical shape segmentation," in *IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07)*, pp. 179–188, IEEE, 2007.

[39] X. Li, T. W. Woon, T. S. Tan, and Z. Huang, "Decomposing polygon meshes for interactive applications," in *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 35–42, ACM, 2001.

[40] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 854–869, 2007.

[41] Z. Xie, K. Xu, L. Liu, and Y. Xiong, "3d shape segmentation and labeling via extreme learning machine," in *Computer graphics forum*, vol. 33, pp. 85–95, Wiley Online Library, 2014.

[42] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, *et al.*, "Extreme learning machine: a new learning scheme of feedforward neural networks," *Neural networks*, vol. 2, pp. 985–990, 2004.

[43] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[44] A. Makadia and M. E. Yumer, "Learning 3d part detection from sparsely labeled data," in *2014 2nd International Conference on 3D Vision*, vol. 1, pp. 311–318, IEEE, 2014.

[45] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of machine learning research*, vol. 6, no. Sep, pp. 1453–1484, 2005.

[46] K. Guo, D. Zou, and X. Chen, "3d mesh labeling via deep convolutional neural networks," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 1, p. 3, 2015.

[47] D. George, X. Xie, and G. K. Tam, "3d mesh segmentation via multi-branch 1d convolutional neural networks," *Graphical Models*, vol. 96, pp. 1–10, 2018.

[48] P. Wang, Y. Gan, P. Shui, F. Yu, Y. Zhang, S. Chen, and Z. Sun, "3d shape segmentation via shape fully convolutional networks," *Computers & Graphics*, vol. 70, pp. 128–139, 2018.

[49] Y. Wang, M. Gong, T. Wang, D. Cohen-Or, H. Zhang, and B. Chen, "Projective analysis for 3d shape segmentation," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 192, 2013.

[50] J. Lv, X. Chen, J. Huang, and H. Bao, "Semi-supervised mesh segmentation and labeling," in *Computer Graphics Forum*, vol. 31, pp. 2241–2248, Wiley Online Library, 2012.

[51] L. Liao, T. Choudhury, D. Fox, and H. A. Kautz, "Training conditional random fields using virtual evidence boosting.," in *Ijcai*, vol. 7, pp. 2530–2535, 2007.

[52] Z. Shu, X. Shen, S. Xin, Q. Chang, J. Feng, L. Kavan, and L. Liu, "Scribble based 3d shape segmentation via weakly-supervised learning," *IEEE transactions on visualization and computer graphics*, 2019.

[53] V. Kreavoy, D. Julius, and A. Sheffer, "Model composition from interchangeable components," in *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pp. 129–138, IEEE, 2007.

[54] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, *Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering*, vol. 30. ACM, 2011.

[55] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[56] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[59] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[61] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8614–8618, IEEE, 2013.

[62] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[63] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[64] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep multilingual convolutional neural networks for lvcsr," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4955–4959, IEEE, 2016.

[65] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[66] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2765–2770, IEEE, 2016.

[67] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[68] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[69] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," in *Advances in neural information processing systems*, pp. 3338–3346, 2014.

[70] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, pp. 2224–2232, 2015.

[71] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[72] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[73] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, pp. 2014–2023, 2016.

[74] J.-C. Vialatte, V. Gripon, and G. Mercier, "Generalizing the convolution operator to extend cnns to irregular domains," *arXiv preprint arXiv:1606.01166*, 2016.

[75] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[76] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[77] B. L. Douglas, "The weisfeiler-lehman method and graph isomorphism testing," *arXiv preprint arXiv:1101.5211*, 2011.

[78] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[79] The Blender Foundation, "Blender (v2.79b)," 2002. https://blender.org.

[80] T. E. Oliphant, *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.

[81] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[82] NYU, NYU Shanghai, AWS Shanghai AI Lab, and AWS MXNet Science Team, "Deep graph library (v0.1)," 2018. https://dgl.ai.

[83] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3D mesh segmentation," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, Aug. 2009.