CSE 881 FINAL PROJECT (Cover Page)

Project Title: 3D Mesh Segmentation and Labeling

Project Type (choose one): Algorithm Development

Difficulty Level (choose one): Moderate

Justification for our rating: Data collection,Data pre-processing,Algorithm development,Evaluation

Summary of Team Member Participation:

Fill out the following table for each team member with a rating from 1 to 3 (1: poor, 2: satisfactory, 3: good). For "responsive to emails" and "attendance at project meetings", the rating must be provided by averaging the rating provided by other members of the group.

Name	**Responsive to emails***	**Attended project meetings***	**Participate in data collection /preprocessing**	**Participate in coding**	**Participate in analysis/ experiments**	**Writing final report**	**Class presentation**	**Completed Assigned Tasks**
Ze Zhang	3	3	3	3	3	3	2	3
Hayam	3	3	3	3	3	3	2	3
Saptarshi	3	3	2	3	3	3	3	3

Team Member Roles and Contributions:

Name	**Roles and Contributions**
Ze Zhang	Responsible for building and training CNN model; help writing the final report
Saptarshi	Coding and perform data/result analysis/causality ; primary author of final report
Hayam	Perform SVM classification for comparison , participated in data formating and preprocessing , writing the final report

I approve the content of the final report (please add your signature below):

Ze Zhang: *Ze Zhang* ---

Saptarshi: *Saptarshi Mitra* ---

Hayam: *Hayam Abdelrahman* ---

3D Mesh Segmentation and Labeling

Ze Zhang
Michigan State University
zhangzezeze@gmail.com

Saptarshi MItra
Michigan State University
mitrasap@msu.edu

Hayam Abdelrahman
Michigan State University
abdelr14@msu.edu

ABSTRACT

Mesh segmentation plays a pivotal role in many computer graphics applications, for example, modeling generation, texture mapping and animation. With more and more 3D labelled models available, data-driven methods can be applied to this area for their application in various aforementioned domains. We use various machine learning methods to do mesh segmentation in our project, such as convolutional neural network and SVM. Convolutional Neural network outperforms in our project, and with more training data, it can get better accuracy.

Keywords
Mesh segmentation, CNN, SVM

1. INTRODUCTION

Segmentation and labeling of 3d meshes into meaning part is a key topic in 3d modeling and many other computer graphics applications.Segmentation assists in parametrization, texture mapping, shape matching, morphing, multi-resolution modelling, mesh editing, compression, animation and more. For example, if we need to texture synthesis to a human body, we need to know which part every triangle of this mesh belongs to, then we can add arm texture to the arm triangles, hand texture to hand triangles.

Since more and more 3D data available these days, we try to apply data-driven methods to mesh segmentation problem. We use CNN and SVM in our project.

We build our CNN model using caffe and use softmax loss function and stochastic gradient descent algorithm to optimize our result. We use human models from Princeton Segmentation Benchmark to train our data. Our model performs quite well. We got 93% accuracy when training on 18 meshes, 89% on 10 meshes, 82% on 5 meshes.

We also apply SVM to classify our data to different 8 classes , exactly as we have done with CNN. The goal of using SVM is to compare our method to baseline method. We used different kinds of kernel with different values of parameters As our data is large scale, nonlinear kernel of SVN was not suitable. Instead , we applied linear SVM. The results was less accurate compared to CNN. CNN performs much better than SVM.

Data extraction and processing challenges and contribution: In our project, we met several challenges. First of all, for feature extraction. Every mesh contains more than ten thousand triangles, some contain twenty of thirty thousands . It takes a long time to extract geometric features for all triangle. Our main problem was the size of the data , it was too time consuming and not suitable for SVM to train large scale data using nonlinear kernel. After spending long time training with nonlinear kernels which took too long time of training, we figured out that it will not work for the complete data. And the maximum we could train was only one mesh with 3 thousands triangles. Instead we used linear SVM to train the whole data , it does not perform so good but worked.

In section 2, we mainly talks about data preprocessing. In section 3, we talks about our CNN model. In section 4, we evaluate our result.

2. PROBLEM STATEMENT

Our main goal is to segment triangle mesh into main meaningful parts of the human body (Please note that to keep the complexity of the project in check we only considered human body. This can be extended to any three dimensional object). We do that by classifying the mesh triangles to different pre-determined classes. The database of 3D models provide different shapes. We chose to work on Human Body shape. We aim to classify each triangle of test mesh to one of 8 different classes that represent the main parts of human body, upper_leg, loer_leg, upper_arm , lower_arm , torso , foot , head and hand. The first step of our project is extracting the geometric features of the data set. We used recent researches that gave good explanation of what are the geometric features we should use. There are 7 types of features we need to extract of each triangle in each mesh in our training set. All triangles are previously labeled to one of 8 classes we mentioned before.

The main challenges of this project is manifold. Our data is just three coordinates of mesh triangles. We must get a pool of features from these coordinates that can be used as feature vector. We already know that CNNs work really well with image classification. Our aim is to test how CNN stands up to 3D mesh labelling and if it is at all better than less expensive methods like SVM. There is no pre-built CNN models for this purpose and we need to build and train our own model..

Related work:

In the past several years, since not too many 3d model data is available, especially 3d models with every triangle labelled manually, most of methods focus on partitioning mesh into patches with geometric constraints and topological constraints. There are mainly two kinds of algorithms, one is called part-type segmentation and the other one surface-type segmentation[1]. There are some commonly used methods:

1. Region growing,
2. Hierarchical clustering,
3. Iterative clustering,
4. Spectral analysis and
5. Implicit methods.

A variety of geometric features have been investigated, but no single feature or collection of features is known to produce high quality results for all classes of shapes. That's the main problem of all these methods.

With more and more 3d dataset available online, data-driven methods become popular these days. Learning 3D mesh segmentation and labeling is the first paper which applies data-driven method to solve mesh segmentation problem. In this paper, they use CRF model and Jointboost classifier to do the classification, and the accuracy outperforms than most of previous methods, and also it can be applied to various classes of models. Also, there are some other paper which use semi-supervised learning algorithm, like [3].

Deep learning method works quite well when dealing with large-scale data problem. Also, in computer vision, people use CNN to solve image segmentation problem. [4] uses CNN to add semantic annotation to every pixel of one image. Since Mesh segmentation is kind of similar to image segmentation, we apply CNN to semantically segment triangle mesh.

3. METHODOLOGY

Data Processing:

We extract 7 kinds of geometric features of every triangle of all the meshes we have. Those feature represent the characteristics of each triangle geometrically and topologically. Our features are : curvature, PCA, shape diameter, distance from medial surface, average geodesic distance, shape contexts, spin images. The next figure explains the procedure.

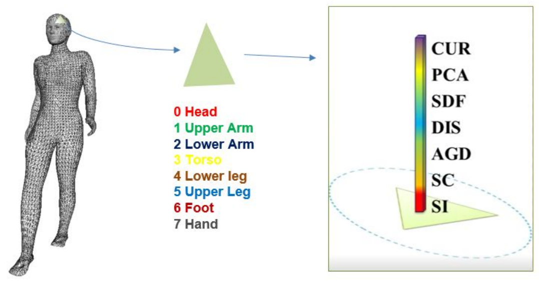

Figure 1, feature extracted from mesh

CNN:

First, we get 600 dimensional feature vector for each triangle. If we classify our data in such a high dimensional space, it may cause some problems, like overfitting. Also, if we wanna use convolutional neural network, it is better that our input looks like an image. So we reshape our feature vector to a 30*20 feature matrix. We also try other size, such as 20*30, 10*60, the result didn't change too much.

Second, We apply twelve 7*5 kernels to the feature matrix first, and get 12 24*16 feature maps. Then do downsampling to all these feature maps, get 12 12*8 feature maps. Then we apply 24 5*5 kernels to get 24 8*4 feature maps, after another downsampling, we obtain 24 4*2 feature maps. Then we concatenate all these feature maps to get a 192-dimensional feature vector. Then apply a fully-connected layer to get our final result. The whole pipeline is as follows.

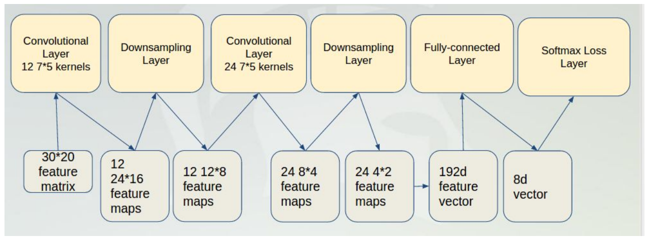

Figure 2, our CNN model

Third, We use Softmax Loss function in our work. We use Stochastic Gradient Descent Algorithm to optimize the loss function.

Softmax Loss Function:

$$\sigma_i(z) = \frac{\exp(z_i)}{\sum_{j=1}^{m} \exp(z_j)}, \quad i = 1, \ldots, m$$

$$\ell(y, o) = -\log(o_y)$$

$$\tilde{\ell}(y, z) = -\log\left(\frac{e^{z_y}}{\sum_{j=1}^{m} e^{z_j}}\right) = \log\left(\sum_{j=1}^{m} e^{z_j}\right) - z_y$$

So we try to minimize the loss function to get the best result.

Stochastic Gradient Descent Algorithm: is a stochastic approximation of the gradient descent optimization method for minimizing an objective function.

SGD updates the weights W by a linear combination of the negative gradient ∇L(W) and the previous weight update Vt. The learning rate α is the weight of the negative gradient. The momentum μ is the weight of the previous update.

Formally, we have the following formulas to compute the update value Vt+1 and the updated weights Wt+1 at iteration t+1, given the previous weight update Vt and current weights Wt:

Vt+1=μVt−α∇L(Wt)Vt+1=μVt−α∇L(Wt)

Wt+1=Wt + Vt+1

Model Parameter: we initialize all the w with small random values generated by Gaussian function. The learning rate of our gradient algorithm is 0.1, momentum is 0.9, and the parameter for L2 norm regularization is 0.004.

SVM:

We applied multi-class SVM to our data. First , we tried to use nonlinear kernel with different values of error parameter C, we used RBF kernel. The training was time consuming , we could not get training model for any data more than 4000 instances. For RBF kernel we used libSVM library[5] . So , we used linear SVM using LIBLINEAR library[6], this library is designed to handle large scale data. Each time we trained our data using SVM we get the 3 values. first, accuracy of the prediction second, vector of probabilities , represents the probability of this triangle to be belonged to any of the 8 classes. Third , the sum squared error.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

1. Characteristics of the data set:

we use 'Princeton Segment Benchmark' as our source of Data. It contains 19 object categories, such as human, plane, cup, fish, ,chair, etc. Each model has many different segmentations. In other words, there are different numbers of classes of each model based on the meaningful parts of the shape.

we chose 'Human' model as our model to work on. 'Human' model has 11 different segmentations, we chose the one that segments the shape (the body) to 7 parts: head, hand, torso, lower leg, upper leg, lower arm and upper arm. we have 20 'human' meshes with different positions of the body parts.

Data Size: 20 meshes, each mesh has almost ten thousand triangles. Each mesh is stored in mesh file, which contains vertex positions of each triangle in this mesh. After data preprocessing, we get a 600 dimensional feature vector for every triangle.

2. Evaluation measures you have used : Accuracy
3. Computing platform: Linux on intel7, windows , Decs servers.
4. Software used to generate the results: matlab. Caffe, libsvm

4.2 Experimental Results

1. The output of CNN

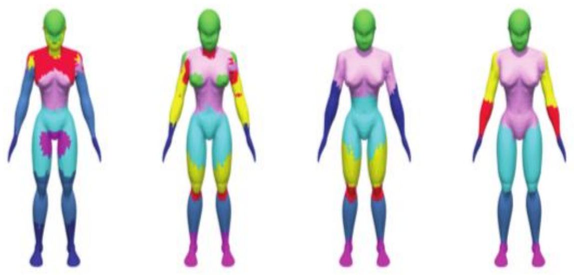

Figure 3, These figure can be produced using the output and using OPENGL library in C++. The first one is k-means cluster result on the original feature vector. It can be clearly seen that the segmentation is far from accurate with parts like thorax, arms, upper legs etc. segmented into random portions that do not correspond with actual desired human anatomy. The second one is the k-means clustering result on the output of the first convolutional layer, the third one is the kmean result on the output of the second convolutional layer, the last one is our final output. As we progress, the merging of segmented areas with corresponding human anatomical regions becomes evident. Linear Svm gives accuracy of 45% on 5 meshes and 50% of 10 meshes.using cross validation , the accuracy increases 2% to 3 % only.

2. The training curve

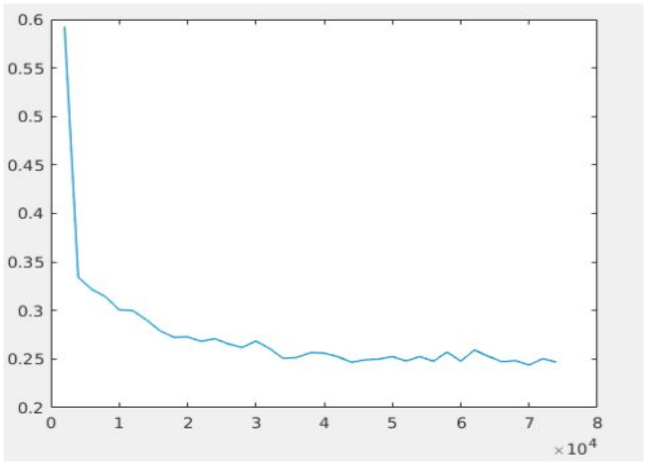

Figure 4 training curve on 5 meshes, x-axis is the iteration number, y- axis is the training error.

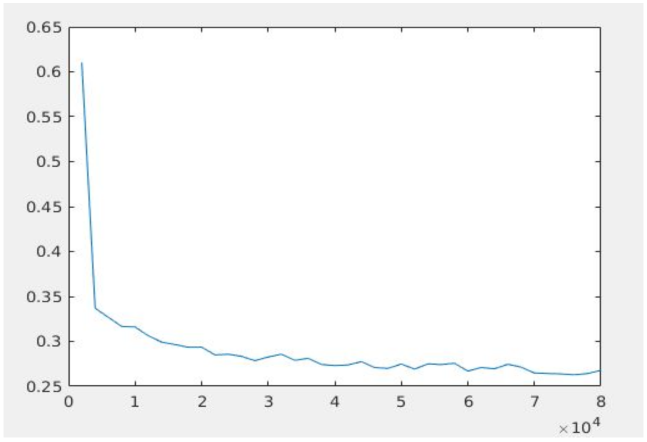

Figure 5 training curve on 10 meshes, x-axis is the iteration number, y- axis is the training error.

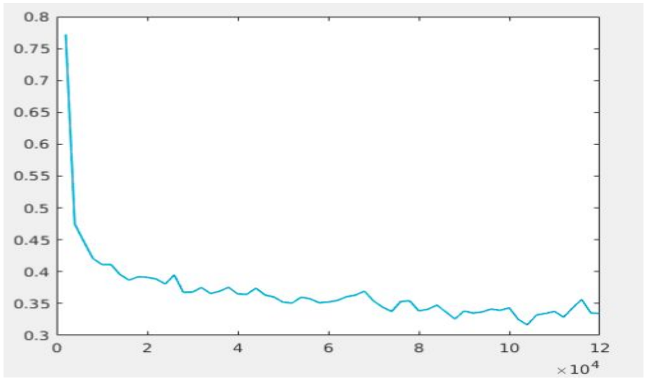

Figure 6 training curve on 18 meshes, x-axis is the iteration number, y- axis is the training error.

3. Comparison

Training dataset	5 meshes	10 meshe s	18 meshes
CNN Accuracy	83%	89%	93%
SVM Accuracy	45%	50%	-

4.3 Discussions

We used k-means as the standard clustering algorithm at every stage of the output of feature set from CNN. As was evident from the result set that keeping the clustering algorithm constant, accuracy increased significantly at every step. Thus it can be safely inferred that CNN was quite successful in extracting the most relevant features from the feature set. Due to time constraints we could not investigate exactly what types of features were given precedence in the selection process and why so. A study into this area would lead to more precise results in any future work done in this field.

In our results, one epoch corresponds to 5000 iterations when we train on 18 meshes. Varying data size changes epoch and iterations correspondingly according to the equation :

*Epoch = Batch Size * Iteration Number*

As evinced, error rates experience a steep fall with each epoch and then smoothen out thereafter. One important observation was that the decrease in error rate was smoother when 10 meshes were used for training over 18 meshes.

Like any other learning algorithm the results clearly show a direct relation between data set size and performance. More data gifts better accuracy. Comparatively our project was based on very small amount of data. This was necessitated by the fact that computing power at our disposal was scant and due to time and scope constraint we did not venture to parallelize our processing.

All code we have written or libraries we used are uploaded to Github, with each code file , there is README file of how to use. A major part of the project involved setting up of the libraries like OPENGL[8], caffe[7] etc which will be necessary while running the code.. Matlab is used as the primary programming interface , and data extraction/processing code is in C++.

The github link is: https://github.com/lovelylightningbug/CSE881_project.git

5. CONCLUSIONS

After our investigating our project results , we can conclude that CNN is the outperforming method of mesh segmentation problems. It gives better results than methods like SVM. This plainly points to the fact that feature selection is possibly the most important aspect of mesh segmentation. We also can say that increasing the number of training meshes gives better results like any other learning problems.. Since our data was highly dependent on the geometric features of shape mesh, training more complicated combinations of position and direction will provide more information about the different parts of the mesh and may possibly make better features. This can be an active research area..

6. Future Work

We just use unary features in our project. In the future, we can apply pairwise features to improve result, such as dihedral angles, shape diameter difference, Distance from medial surface differences. if the dihedral angle between two triangles are close to pi, the probability is higher that they have the same label.

We can also modify our neural network to get better result. RNN is used for sequence data, since there are relations between the labels of the neighboring triangles, we can also add RNN to our model to see whether we can get higher accuracy.

7. REFERENCES (at least 5 references)

[1] Shamir, Ariel. "A survey on mesh segmentation techniques." Computer graphics forum. Vol. 27. No. 6. Blackwell Publishing Ltd, 2008.

[2] Kalogerakis, Evangelos, Aaron Hertzmann, and Karan Singh. "Learning 3D mesh segmentation and labeling." ACM Transactions on Graphics (TOG)29.4 (2010): 102.

[3] Lv, Jiajun, et al. "Semi supervised Mesh Segmentation and Labeling." Computer Graphics Forum. Vol. 31. No. 7. Blackwell Publishing Ltd, 2012.

[4] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[5] Multiclass SVM , https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[6] LIBLINEAR, http://www.csie.ntu.edu.tw/~cjlin/liblinear/

[7] CAFFE, http://caffe.berkeleyvision.org/

[8] OpenGl, https://www.opengl.org/