Mesh Segmentation and Labeling: A Data Driven Approach

Rohit Rao Padebettu
Dept. of Computer Science
111447392
rraopadebett@cs.stonybrook.edu

***Abstract*— Mesh labeling for a 3D mesh is a process of assigning a part object label for every face of the mesh. This paper details few data driven approaches to automatically segment and label a 3D mesh with minimal or no human intervention. The machine learning algorithms are trained with the help of an input feature vector, which is a set of descriptive features which can compactly represent the mesh. This paper also gives an overview of various algorithms used to effectively generate a robust description of the given 3D mesh.**

***Key Words* – 3D Mesh Segmentation, 3D Mesh labeling, 3D Mesh features.**

I Introduction

One of the fundamental problems of shape understanding and processing is segmentation of the shape. Many of the problems require labelled segmentations where parts of the mesh are mapped to identified parts. Segmentation of 3D meshes has extensive applications in mesh editing, deformation, modelling, manufacturing, animation and many other fields.

Manually labeling a mesh can be labour and time intensive. This serves a motivation to devise methods which automate this process. One of the approach to solve the problem is utilizing a specially designed geometric feature of a mesh face and using this signature to match faces having similar signature E.g. Using Shape diameter function as a feature to match the faces. However, these types of geometric features usually work only for limited types of 3D meshes. Alternatively, a data driven approach can be taken, where in a classifier is trained to segment the mesh into its constituent segments.

I.a Overview

This paper details the key concepts of five papers. A brief overview of the papers are as follows. The paper [Kalogerakis et al] presented a data driven supervised learning approach for mesh segmentation. The author models the problem of simultaneous segmentation and labeling as a conditional random field (CRF). It introduces unary and pairwise terms to accurately classify a mesh including faces at the border of mesh segments. However, this paper assumes a fully labeled mesh to train a Joint boost classifier.

Paper two [Lv et al] introduces idea of semi-supervised learning to handle mislabeled and complex meshes. This paper describes a method to robustly handle inconsistently labeled mesh by making use of results from both supervised and unsupervised learning techniques. It achieves this by building a semi-supervised mesh segmentation model using virtual evidence boosting.

Paper three [Benhabiles et al] focuses on producing smooth closed boundaries on the 3D mesh. An adaboost classifier is used to optimize a boundary function during an off-line step. During the on-line step, the edge function is used to select a set of candidate boundary contours, to close them and to optimize them using a snake movement to produce the final segmentation

Previously mentioned techniques take approximately eight hours to train a classifier on meshes of size 20k-30K faces. Paper four [Xie et al] reduces the training time on the sample meshes by factor of two by using extreme learning machine.

Paper five [GUO et al] outlines a convolution neural network based deep learning architecture for 3D mesh feature generation and mesh labeling. Utilizing the convolution and subsampling kernels in the CNN architecture, the size of feature vector is reduced from to 192 from 600 features. Moreover, a reduced feature set is less likely to overfit the classifier.

Finally, this paper provides an overview for a set of algorithms to used extract features out of a 3D mesh. Features such as curvature, average geodesic distance, shape diameter function, and volumetric shape images are crucial in training the classifiers and label prediction process.

II) Learning 3D Mesh Segmentation and Labeling

II.a) Objective function in a conditional random field model

The objective of mesh labeling can be defined as assigning a label $l \in C$, where C is a predefined set of possible labels to every face mesh label i. Every mesh face i can be described using a set of local surface geometry features (x_i) and context based features such as curvature, shape diameter and shape context. Moreover, in-order to capture the relationship between adjacent features a pairwise term($y_{i,j}$) is also introduced. This include features like dihedral angle between pair of mesh faces.

Computing all mesh labels involves minimizing the following objective function.

$$\mathbf{E(c;\theta) = \Sigma_i\, a_i E_1(c_i; x_i, \theta_i) + \Sigma_{i,j}\, l_{i,j} E_2(c_i c_j; y_{ij}, \theta_2)}$$

Here E_1 denotes the unary term for a mesh face i having the features x_i and label c_i. Similarly E_2 denotes the pairwise term for face pair of i,j and labels c_i, c_j. Additionally, the terms are weighted by the area of the face a_i and edge length $l_{i,j}$.

In CRF, probability of labeling a mesh label is conditionally defined as:

$$\boldsymbol{P(C|x,y,\theta) = \exp(-E(c,\theta))\,/Z(x,y,\theta))}$$

E denotes the objective function and Z is the normalizing function. The optimized solution to the objective function is evaluated using a joint boost classifier.

II.b) Unary term.

Unary term E_1 evaluates a joint boost classifier, it evaluates probability of a label c_i depending on the feature vector as an input. The energy of the unary term is equal to negative log of the probability.

$$\boldsymbol{E_1(c;x,\theta) = -\log P(c|x,\theta_1)}$$

Unary classifier alone gives good results in part interiors but performs poorly near the boundaries.

II.c) Pairwise term.

Pairwise term is added to improve segmentation at the boundaries i.e. it penalizes neighbouring faces being assigned different labels c. It consists of label compatibility term L weighted by geometry dependent term G.

$$\boldsymbol{E_2(c,c';y,\theta_2) = L(c,c')\;G(y)}$$

The consistency between two adjacent labels is measured by a label-compatibility term $\boldsymbol{L(c,c')}$. This term is modelled as a matrix of penalties for each possible pair of labels, where in different pairs of labels to incur different penalties.

Geometry dependent term $\boldsymbol{G(y)}$ evaluates a classifier as a measure of likelihood of difference in label as a function of geometry.

$$\boldsymbol{G(y) = -\kappa\log P(c \neq c'|y,\varepsilon) - \lambda\log\left(1 - \min\left(\frac{\omega}{\pi},1\right) + \epsilon\right) + \mu}$$

$\boldsymbol{P(c \neq c'|y,\varepsilon)}$ evaluates the probability of two adjacent faces having distinct labels. The second term penalizes boundaries between faces with high exterior dihedral angle $\boldsymbol{\omega}$, penalty $\boldsymbol{\mu}$ *is* added on boundary length to prevent jaggy boundaries. A small constant $\boldsymbol{\epsilon}$ is added to avoid computing log 0.

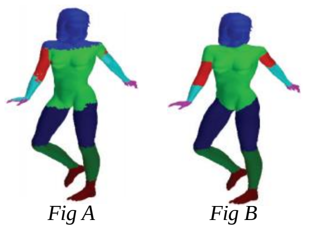


Fig A only uses unary term for segmentation, Fig B uses both unary and pairwise terms for segmentation.

II.d) Joint Boost classifier.

A joint boost classifier is used to evaluate both the unary term and geometry dependent term of pairwise term. A joint boost classifier consists of decision stumps which outputs a score for class l, given a feature vector z.

$$\boldsymbol{h(z,l,\emptyset) = \begin{Bmatrix} a & z_f > \tau & l \in C_s \\ b & z_f < \tau & l \in C_s \\ & k_l\; l & \notin C_s \end{Bmatrix}}$$

The probability of the class label is computing the soft-max transformation on summation on the decision stump values.

$$\boldsymbol{H(z,l) = \sum_j h(z,l;\emptyset_j)}$$

$$\boldsymbol{P(c = l|z,\xi) = \exp(H(z,l))\,/\sum \exp(H(z,l))_{l\in C}}$$

Further, in-order to reduce the error between the prediction and ground truth, a segmented weighed error is introduced which is weighed based on A_{c_i} is the total area of all the faces segments having label C_i

$$\boldsymbol{E_S = \sum\left(\frac{a_i}{A_i}\right)((I(c_i;c_i^*) + 1)/2}$$

III) Semi-supervised mesh segmentation and labeling

Supervised mesh segmentation delivers excellent results when its trained using fully labelled meshes, where as an unsupervised segmentation does not require a labeled data set but performs poorly when compared to supervised segmentation. In this paper [Lv et al], author highlights the abundance of unlabeled meshes when compared to fully labeled meshes. Additionally, author also notes that manually labeled meshes are inconsistent and often mislabeled, this is especially true when the meshes are complex. Hence, the author combines advantages of both the method by augmenting the existing CRF model (given below) with additional objective function which considers unlabeled mesh along with labeled mesh.

$$\boldsymbol{P(Y|X,\omega) = \frac{exp(F(X,Y) + G(Y,\omega))}{Z(X,Y,\omega)}}$$

F(X,Y) denotes the unary energy term of mesh face and the **G(y,ω)** denotes the pairwise energy term for a pair faces in the mesh.

III.a) Semi-supervised Objective Function

This idea is driven by intuition that minimizing the conditional energy of the objective function using the unlabeled mesh faces helps to discover labels for the unlabeled mesh faces, which is supported by a similar labeled mesh face I.e. greater the likelihood on the supervised mesh implies greater certainty of labeling of the unlabeled mesh.

Given, labeled mesh set $\boldsymbol{D^l} = \{(\boldsymbol{x^1}, \boldsymbol{y^1}), \ldots, (\boldsymbol{x^N}, \boldsymbol{y^N})\}$ and unlabeled mesh set $\boldsymbol{D^u} = \{(\boldsymbol{x^{N+1}}, \boldsymbol{y^{N+1}}), \ldots, (\boldsymbol{x^M}, \boldsymbol{y^M})\}$, then the objective of the semi supervised CRF is given by

$$\sum_{i=1}^{N} \log P(Y^i|X^i) + \gamma \sum_{i=N+1}^{M} \sum_y P(Y|X^i) \log P(Y|X^i)$$

The first term considers summation of log likelihood of labeled meshes and the second term is negative conditional entropy of the CRF on the unlabeled meshes. The entropy of the unlabeled meshes is controlled by the parameter γ.

The CRF is evaluated using Virtual evidence boosting (VEB)

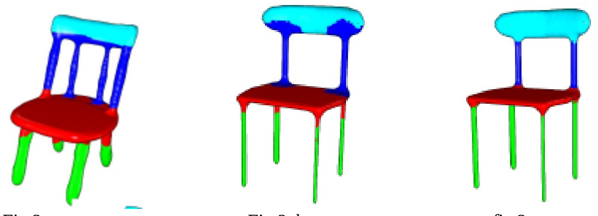


Fig 2.a shows a mislabelled training data, fig 2.b is segmentation obtained from simple supervised learning. Fig 2.c is obtained from semi supervised learning.

III.b) Virtual Evidence Boosting.

Virtual evidence boosting consists of two iteratively running components. First, Belief propagation which is used to derive probabilistic labels of the mesh face(nodes) simultaneous with pairwise label relationship which gives us the mesh segments.

Second, Logit boost to propagate the information from the immediate neighborhood

Belief propagation.

A face in a VEB is labeled depending on two evidences. First is the feature of the mesh and second is the message from the neighborhood. The message is obtained by running belief propagation with the current functions F(X, Y) and G(Y,ω).

The likelihood of adjacent faces having different labels is proportional to angle of the face normal. Hence, the pairwise term is G(y_{s1}, y_{s2}) is divided by $\rho(2 - \ cos\theta)$

$$\boldsymbol{\lambda_{s1\to s2}(y_{s2}) = \ \alpha e^{F(y_{s2},x_{s2})}\ {}_{s3\in n(y_{s2})}\pi\ \mu_{s3\to s2}(y_{s2})}$$

And

$$\boldsymbol{\mu_{s2\to s1}(y_{s1}) = \ \beta \textstyle\sum_{y_2} e^{G(y_{s2},y_{s1})/\rho(2-cos\theta)}\ \lambda_{s2\to s1}(y_{s2})}$$

Hence the belief propagation can be written as

$$\boldsymbol{P(y_{s1}, X) = \ \aleph e^{F(y_{s2},x_{s2})}\ {}_{s3\in n(y_{s2})}\pi\ \mu_{s3\to s2}(y_{s2})}$$

Where $\aleph$ is used for normalization.

Logit Boosting

Logit boosting performs the task of fast sequential learning and automatic feature section. For given set of labels J there are F_j functions which determines the probability of vertex $x_s$to take the label j. These functions are separately defined for both unary and pairwise terms as follows

$$\mathbf{P_j(X_s) = \ e^{F_j(X_s)}/(\textstyle\sum_{k=1}^{J} e^{F_k(X_s)})}$$

For the pairwise term measure the consistency between the neighboring labels j

$$\boldsymbol{G_j(j^*) = \textstyle\sum_{k=1}^{J} \alpha_{j,k}(k = j^*)}$$

As these functions are executed iteratively, the weights (w_s) and their responses (d_s) are updated each cycle. Moreover, there weighing of unlabeled face in conditionally dependent on the weights of labeled face after belief propagation

Weights for labeled unary term

$$\mathbf{w_s = \ p_j(x_s)(1 - p_j(x_s))}$$

Weights for unlabeled unary term

$$\mathbf{w_s = \ \alpha P_j(x_j)(1 - logP_j(x_j) + (1 - 2logP_J(x_j))}$$
$$\mathbf{\textstyle\sum_j P(j|x_s) \log P(j|x_s, y_s) - logP_j(x_s)}$$

Here, $P_j(x_s)$ denoted the probability of face s taking thing the label j.

IV) Learning boundary edges for 3D-mesh segmentation

While mesh labeling implicitly segments the mesh by assigning labels to each face i.e. identifies compact regions which can be assigned a common label. The boundaries generated from these methods are not explicitly defined. As the boundaries are not well defined, the boundaries are not thin, well defined closed contours.

The primary motivation of the Author [Benhabiles et al] is to identify thin, well defined closed contours using a boundary function learnt using Adaboost classifier. They further use the generated feature vector to further optimize the boundary with a region thinning, contour completion and snake moment algorithms. Complex boundaries can be captured using this method without having to train the classifier on a specific category of mesh.

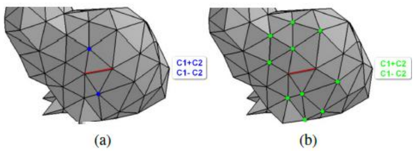
The algorithm consists of two phases, offline learning phase and online classification and boundary improvement phase.

IV.a) Off-line Learning phase

The author formulates the problem of learning boundary edges as a classification problem. Where a classifier, in this case AdaBoost classifier learns to outputs a signed scaler value depending on a set of feature vector F_e.

IV.a.i) Feature selection.

The classifier is trained using mesh surface and volume features which comprises of dihedral angle, curvature, global curvature and shape diameter. Each of the features is amalgamated into a single feature vector. The classifier uses these feature to allocate a class label, boundary or not boundary. It outputs label L = +1 if the edge is a boundary and L = -1 otherwise. In order introduce robustness in the edge feature, the features is not only computed using the end points of the edge but also considering 1-ring neighborhood surrounding the edge



edge criterion computation with one vertex and with a set of vertices

IV.a.ii) Ada Boost Classifier

Ada boost classifier generates a strong classifier **H** by linearly combining set of weak classifiers **h**.

$$\boldsymbol{H(x)} = \sum_{t=1}^{T} \boldsymbol{\alpha_t h_t(x)}$$

Considering a set inputs $(x_1, y_1), \ldots, (x_n, y_n)$ where, x denotes the geometric features and y denotes if it's a boundary or not. Using these inputs large number of weak classifiers are generated $h_i: X \rightarrow Y$. These weak classifiers are filtered iteratively based on classification error. Author also notes that other classification algorithms such as SVM and HME also generate similar results.

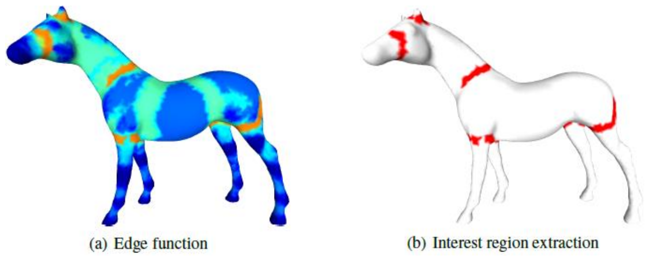
IV.b) On-line classification.

Using the trained classifier, the mesh edges are classified as boundary or not. However, the generated boundaries are fuzzy and unconnected regions. The result is as expected as the classifier learns from different categories of objects and on multiple ground truths.

IV.c) Boundary optimization.

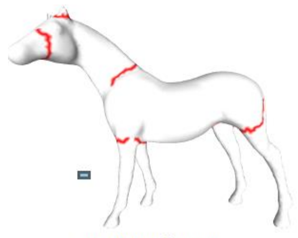
Generating thin, closed contours from fuzzy regions obtained from the classifier involves 4 steps.

IV.c.a) Interest region extraction



Classifier generates binary label L = +1 for a boundary and L = -1 otherwise. Then all edges having positive function values are filtered. Theses edges constitute a set of interest regions as seen in figure (b). Then, for each interest region, a thinning algorithm is applied.

IV.c.b) Region thinning


(c) Region thinning

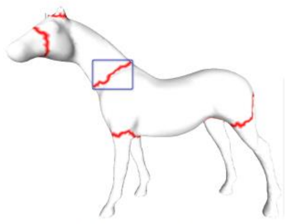
The region of interest extracted from previous phase consists of set of edges. The following algorithm is used to generate piecewise linear contour. The algorithm defines border edge as edge of which at least one of the four edges of its two opposites triangles do not belong to the interest region.

1) Initialization step:
   Insert all the border edges (b_1, b_2,…, b_n) present in the region of interest R into list.
2) Delete border edge $b_m \in$ R
3) Check if region R is disconnected, if yes restore border edge.
4) Check for new border edge, if found add to list
5) If list is empty stop iteration, else go to step 2.

The region thinning algorithm can produce branching skeleton in case the interest region is large. A branch skeleton consists of internal and external branches. An external branch is limited by one endpoint and one junction point while an internal branch is limited by two junction points. Only two external branches are considered as real boundary and other branches are relabelled as non-borders.

IV.c.c) Contour Completion

In this phase, the open contours produced by region thinning algorithm, is completed to produce a closed boundary by finding the shortest weighted path between two end points of the contour.


(d) Contour completion

The weight of an edge is composite of three functions, distance function $\boldsymbol{\eta_d}$, normal function $\boldsymbol{\eta_n}$ and feature function $\boldsymbol{\eta_e}$.

The cost of an edge e in the open contour is defined as

$$\boldsymbol{cost(e)} = \boldsymbol{\eta_d}(e)^{w_d}.\boldsymbol{\eta_n}(e)^{w_n}.\boldsymbol{\eta_e}(e)^{w_e}$$

The distance function measures the distance of the vertices to the contour. Hence, the distance function for a vertices v of the edges is given by

$$\boldsymbol{\eta_d(V)} = {}_{v_i \in contour}\sum 1/d(v, v_i)$$

$d(v, v_i)$ is the Euclidian distance between the two vertices. If the vertex is close contour it generates a high value and low value otherwise.

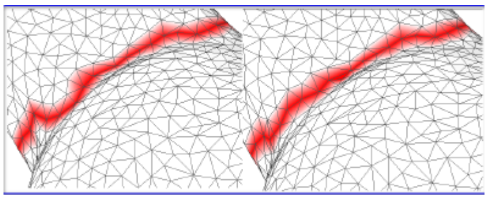
The normal functions consider the angle between vertex normal and contour normal.

$$\boldsymbol{\eta_n} = \begin{cases} \mathbf{1} & \text{if } (\boldsymbol{n_\lambda}.\boldsymbol{n_v}) > \boldsymbol{cos(\alpha)} \\ (\boldsymbol{n_\lambda}.\boldsymbol{n_v} + \mathbf{1})/\boldsymbol{cos(\alpha)} + \mathbf{1} & \boldsymbol{otherwise} \end{cases}$$

Here, n_λ is the average contour normal, n_v is the vertex normal of v and α is the angle between normal of two endpoints of contours. Feature function $\boldsymbol{\eta_e}$ is guided by edge function learnt using the classifier.

The boundaries generate are closed but are not smooth.

IV.c.d) Contour optimization using snake movement.


Contour smoothing

The algorithm iteratively minimises the energy which consists of internal and external parts. The internal energy controls the length and smoothness of the snake, whereas the external energy is dependent on the edge feature function. Here the Author uses the feature function generated by the classifier. The energy of a given vertex is given by

$$\mathbf{E_{int}(v_i)} = \boldsymbol{\alpha}||\mathbf{v_i} - \mathbf{v_{i-1}}|| + \boldsymbol{\beta}||\mathbf{v_{i+1}} - \mathbf{2v_i} + \mathbf{v_{i-1}}||$$

The values of α is set to 0.2 and β is set to 0.8

IV) 3D Shape Segmentation and Labeling via Extreme Learning Machine.

While the mesh labeling is highly accurate, the training time to train the models for a given feature vector can be very high. The goal of the author [Xie et al] is to reduce the training time without compromising on the accuracy of the predicted labels. Hence, the author proposes to use extreme learning machine (ELM) as classifier. Furthermore, the proposed solution scales well on large datasets and can be used for online learning.

The paper performs mesh segmentation in the following phases:

1) ***Training phase***: A extreme learning classifier is trained with the help of mesh features.
2) ***Testing phase***: A test mesh is labelled using the trained classifier.
3) ***Segmentation optimization***: the labels at the boundaries are smoothed to have minimum energy using graph cut algorithm.

V.a) Extreme leaning machine overview, training and classification

V.a.a) Overview

A single-hidden layer feedforward network (SLFN) with L hidden nodes is represented as

$$\sum_{i=1}^{L}\boldsymbol{\beta_i G(a_i, b_i, x_j) = o_j}$$

Assuming the network can predict the labels without any error for given parameters β_i, a_i, b_i. This can be written as

$$\sum_{i=1}^{L}\boldsymbol{\beta_i G(a_i, b_i, x_j) = t_j}$$

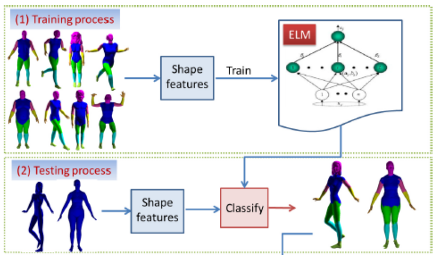
For N samples, this can be written as

$$\boldsymbol{H\beta = T}$$

Finding the weights of the network i.e. training the classifier is equivalent to finding the solution to β

$$\boldsymbol{\beta' = H'T}$$

V.a.b) Training and Classification.

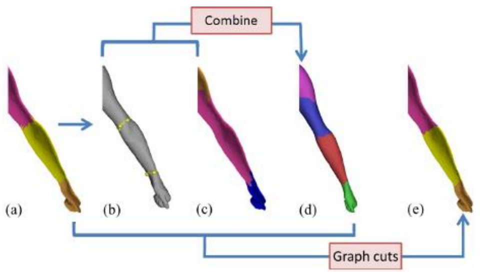


Mesh features are extracted and normalized as suggested in Kalogerakis et al. Then the parameters, such as the number of hidden neurons, the number of training meshes and the activation function, are chosen for the classifier.

The ELM is trained using the uniformly weighted features. In the testing stage, for a given test meshes, every face of the mesh is classified into a specific class by the ELM classifier.

V.b) Boundary Optimization.

Labels obtained from classification procedure are noisy at the boundaries. The refining the boundaries into smooth and well-defined borders is done is two steps.



V.b.a) Boundary extraction.

i) Boundary extraction using over-segmentation ($B_{overseg}$)

The 3D mesh is decomposed to primitive patches using normalized cuts. The patches are then aligned using fuzzy cut algorithm. The alignment is done based on shape features.

ii) Boundary extraction using mesh contours (B_{ELM})

Smooth boundaries are extracted from ELM segmentation by contour completion as described in [Benhabiles et al].

iii) Boundary fusion

Boundary detected from above two methods are combined by taking the union of two sets of edges. This new edge set results in a finer segmentation.

$$\boldsymbol{B_{comb}} = \boldsymbol{B_{overseg}} \cup \boldsymbol{B_{ELM}}$$

V.b.b) Graph Cut optimization.

Consider the given mesh as a graph, where nodes are the mesh face and the arc represents the adjacency between the faces. From the classifier, the face would be assigned a label l with a probability P_l. Additionally, $\boldsymbol{B_{comb}}$ value is used to weigh the edges.

Finding distinct boundary is equivalent to segmenting the graph into its components by removing weakly weighted edges.
The graph can be segmented using mincut-maxflow algorithm.

Segmentation of the graph can also be represented energy minimization problem E(l), where E_D and E_l are the data and smoothness energy terms respectively. λ is a tuning parameter and is set to 50.

$$\boldsymbol{E(l)} = {}_{l\in V}\sum \boldsymbol{E_D(l, p_l)} + \boldsymbol{\lambda} {}_{l,k\in E}\sum \boldsymbol{E_s(u, v, l_u, l_v)}$$

The energy value of data energy term

$$\boldsymbol{E_D(u, l_u)} = -\boldsymbol{log(P(l_u|u))}$$

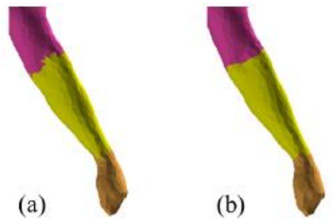
$P(l_u|u)$ is the probability the node u is assigned the label l_u.

Similarly, the smoothness term is given by

$$\boldsymbol{E_s(u, v, l_u, l_v)} = \begin{Bmatrix} \boldsymbol{0} & \boldsymbol{if\ lu\ =\ lv} \\ \boldsymbol{\omega_{uv} l_{uv} (1 - log(\theta_{uv}/\pi))} & \boldsymbol{otherwise} \end{Bmatrix}$$

Here, l_{uv} is the edge length and θ_{uv} is the dihedral angle between the faces. ω_{uv} is used favour graph cuts across the boundaries.

$$\boldsymbol{\omega_{uv}} = \begin{Bmatrix} \boldsymbol{10} & \boldsymbol{e_{uv}\ \epsilon\ B_{comb}} \\ \boldsymbol{1} & \boldsymbol{otherwise} \end{Bmatrix}$$



(a) Initial labeling resulted from ELM (b) optimized segmentation from graph cuts.

VI) 3D Mesh Labeling via Deep Convolutional Neural Networks

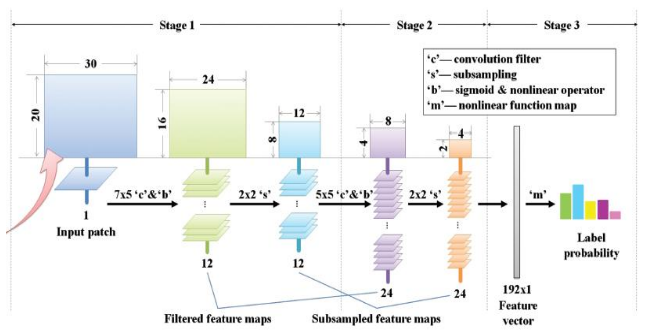
Mesh feature representation generated from the CRF and ELM method seen in previous sections are relatively simple and might not optimally label complex meshs. Moreover, the dimensions of the feature vector remain unaffected. Author also mentions that large, linearly combined feature vectors tend to overfit the training set.

To overcome the above challenges, author presents a deep convolution neural network to generate compact representation of the input feature vector by nonlinearly combining and hierarchically compressing various geometry features. Optimized feature vector is used to label the mesh faces.

Mesh features like curvature, PCA feature, shape diameter function, distance from medial surface, average geodesic distance, shape context, and spin image is extracted per mesh face. Author reorganizes these features into a CNN compatible feature matrix of size 30x20.

VI.a Convolution neural networks structure

The architecture of the CNNs with consists three stages. Each of the first two stages apply convolution, nonlinear function mapping and subsampling functions. The last stage consists of nonlinear function map which is used to obtain the label probability of each triangle on the mesh.

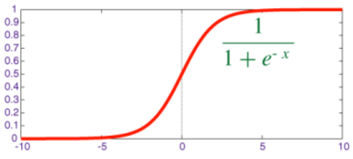


In the first stage, twelve 7 $\times$ 5 convolution kernels $\{w_i\}_{i=1}^{12}$ and bias values $\{b_i\}_{i=1}^{12}$are applied to feature matrix. The convolution operation (*) can be represented as

$$\boldsymbol{Y_i = W_i * X + b_i \; i} \in [\mathbf{1, 12}]$$

This is followed by non-linear sigmoid function module used to introduce nonlinearity in the features.

$$\boldsymbol{\varphi(x) = (1 + \exp(-x))^{-1}}$$

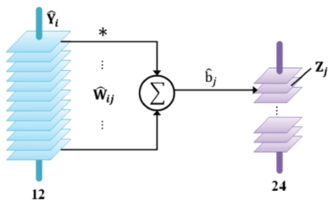


The non-linearly mapped feature is down sampled by a factor of 2 to generate twelve 12 $\times$ 8 feature maps $\{Y_i'\}_{i=1}^{12}$.

The stage two similarly generates 24 feature maps of size 4x2 by applying convolution, nonlinearity and down sampling functions.

$$\boldsymbol{Z_i} = \sum_{i=1}^{12} \boldsymbol{W'_{ij} * Y'_i + b'_j , j} \in [\mathbf{1, 24}]$$

The feature map $\boldsymbol{Z_i}$ is unrolled to from a linear vector of size 192 elements.



The label for a mesh face given feature vector v and bias vector b can be represented as

$$\boldsymbol{P} = \emptyset(\boldsymbol{Mv + b})$$

VI.b Training

To obtain accurate label prediction, appropriate values of M weight matrix and bias should be learnt such that the prediction error is minimized. These parameters are learnt by minimizing the following energy function.

$$\boldsymbol{\pi = \arg\min \sum_{t \in T} |g_t - \emptyset(MV_t + b)|^2}$$

The feed forward stage consists of updating the weights of the node by updating the label indication p.

Back propagation phase is used to update/propagate the prediction error and reducing it layer by layer using gradient descent. This update can be represented as

$$\boldsymbol{\pi \leftarrow \pi - \alpha \nabla_\pi \sum_{t \in T} |g_t - \emptyset(MV_t + b)|^2}$$

α: learning parameter

This iteration is executed for a predefined number of iteration or until the function converges to a local minimum.

VI.c Boundary optimization using graph cuts.

The labels at the boundary is smoothed using similar graph cut algorithm introduced in ***V.b.b***. The smoothness term in the energy function is modified as follows

$$\boldsymbol{E_s(u, v, l_u, l_v)} = \begin{cases} \mathbf{0} & \boldsymbol{if\ l_u = l_v} \\ \boldsymbol{-d_{uv} log(\theta_{uv}/\pi)} & \boldsymbol{otherwise} \end{cases}$$

Here, $\boldsymbol{d_{uv}}$ is the geodesic distance between the mesh faces and $\boldsymbol{\theta_{uv}}$ is the dihedral angle between the faces.

Mesh Features Algorithms

Mesh features provide an effective description of the 3D mesh by capturing local and global mesh properties. This representation can be used to train any classifier such as JointBoost, AdaBoost, virtual evidence boosting, ELM, CNN as seen in previous algorithms. The section below is an overview of 3D mesh features.

VIII) Shape Diameter Function.

Shape diameter functions is a poise oblivious scalar feature, which does not change on any rigid body transformation. Any objects volume from surface is defined as the distance from the surface to the medial axis. However, computing the medial axis is complex and error prone.SDF is an effective approximate to object's volume from its surface which is also simple to compute.

Shape diameter function on different points on the mesh

Algorithm to compute the SDF for any point on the surface of the mesh is given as follows

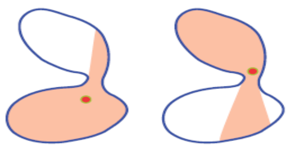
1) Construct cone of a given angle, centered around inward normal. The inward normal is the opposite direction of surface normal.
2) Shoot predefined number of rays into the cone.
3) Compute intersection of the ray and mesh.
4) Filter ray which lie outside one standard deviation from the median of all lengths.
5) Allocate weights inversely proportional to
6) SDF of the point is the weighted average of all the ray lengths.

The ray-mesh intersection can be computed using algorithm like Möller–Trumbore ray - triangle intersection algorithm.

The authors suggest using default values to an opening angle of 120 θ and project 30 rays per point. Using a small cone angle will not create a good discrimination between different object parts and would be too sensitive to local features of the mesh. On the other-hand, using a large opening angle, exposes the SDF measure to noise and errors.

IX) Volumetric Shape Images (VIS).

Volumatic shape image is motivated that change in part mostly corresponds to distinct change in volume.

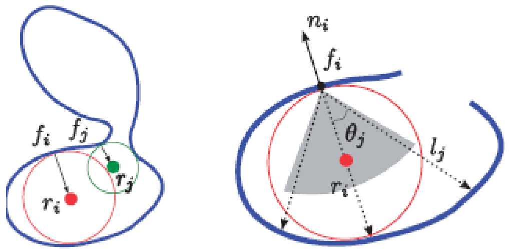


Change in part has change in volume.

Matching full volumes of a model is an expensive process. Hence to capture the shape, the volume is sampled across multiple faces.

The algorithm to compute the VIS of a mesh is given as follows

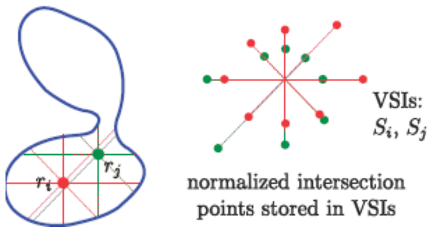
IX.a) Computing the reference point R_j



Computing the VIS using SDF

The volume of the shape can be best captured from the medial centre. However, computing the medial centre can be expensive. Hence, medial centre is approximated using SDF. Using SDF algorithm, the longest ray intersecting the mesh is identified. The mid-point of the ray θ_j can be considered as good approximate to the medial centre.

IX.b) Computing VIS



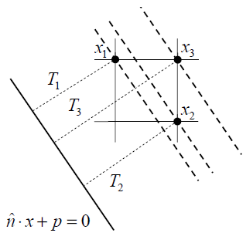
Computing the VIS

With medial centre as a reference point, m rays are sent out from this point onto a uniformly sampled gaussian sphere. The length of ray from the reference point to the intersection point on the gaussian sphere is called the volumetric shape image. A larger *m*, leads to better approximation of visible regions by VSIs, but at a higher complexity. To achieve a good trade-off between accuracy and efficiency, Author sets the value of m =100.

X) Geodesic distance.

Geodesic distance between two points on the surface of the mesh is the shortest distance between without leaving the manifold. The shortest path algorithms like Dijkstra's algorithm do not converge on the geodesic distance but can provide a good approximation.

One of the methods used to compute the geodesic distance is using fast marching algorithm(FMA). Geodesic distance computation using FMA assumes that propagation plane on the surface is planar. This is a reasonable assumption as further from the point from source, more planar the propagation plane becomes.



Planal front propagation

Fast marching algorithm is as follows:

Initialization:

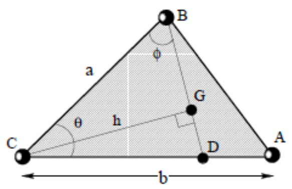
1) Choose a source point $T(X_0) = 0$, Mark point X_0 as Alive
2) Mark all neighbours of $T(X_0)$ as Close. The T values of points are initialised as the Euclidian distance.
3) Mark all other points as Far and initialized T value to infinity.

Loop:

1) Find point P in Close set with the smallest T values rename as Trail.
2) Add Trail set to Alive, remove from Close set.
3) Move all the adjacent vertices of to Close set. Also consider points in the Far set.
4) Update the T value of the trail.
5) Loop until there are no vertices in Far and Close set.

Update equation:

Consider a triangulated mesh, here the Trial C is being update And vertices A and B are in Alive set.



Triangular face of mesh

The equation to update the T values is given by the solution to the quadratic equation.

$$(a^2 + b^2 - 2ab\cos\theta)t^2 + 2bu(a\cos\theta - b)t + b^2(u^2 - a^2 * \sin\theta * \sin\theta) = 0$$

The value of u is given the difference in the T values of vertices in the alive set from which the plane is propagating.

$$u = T(B) - T(A)$$

The algorithm also assumes the vertex B and vertex A is updated before the vertex C is updated. This assumption will not hold for obtuse triangle.

Hence the solution should also satisfy the equation u < t

$$a\cos\theta < \frac{b(t-u)}{t} < a/\cos\theta$$

The T value of the vertex is updated according to the equation,

If u < t and $a\cos\theta < \frac{b(t-u)}{t} < a/\cos\theta$

$$T(C) = \min\{T(C), t+T(A)\}$$

else

$$T(C) = \min\{T(C), b+T(A), a+T(B)\}$$

X) Principle Curvature.

As triangular meshes are approximately represent the continuous surfaces, defining continuous normal vectors and curvature on the piecewise linear meshes is challenging.

In this paper [Meyer et al], defines and derives first and second order differential attributes such as surface normal, gaussian and mean curvature on piece wise linear triangular mesh.

X.a) Principle curvature normal.

Consider a triangle face in a mesh, the point in the triangle is given by

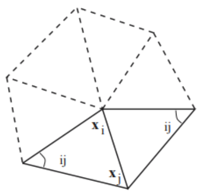
$$x = x_i B_i(u,v) + x_j B_j(u,v) + x_k B_k(u,v).$$

The gradient on the face is given by

$$\nabla_{u,v} x = x_i \nabla_{u,v} B_i(u,v) + x_j \nabla_{u,v} B_j(u,v) + x_k \nabla_{u,v} B_k(u,v).$$

Applying Gauss theorem, we get the solution which represents the mean curvature using angles of the triangle and vertex edge.

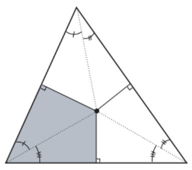
$$\iint_{Am} K(x)dA = 1/2 \sum_{j\in Ni(i)} (Cot\,\alpha_{ij} + Cot\,\beta_{ij})(x_i - x_j)$$



1-ring neighbor vertices of vertex i, with angles as α, β

Area of the voronoi around the point, assuming the triangle is a non-obtuse triangle is given by

$$A_{Vorono} = \frac{1}{8} \sum_{j \in Ni(i)} (Cot\, \alpha_{ij} + Cot\, \beta_{ij})(x_i - x_j)$$



Voronoi region of a triangle.

The algorithm to compute the area in the 1-ring neighborhood x, which may contain obtuse triangle is given as follows.

```
1) Initialize $A_{mixed}$= 0
2) Loop for each triangle in the 1 ring neighborhood of point P
3) if T is non -obtuse
            $A_{mixed}$ += Voronoi region of x
   Else
        If the angle at x is obtuse
            $A_{mixed}$ += (Voronoi region of x)/2
        Else
            $A_{mixed}$ += (Voronoi region of x)/4
```

X.b) Mean Curvature:

From the area of 1-ring neighbourhood the mean curvature K can be computed using the equation

$$k(x_i) = 1/(2 * A_{mixed}) \sum_{j \in Ni(i)} (Cot\, \alpha_{ij} + Cot\, \beta_{ij})(x_i - x_j)$$

X.c) Gaussian Curvature

Similarly, the gaussian curvature can be computed

$$k_G(x_i) = (2 * \pi - \sum_{j=1}^{\#f} \theta_j)/A_{mixed}$$

XI) Conclusion

Given the abundance of data and high accuracy in mesh labeling, Data driven mesh labeling algorithms provides a promising solution to the problem. This paper highlighted the pros and cons of different mesh labeling algorithms. Additionally, provided an insight into multiple subproblems to consider such as training time, accuracy, availability of data. Finally, the paper outlines multiple mesh feature extraction algorithms which is used to generate the feature vectors to train the classifiers.

XII) References

[1] KALOGERAKIS E., HERTZMANN A., SINGH K.:Learning 3d mesh segmentation and labeling. ACM Trans. On Graph (SIGGRAPH) 29, 4 (2010), 102:1–102:12. 1, 2, 3, 5, 7, 8

[2] Lv J, Chen X, Huang J, Bao H. Semi-supervised mesh segmentation and labeling. Computer Graph Forum 2012;31(7):2241–7.

[3] BENHABILES H., LAVOUÉ G., VANDEBORRE J.-P., DAOUDI M.: Learning boundary edges for 3d-mesh segmentation. Computer Graphics Forum 30, 8 (2011), 2170–2182. 1, 2, 4, 8, 10

[4] Xie, K. Xu, L. Liu, and Y. Xiong. 2014. 3D shape segmentation and labeling via extreme learning machine. *CGF* 33, 5, 85–95.

[5] K. Guo, D. Zou, X. Chen, 3D mesh labeling via deep convolutional neural networks, ACM Trans. Graph. (TOG) 35 (1) (2015) 3.

[6] SHAPIRA, L., SHAMIR, A., AND COHEN-OR, D. 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. 24, 4, 249–259.

[7] LIU, R. F., ZHANG, H., SHAMIR, A., AND COHEN-OR, D. 2009. A Part-Aware Surface Metric for Shape Analysis. Computer Graphics Forum, (Eurographics 2009) 28, 2.

[8] KIMMEL, R., AND SETHIAN, J. A. 1998. Computing geodesic paths on manifolds. Proc. of National Academy of Sci. 95(15) (July), 8431–8435

[9] MEYER, M., DESBRUN, M., SCHRO¨ DER, P., AND BARR, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In Visualization and Mathematics III (Proceedings of VisMath 2002), Springer Verlag, Berlin (Germany), 35–54.