Segmentation of Articulated 3D Mesh Sequences

by

Emre Kalafatlar

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science
in
Electrical and Computer Engineering

Koç University

October 2010

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Emre Kalafatlar

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assoc. Prof. Yücel Yemez (Advisor)

_____

Assoc. Prof. Çağatay Başdoğan

_____

Assist. Prof. T. Metin Sezgin

Date: _____

ABSTRACT

3D shape segmentation is a key step for processing and understanding of digitally acquired mesh models of 3D objects and has numerous applications in computer graphics and vision such as skeleton extraction and model deformation, mesh morphing, gesture recognition, shape retrieval, compression and collision detection. While most of the existing 3D shape segmentation methods consider only static meshes, dynamic shape models of moving objects are becoming more and more commonplace with the recent advances in 3D acquisition techniques. A typical example of this trend is the use of fixed connectivity mesh sequences to represent human actors with articulated motion.

In this thesis, we present a method to segment an articulated shape given in the form of a dynamic mesh sequence by incorporating motion information so as to further improve the segmentation results obtained by static segmentation methods. We assume that the articulated shape is given in the form of a mesh sequence with fixed connectivity so that the interframe vertex correspondences, hence the vertex movements, are known a priori. We use different postures of an articulated shape in multiple frames to constitute an affinity matrix which encodes both temporal and spatial similarities between surface points. The shape is then decomposed into segments in spectral domain based on the affinity matrix using one of the merge-cluster algorithm or the standard K-means clustering algorithm. The performance of the proposed segmentation method is demonstrated on various mesh sequences.

ÖZETÇE

3B biçimlerin kesimlemesi, 3B nesnelerin sayısal olarak elde edilmiş örgü modellerinin anlaşılması ve işlenmesinde önemli bir basamak olmakla beraber, iskelet özütleme, model deformasyonu, örgülerin şeklini değiştirme, hareket tanıma, biçimlerin geri kazanımı, sıkıştırma ve çarpışma sezimi gibi bilgisayar grafiği ve bilgisayarla görme konularında da birçok uygulamaya sahiptir. Mevcut 3B biçim kesimleme yöntemlerinin büyük çoğunluğunun yalnızca durağan örgüleri göz önünde bulundurmasına karşın, hareketli nesnelerin devingen biçim modelleri, 3B edinim yöntemlerinin yakın zamandaki gelişimiyle beraber hızla yaygınlaşmaktadır. Bu akımın tipik bir örneği de, insan aktörlerin eklemli hareketini gösteren, değişmez bağlanırlık sahibi örgü dizilerinin kullanımıdır.

Bu tezde, hareket bilgisi ekleyerek devingen örgü dizisi olarak verilmiş 3B eklemli şekillerin kesimlemesinde kullanılmak üzere ve durağan kesimleme yöntemleriyle elde edilen sonuçları iyileştirecek bir kesimleme yöntemi tanıtılmaktadır. Eklemli şeklin değişmez bağlanırlığa sahip bir örgü dizisi olarak verildiği, bu yüzden tepe noktalarının çerçeveler arası karşılıklarının ve dolayısıyla hareketlerinin önceden bilindiği varsayılmaktadır. Tepe noktaları arasındaki hem zamansal hem de uzamsal benzerlikleri kodlayan bir ilginlik matrisi oluşturmak için eklemli şeklin birden fazla çerçevedeki duruşu kullanılmaktadır. Daha sonra biçim, standart K-means ya da merge-cluster gruplandırma algoritmalarından birisi kullanılarak ilginlik matrisine dayalı olarak spektral alanda kesimlere ayrılır. İleri sürülen kesimleme yönteminin başarımı çeşitli örgü dizileri üzerinde gösterilmiştir.

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to Assoc. Prof. Yücel Yemez, my advisor, guide and master, for enlightening my way with his profound knowledge and his great ideas, for not giving up on me until the end, and for his immense patience from the initial to the final moment.

I would like to thank Assoc. Prof. Çağatay Başdoğan and Assist. Prof T. Metin Sezgin for taking part in my thesis committee. I am also grateful to Assoc. Prof. Alper T. Erdoğan for his patience and valuable help.

I am indebted to many of my colleagues for their support. I am especially grateful to Yusuf Sahillioğlu and S. Cihan Bilir for sharing their work with me, to C. Göktuğ Gürler for his help on coding with his fathomless programming knowledge and to Ferda Ofli for keeping my computer alive.

I owe thanks to my dear friends for putting up with me whenever I used my thesis as an excuse.

My final gratitude is for my precious family, which I am so glad to be a part of.

This work has been supported by TUBITAK.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

b_i, b_j	base vertices
K	number of clusters/initial centers
$\mathbf{D}$	overall distance matrix
$\mathbf{D}_G$	matrix encoding geodesic distances between all base vertices
$\mathbf{D}_g$	matrix encoding geodesic distances between adjacent base vertices
$\mathbf{D}_a, \mathbf{D}_a'$	angular distance matrices
$\mathbf{D}_m'$	matrix encoding variances of Euclidian distances between base vertices
$\mathbf{D}_m$	motion distance matrix
$\mathbf{D}_s$	spatial distance matrix
$w(i, j)$	weight of the edge between b_i, b_j
$\tilde{D}_g, \tilde{D}_a$	average value calculated over all nonzero entries
$\mathbf{W}$	affinity matrix
std	standard deviation
var	variance
mean	average value calculated over all entries of a matrix
$\mathbf{L}$	normalized symmetric Laplacian of $\mathbf{W}$
$\mathbf{\Lambda}$	diagonal matrix whose i^{th} diagonal element is the sum of the i^{th} row of $\mathbf{W}$
$\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_K$	first K eigenvectors of $\mathbf{L}$
$\lambda_1, \lambda_2, ..., \lambda_K$	K largest eigenvalues of $\mathbf{L}$

$\mathbf{V}$	matrix containing the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_K$ as columns
$c_1, c_2, ..., c_N$	cluster centers
T	center merging threshold distance for merge-cluster
μ	hyper-parameter used to calculate T
α	blending ratio of geodesic distance information
β	blending ratio of motion distance information
frame-cut	number of frames with smallest and largest Euclidian distances used in variance calculation
S	segmentation result
CD	cut discrepancy
avgRadius	mean of the distances of the points to the centroid of the mesh
d_G	geodesic distance
C	cut region/set of points on a boundary
R_m	missing rate
R_f	false alarm rate
$\|x\|$	surface area
$D_H(S_1 \Rightarrow S_2)$	directional hamming distance
RI	rand index
LCE	local consistency error
GCE	global consistency error
r	radius of a patch
$\dot{r}$	coefficient used to calculate the patch radius r

Chapter 1

INTRODUCTION

3D shape segmentation is a key step for processing and understanding of digitally acquired mesh models of 3D objects. 3D shape segmentation not only divides a mesh into semantic pieces but also reduces various analysis problems into smaller and simpler-to-solve pieces. The information obtained using 3D shape segmentation can be used in numerous applications in computer graphics and vision such as skeleton extraction and model deformation [8], mesh morphing [15], gesture recognition, compression [6] and collision detection [10]. While most of the existing 3D vision methods consider only static meshes, dynamic shape models of moving objects are becoming more and more commonplace with the recent advances in 3D acquisition techniques. A typical example of this trend is the use of fixed connectivity mesh sequences to represent human actors or animals with articulated motion. In this thesis, we present an automatic method to segment the articulated shape embedded in a dynamic mesh sequence by incorporating the motion information so as to further improve the segmentation results obtained by static segmentation methods.

Many algorithms have been proposed to decompose a static 3D mesh into meaningful segments, as surveyed and comparatively evaluated by Chen etal. in [3]. The method proposed by Lai et al. [9] for instance utilizes Grady's random walks method which was initially introduced for segmentation of images [16]. The algorithm first asks for manual selection of a user specified number of seed faces (or automatically selects them) as in

Grady's seed pixel selection, and then assigns each non-seed face to the seed for which it has the highest probability to reach after a random walk on the dual graph.

The method of fitting primitives, which is an effective segmentation method proposed by Antenne et al. [1], is based on the hierarchical face clustering method described in [17]. The method initially puts each face in a separate cluster and then hierarchically clusters faces by iteratively merging them according to the best fitting primitive shape (planes, spheres etc.). Another hierarchical decomposition method, suggested by Golovinskiy et al. [4], uses a randomized approach to find minimum cuts. In this method, a decimated version of the mesh is set to be the initial single cluster as a whole and then hierarchically divided into parts with binary segmentations at each iteration using the best of a set of randomized cuts, i.e., the one with the minimum normalized cut cost.

Shlafman et al. [15] describe a decomposition algorithm for morphing polyhedral surfaces, which applies K-means clustering to the matrix of pairwise face distances composed of angular and Euclidean distances. Katz et al. [8] improve this idea by replacing Euclidean distances with geodesic distances between adjacent faces, by differentiating the relative weight of convex and concave dihedral angles, and by avoiding jaggy boundaries with the use of fuzzy clustering instead of K-means. Liu et al. [11] simplify the problem computationally using spectral clustering, where the basic idea is to embed the data into a lower dimensional space by using the same distance matrix as Katz et al. [8] and then to perform K-means clustering.

Certainly, there are many other methods beside the ones evaluated in [3]. Yamauchi et al. [20] for example use a Gaussian curvature driven segmentation algorithm, while Chen et al. [21] propose a watershed-based algorithm combining Gaussian curvature with concavity estimation in order to compensate Gaussian curvature's lack of ability to differentiate convex and concave corners. Yu et al. [22] extract parallel slices from the 3D model, and then divide the model based on the contours of slices.

All the aforementioned segmentation techn̨ques apply only to static meshes. However, in a number of cases, especially for articulated shapes, depending on the posture of the model, it becomes almost impossible to extract sufficient information to identify some of the cut regions. When the benchmark results obtained in [3] are investigated for example, one can observe various segmentation failures. In Figure 1.1 we see an example of how the algorithms compared in [3] may fail to locate some parts of the given surface mesh, e.g., the elbows on the arms, due to the straight stance of the arm and the resulting smooth surface from wrist to shoulder, or even some fail to separate the torso from the legs.

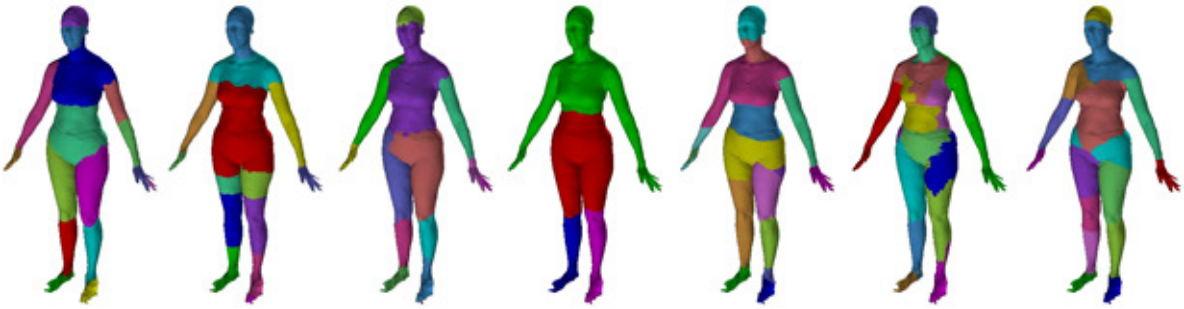

Figure 1. 1 Segmentation results for a static human model extracted from [3]. From left to right, Randomized Cuts [4], Shape Diameter Function [14], Normalized Cuts [4], Core Extraction [7], Random Walks [9], Fitting Primitives [1] and K-Means [15].

In this thesis, we propose to incorporate the motion information, whenever available, to circumvent the problems encountered in segmentation of static meshes such as the one demonstrated above. To this effect, we describe a segmentation method that is applicable to mesh sequences with articulated motion. By analyzing multiple postures of an articulated shape and positions of different parts of the mesh in multiple frames, we deduce information about the semantic relations between different parts of the shape. We assume that semantically different parts of an articulated object exhibit different rigid motions. So

far this idea has been exploited for mesh segmentation only a few times, such as in [19] and [18].

Lee et al.[19] use a distance metric based on the deformation between faces to define boundary regions. The idea originates from the fact that the distance between two neighboring faces changes over time and the amount of this change is maximum at the joints. Therefore, regions of large deformation, where the distance, defined as a combination of geodesic and deformation distances, between two adjacent faces is greater than an experimentally set threshold, are regarded as boundary regions.

Arcila et al. [18] use displacement vectors to segment a mesh into clusters. One of the strengths of the algorithm is that it can be applied to mesh sequences with varying number of vertices. The first step of the technique is to find approximate displacement vectors between two frames. The second step is to perform an initial segmentation using the displacement vectors. The mesh is divided into clusters within which vertices either stay static, or move rigidly, or are stretched in a uniform fashion. At the third step, the initial segmentation is refined iteratively with each incoming frame, that gives the algorithm the ability to segment on the fly.

Our approach differs from existing methods in that we use both spatial information and temporal information at the same time for dynamic mesh segmentation as in our previous work [28]. In other words, we aim to take benefit of static mesh segmentation methods as well as the information that we can extract by analyzing the motion in a dynamic mesh sequence. To this effect, we generate a distance matrix based on a blend of angular distances, geodesic distances and variances of Euclidian distances between vertices. The block diagram depicting our algorithm is given in Figure 1.2. We compute the blending components of the overall distance matrix after downsampling the vertices to a number of uniformly distributed base vertices. The overall distance matrix is then transformed into an affinity matrix encoding the likelihood of the vertices belonging to the same cluster,

followed by calculation of the Laplacian of the affinity matrix, hence implying reduction in dimensionality and simplification of the clustering problem. The last step is clustering the base vertices in the spectral domain using K-means algorithm, which results in a pre-determined number of clusters, or using merge-cluster algorithm which automatically selects the number of clusters.

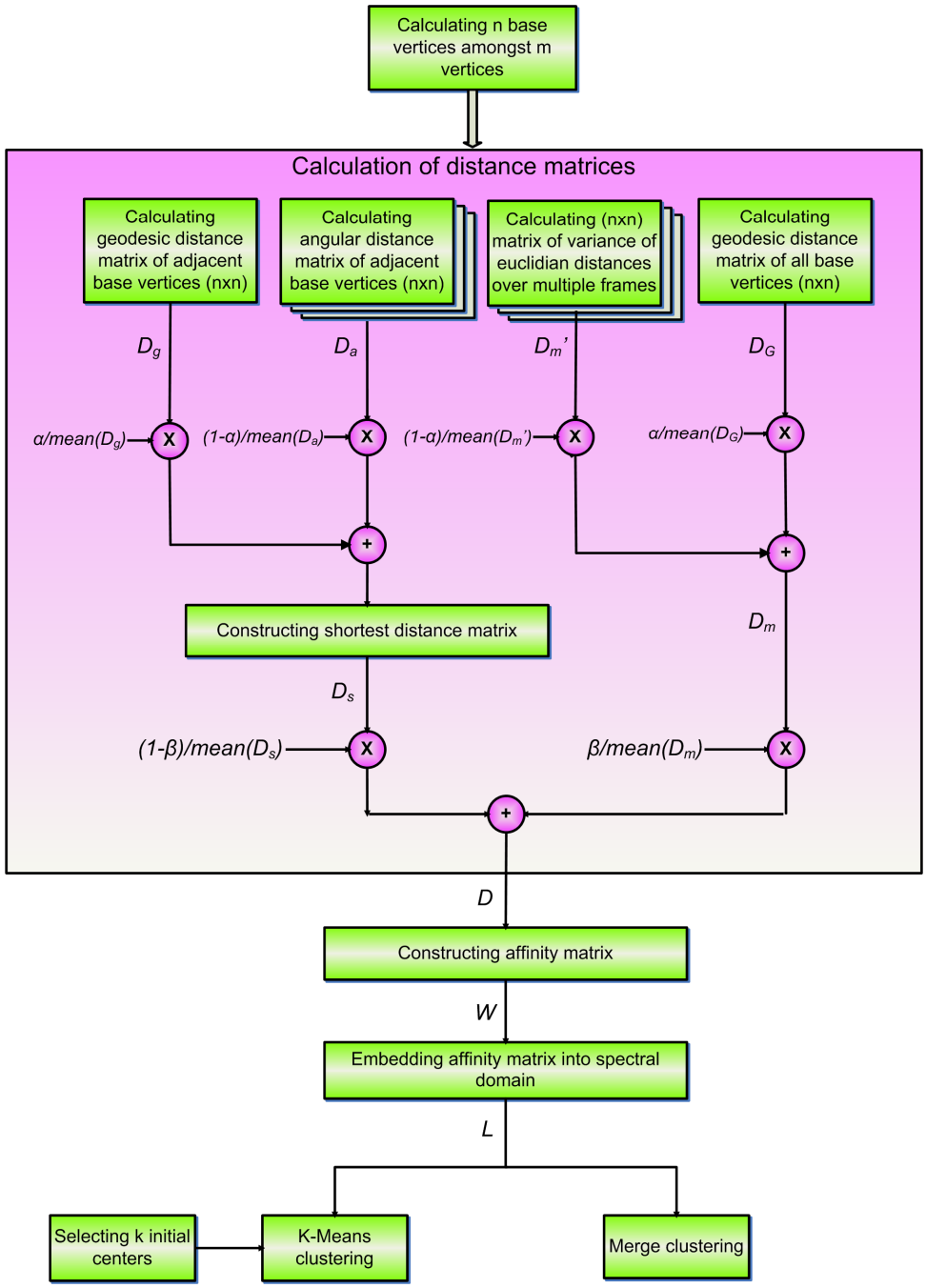

Figure 1. 2 The block diagram depicting the algorithm.

Chapter 2

THE PROPOSED SEGMENTATION ALGORITHM

We assume that the mesh sequence to be segmented is of fixed connectivity, meaning that the number of vertices and mesh connectivity do not change through the frames of the sequence. If the sequence is not of fixed connectivity, there exist a vast number of methods in the literature, such as [23, 24] and [30], to find vertex-to-vertex shape correspondences that can be used prior to our segmentation method. Analyzing multiple frames of a time-varying mesh model is time and memory consuming. Therefore, we sought for a way to reduce this vast resource demand. The high computational cost and the big memory requirement is a result of the huge matrices we use to encode vertex-to-vertex distances for each frame of the mesh sequence. Because we employ such matrices, doubling the number of vertices in a mesh quadruples the size of matrices. To lower the resource amount needed, as a first step, we downsample the vertices of the mesh sequence to a desired number of vertices, called *base vertices*. The base vertices, computed by using the *Dijkstra's shortest paths algorithm* as suggested in [5], are enforced to be distributed equally on the surface of the mesh (see Figure 2.1).

Figure 2. 1 Base vertices.

Every base vertex has its own base area, called a patch. The number of base vertices generated by the algorithm, hence the size of the resulting patches, can be controlled by adjusting the patch radius *r* using $r = \sqrt{\dot{r} \cdot area(M)}$, where $area(M)$ is the surface area of the mesh *M*. In order to adjust the radius *r,* we change the value of coefficient $\dot{r}$, and in our experiments we always set $\dot{r}$ such that the number of base vertices is around 1000. The patch of a base vertex is composed of faces nearby. Each face is assigned to the patch of the base vertex that is closest to its center. After each base vertex is clustered, faces

residing within its patch are assumed to be in the same cluster with it. In Figure 2.2, we display the patches for a model.

Figure 2. 2 Patches. Each surface segment is the patch of a base vertex.

2.1 Distance Matrices

Our algorithm is based on clustering base vertices according to their likelihood of being together. In order to determine the likelihood of two base vertices *i* and *j* to be in the same cluster, we first constitute a distance matrix $D(i, j)$, which is based on the matrix used by Katz et al. [8] and the one used in [11]. The overall distance matrix $\mathbf{D}$ is a blend of four separate distance matrices encoding different distance associations between base vertices. These four matrices are two geodesic distance matrices, an angular distance matrix and a motion distance matrix, denoted by $\mathbf{D}_g, \mathbf{D}_G, \mathbf{D}_a$ and $\mathbf{D}_m$ respectively.

2.1.1 Geodesic Distance Matrix

In order to incorporate geodesic distance information into our method, we constitute two different geodesic distance matrices. Our first geodesic distance matrix is simply the matrix where each entry, $D_g(i, j)$, encodes the geodesic distance between adjacent base vertices b_i and b_j. If b_i and b_j are not adjacent, the corresponding value is set to *zero*. The second geodesic distance matrix $\mathbf{D}_G$ encodes the geodesic distances between all base vertices regardless of their adjacency. We calculate geodesic distances using Dijkstra's shortest path algorithm [27]. The basic assumption in using this matrix is that the probability of two base vertices being in the same cluster decreases as the distance between them increases.

2.1.2 Angular Distance Matrix

The angular distance matrix $\mathbf{D}_a$ encodes the distances between the normals of the adjacent base vertices. The normal of a base vertex is computed as the area-weighted average of the normals of the faces within the individual patch of that base vertex. The angular distance $D_a(i, j)$ between two adjacent base vertices b_i and b_j is calculated as follows:

$$D_a(i, j) = \frac{1}{L}\sum_{l}\eta(1-\cos\theta(i, j, l)) \tag{1.1}$$

where $\theta(i, j, l)$ is the angle between the normals of the base vertices b_i and b_j in frame *l*, and L is the total number of frames in the sequence. If b_i and b_j are not adjacent the corresponding value is set to *zero*. When b_i and b_j are coplanar, they have the highest probability of being in the same cluster so they have the minimum distance between. When b_i and b_j are facing the opposite directions, they have the lowest probability of being in the same cluster so they have the maximum distance between. Note that the range of values for $D_a(i, j)$ is so that $0 \le D_a(i, j) \le 2$, and when θ is 0, i.e. b_i and b_j are coplanar, its value becomes minimum and when θ is π, i.e. b_i and b_j are facing towards exactly the opposite directions, its value becomes maximum. Another fact, based on [8] and [11], is that a concave angle between two base vertices indicates a higher probability of belonging to a cut region than a convex angle. Therefore, η =1 is used for concave angles to express their property of being better candidates to be on a boundary, whereas η =0.01 is used for convex angles. These values have been experimentally set and as long as η is a smaller positive value for convex angles, many other values can be tried.

In addition to $D_a(i,j)$, another matrix, that we denoted by $\mathbf{D}_a'$, is also calculated as shown in Equation 1.2. The difference between two matrices is that $D_a(i,j)$ encodes the average angular distances calculated over all frames whereas $D_a'(i,j)$ encodes the angular distances calculated on the first frame of a sequence. As we will discuss in Chapter 3, we use $D_a(i,j)$ and $D_a'(i,j)$ interchangeably to show the effect of using multiple frames in angular distance calculation. Therefore, $D_a(i,j)$ can be replaced with $D_a'(i,j)$ wherever we use it in this chapter:

$$D_a'(i,j) = \eta(1-\cos\theta(i,j)) \tag{1.2}$$

2.1.3 Motion Distance Matrix

In the case of articulated motion, the Euclidian distance between two surface points belonging to the same segment is expected to remain almost constant. Thus, if the distance between two base vertices significantly varies with time, we say, they do not exhibit the same rigid motion so that they are unlikely to be in the same segment. In order to incorporate the motion information, we introduce a motion distance term, denoted by $D_m'(i,j)$, which holds the variance of pairwise Euclidian distances of the base vertices b_i and b_j over the mesh sequence, calculated as follows:

$$D_m'(i,j) = var(d_E(i,j,l)) \tag{1.3}$$

where $d_E(i,j,l)$ is the Euclidian distance between b_i and b_j in frame l and *var* denotes the variance over multiple frames. During variance calculation, only some equal number of smallest and largest Euclidian distances are taken into account so as to eliminate the frames

in which the pairs of base vertices stay relatively stationary. We denote the number of smallest and largest Euclidian distances used by *frame-cut*. The overall motion distance matrix $\mathbf{D}_m$ is then calculated as a blend of $\mathbf{D}_m'$ and $\mathbf{D}_G$:

$$D_m(i,j) = \alpha \frac{D_G(i,j)}{mean(\mathbf{D}_G)} + (1-\alpha)\frac{D_m'(i,j)}{mean(\mathbf{D}_m')} \tag{1.4}$$

where *mean* is the average value calculated over all entries of the corresponding matrices.

2.1.4 Spatial Distance Matrix

The spatial distance matrix, that we denote by $\mathbf{D}_s$, is computed based on $\mathbf{D}_g$ and $\mathbf{D}_a$. We assign a weight, $w(i,j)$, to each edge of the mesh, that is, for each adjacent base vertex pair b_i, b_j:

$$w(i,j) = \alpha \frac{D_g(i,j)}{\tilde{D}_g} + (1-\alpha)\frac{D_a(i,j)}{\tilde{D}_a} \tag{1.5}$$

where $\tilde{D}_g$ and $\tilde{D}_a$ are the average values computed over the nonzero entries of the corresponding distance matrices, and the value of the blending ratio α is set to be the same as the value of α used for motion distance term. Each entry of the spatial distance matrix, $D_s(i,j)$, is then given by the length of the shortest path computed on the resulting weighted graph between base vertices b_i and b_j For this purpose, we employ *Dijkstra's shortest path algorithm* [27].

2.1.5 Overall Distance Matrix

Given the above definitions the overall distance matrix **D** is computed by the following weighted summation:

$$D(i,j) = \beta \frac{D_m(i,j)}{mean(\mathbf{D}_m)} + (1-\beta)\frac{D_s(i,j)}{mean(\mathbf{D}_s)} \tag{1.6}$$

where β value is the weight used to adjust the relative importance of the motion distance and the spatial distance values and mean is the average of all entries of the corresponding matrices. As previously mentioned, our distance matrix is based on the one used by Katz et al. [8] and Liu et al. [11]. However, unlike our approach, their distance matrices are based on the distances between face centers. Because we use base vertices, which are located at the centers of patches formed of faces, we use the positions and normals of base vertices analogously to using the positions of face centers and face normals, respectively. Since we stick to their distance matrix to some extent, we set the value of α based on their experimental results and adjusted only β, which will be mentioned in detail in the next chapter.

2.2 Affinity Matrix

Before starting to cluster base vertices with respect to their similarity, we construct an affinity matrix, which encodes the likelihood of base vertices to be in the same segment. This is basically the transformation of the distance matrix **D** encoding the unlikelihood of base vertices to be in the same segment into a matrix doing just the opposite. The

transformation of the distance matrix into the affinity matrix **W** is achieved the following exponential kernel:

$$W(i,j) = e^{-D(i,j)/2\sigma^2} \tag{1.7}$$

This kernel normalizes the affinity values into [0,1] interval. In order to properly set the parameter σ, we use the following equation:

$$\sigma = e^{-std(\mathbf{D})} \tag{1.8}$$

where *std(***D***)* is the standard deviation computed over the all entries of the matrix **D**.

2.3 Spectral Clustering

In order to achieve clustering based on the affinity matrix, there exist various schemes in the literature, among which we favor spectral clustering for its simplicity and computational efficiency [2, 11, 12]. Spectral clustering is the general name of many clustering methods utilizing dimensionality reduction on the spectrum of the affinity matrix. We reduce the dimensionality of our NxN affinity matrix, embedding the shape coordinates into a lower K-dimensional space. For this purpose we calculate the normalized symmetric Laplacian **L** of the affinity matrix **W** by $\mathbf{L} = \mathbf{\Lambda}^{-1/2}\mathbf{W}\mathbf{\Lambda}^{1/2}$, where $\mathbf{\Lambda}$ is a diagonal matrix whose i^{th} diagonal element is the sum of the i^{th} row of **W**. Next step is applying a spectral decomposition, i.e., eigenvalue decomposition on **L.** Let $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_K$ be the first K eigenvectors of **L** corresponding to the K largest eigenvalues $\lambda_1, \lambda_2, ..., \lambda_K$. We then compute the NxK matrix **V** containing the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_K$ as columns. The rows of **V**, normalized so as to have unit norm, provide us with the K dimensional points in the spectral domain. Following the common practice, as also in [11, 25, 26], we set the number of eigenvectors to be used as *K*, i.e., the same as the number of clusters we desired.

Polarization theorem [2] states that the projection of data points to a lower rank K amplifies the unevenness in the point distribution on the K dimensional sphere. Truncating the dimensions corresponding to the smallest eigenvalues creates an embedding which distorts the large angles more than the small angles between vectors. This implies that the points of high affinity, i.e. the points whose corresponding vectors have small angles between, will be grouped together on the K dimensional sphere while the points of low affinity, i.e. the points whose corresponding vectors have large angles between, will move apart. Therefore, clustering of the data embedded on lower dimensional space will be simple enough to obtain by employing basic clustering techniques. In order to perform clustering, we utilize two methods, which are K-means clustering and merge clustering.

2.3.1 K-Means Clustering

K-means clustering is a well known and simple clustering algorithm. The algorithm is an iterative one which, at each iteration, assigns each data point to one of the K clusters whose mean is closest to that point, and updates the cluster means at the end of each iteration. Although, the cluster means are updated at each iteration, before the first iteration they have to be set somehow. This initialization of the cluster means is an important problem for all K-means clustering techniques, since the bad choice of initial centers may lead to undesired clustering results. In our K-means scheme, for the sake of starting with good initial centers and having better clustering results, we initially find K base vertices scattered as evenly as on the mesh surface by utilizing the same method that we use to find the base vertices. A standard K-means clustering is then performed on the points in spectral domain, using the rows of $\mathbf{V}$ corresponding to the indices of these K base vertices as initial cluster centers. However, this method has a drawback, which is the necessity of setting the number of clusters manually. Since the high number of hyper-parameters that are set

manually is undesirable, we also employ a second method which chooses the number of segments automatically.

2.3.2 Merge Clustering

In order to increase the degree of automation of our algorithm, we utilize a basic merging based clustering algorithm similar to the Isodata clustering algorithm [31]. In this method, instead of choosing K initial centers out of N points in spectral domain, we initially regard each of N points as individual centers, and start with N clusters. At each iteration, the distances between the cluster centers are calculated and if the distance between the centers of two clusters is below a certain threshold, these two clusters are merged. Iterations continue until all cluster means are apart enough from each other so that we cannot find a center-to-center distance value smaller than the threshold. The mentioned threshold T is not constant and adaptively calculated at each iteration as follows:

$$T = mean\left(dist\left(c_{i,j}\right)\right) - \mu \,\mathrm{var}\left(dist(c_{i,j})\right) \tag{1.9}$$

where $mean\left(dist\left(c_{i,j}\right)\right)$ is the mean of the distances between centers $c_1, c_2, ..., c_N$ and $var\left(dist\left(c_{i,j}\right)\right)$ is the variance of the distances between them. The value of μ is set $2.5 \le \mu \le 3.5$. This range has been experimentally found to work properly. When μ is larger than 3.5, T becomes smaller and the centers need to be closer to each other in order to be merged leading to an over-segmentation of the mesh with too many segments. When μ is smaller than 2.5, T becomes larger and more of the centers will be interpreted as belonging to the same segment and will be merged leading to an under-segmentation of the mesh. The overall merge cluster algorithm is as follows:

1. Set each row of **V** as an individual center c_i.
2. Calculate the threshold T using:
   $T = mean(dist(c_{i,j})) - \mu \operatorname{var}(dist(c_{i,j}))$
3. For every c_i, c_j pair, if $dist(c_i, c_j) < T$, then merge all points belonging to c_i and c_j to form a single cluster. If such a pair cannot be found, jump to step 7.
4. Update the clusters and the number of clusters.
5. Update the cluster centers by taking the means of the coordinates belonging to each point.
6. Return to step 2.
7. Apply K-means clustering to refine the result

One drawback of this algorithm, as compared to standard K-means clustering, is that, although we can force the K-means algorithm to produce the exact number of segments we favor, there is no way of knowing the number of segments beforehand when we use merge-cluster. If the information carried by the affinity matrix is fuzzy for some regions, merge-cluster may fail to keep two segments apart and mistakenly merge them, although K-means may keep them separated if it is forced to produce two different segments in that region.

Chapter 3

EXPERIMENTS AND RESULTS

We have tested our method with experiments conducted on four different mesh sequences, three of which are acquired by 3D surface tracking methods, therefore real sequences, whereas the fourth one is constructed artificially, therefore a synthetic sequence. For each sequence, we assume that mesh connectivity is fixed and does not change through frames. Therefore, prior to calculating distance matrices, the vertex-to-vertex correspondences are given. For further information on 3D shape correspondence problem one can refer to [23, 24] and [30].

All four meshes are segmented using varying values of hyper-parameters, which can not be selected automatically and adaptively by the algorithm, in order to find an optimum blend of the parameters. These parameters include:

- The blending ratio β of the motion distance.
- The number of frames taken into account for each base vertex while calculating the variance of Euclidian distance in order to maximize the effect of motion, which we call *frame-cut*.
- μ value which is used to calculate the distance threshold utilized to decide on which clusters will be merged at each iteration of our merge-cluster algorithm.

Note that the parameter α, which is the blending ratio of the geodesic distance, is set as $0.01 \leq \alpha \leq 0.05$ by Liu et al [11]. In our experiments, we have used the mean of this interval, that is $\alpha = 0.03$.

For each sequence, we have prepared a ground truth segmentation. Our ground truth segmentations are based on human generated segmentations given in [3] as well as on how we perceive each model. The quality of the mesh is also a factor affecting our ground truth segmentation. For instance, we do not divide the hands of the *Jumping Man* model (see Fig. 3.11) because both hands are not constructed well enough to be distinguished from the rest of the arm. Each ground truth segmentation is shown in their respective sections. We post-process the ground truth segmentations, which we initially obtain, to prepare them for evaluation of our segmentation results. Since our segmentation algorithm is based on base vertices and the information for each base vertex comes from the faces belonging to its patch, we first downsample the ground truth mesh by finding corresponding segment for each base vertex and then upsample the model by assigning each face to the segment of the base vertex it is claimed by.

3.1 Error Metrics

Besides evaluating the performance of our algorithm by providing visual segmentation results, we also assess the performance quantitatively based on a set of error metrics. To measure the deviation of a segmentation result from the ground truth segmentation, we utilize the four different error metrics described by Chen et al. [3], including one boundary and three region based methods. We use all four of these metrics since the error information provided by each is about a different aspect of segmentation results. Each metric algorithm is explained in the sequel.

Executable files, source codes, mesh and segmentation data are provided by the authors on the project web site[†]. The executables we use for error calculation are also fetched from the project web site.

For each mesh we provide visual segmentation results as well as plots showing metric errors. We expect both the visual and the metric results match.

3.1.1 Cut Discrepancy

The first metric, *Cut Discrepancy*, basically measures the distances between resulting segmentation boundaries and the closest cuts in the ground truth segmentation. The error, denoted by *CD*, between segmentations S_1 and S_2 is calculated as the mean of each directional cut discrepancy *DCD* as follows:

$$CD(S_1, S_2) = \frac{DCD(S_1 \Rightarrow S_2) + DCD(S_2 \Rightarrow S_1)}{avgRadius} \tag{3.1}$$

where *avgRadius* is the mean of the distances of the points to the centroid of the mesh. Directional cut discrepancy from S_1 to S_2 is given by

$$DCD(S_1 \Rightarrow S_2) = mean\{d_G(p_1, C_2), \forall p_1 \in C_1\} \tag{3.2}$$

where $d_G(p_1, C_2)$ is calculated using:

$$d_G(p_1, C_2) = \min\{d_G(p_1, p_2), \forall p_2 \in C_2\} \tag{3.3}$$

$d_G(p_1, p_2)$ denotes the geodesic distance between the points p_1 and p_2, and C_1 and C_2 are the sets of points on the boundaries of S_1 and S_2, respectively.

[†] http://segeval.cs.princeton.edu

3.1.2 Hamming Distance

The second metric, *Hamming Distance*, measures the regional differences between segmentations. For two mesh segmentations S_1 and S_2, the *Hamming Distance* is the average of the missing rate R_m and the false alarm rate R_f, which are given by:

$$R_m(S_1, S_2) = \frac{D_H(S_1 \Rightarrow S_2)}{\|S\|} \tag{3.4}$$

$$R_f(S_1, S_2) = \frac{D_H(S_2 \Rightarrow S_1)}{\|S\|} \tag{3.5}$$

where $\|S\|$ is the overall surface area and the *Directional Hamming Distance* $D_H(S_1 \Rightarrow S_2)$ is the sum of the differences between the best corresponding segments in S_1 and S_2. By sum of the differences, we mean the sum of the areas of each missegmented face.

3.1.3 Rand Index

The third metric, *Rand Index*, measures the number of agreements between segmentations S_1 and S_2 in terms of face pairs without the need for finding segment correspondences. If a pair of faces *i* and *j* are in the same segment in S_1, C_{ij} is set to 1, otherwise it is set to 0, and if they are in the same segment in S_2, P_{ij} is set to 1, otherwise it is set to 0. The function encoding the dissimilarity between S_1 and S_2 in terms of C_{ij} and P_{ij}, where N is the number of faces, is as follows

$$RI(S_1, S_2) = 1 - \binom{2}{N}^{-1} \sum_{i,j,i<j} \left(C_{ij} P_{ij} + (1 - C_{ij})(1 - P_{ij}) \right) \tag{3.5}$$

3.1.4 Consistency Error

Consistency Error, the forth error metric, is also based on regional differences. The method first computes the *Local Refinement Error* $E(S_1, S_2, f_i)$ for each face f_i for segmentations S_1 and S_2 using

$$E(S_1, S_2, f_i) = \frac{\|R(S_1, f_i) \setminus R(S_2, f_i)\|}{\|R(S_1, f_i)\|} \tag{3.6}$$

where $\|x\|$ is the total area of faces in set x, "\" is the set difference operator and $R(S, f_i)$ is the cluster which f_i resides in S. Using the *Local Refinement Error*, two error types, *Global Consistency Error (GCE) and Local Consistency Error (LCE)*, are calculated as follows

$$GCE(S_1, S_2) = \frac{1}{n} \min\left\{ \sum_i E(S_1, S_2, f_i), \sum_i E(S_2, S_1, f_i) \right\} \tag{3.7}$$

$$LCE(S_1, S_2) = \frac{1}{n} \sum_i \min\{E(S_1, S_2, f_i), E(S_2, S_1, f_i)\} \tag{3.8}$$

The main disadvantage of this metric is that it tends to produce higher scores when the numbers of segments in S_1 and S_2 are different. Although they may be misleading, especially for the results of merge-cluster since the number of segments produces by the algorithm is unpredictable, we will provide *Consistency Error* values besides other metrics.

3.2 Segmentation Results

3.2.1 Sequence 1

Our first mesh sequence, *Horse Gallop*, was made available by Robert Sumner and Jovan Popović from the Computer Graphics Group at MIT[‡]. The sequence consists of 48 frames of a galloping horse model. In Figure 3.1 we display sample frames from the sequence to give an idea on the motion exhibited by the mesh. The particularity of this model is that, it contains a pretty smooth surface for each different segment and sharp edges at segment borders, which we expect helps segmentations obtained without incorporating any motion information. Furthermore, it exhibits clear, rigid and distinct motions for each segment, which we expect helps segmentations obtained using motion information. The mesh is composed of 8431 vertices and 16843 faces. The number of vertices subsampled to 995 base vertices by setting $\dot{r} = 0.00065$.

[‡] http://people.csail.mit.edu/sumner/research/deftransfer/data.html

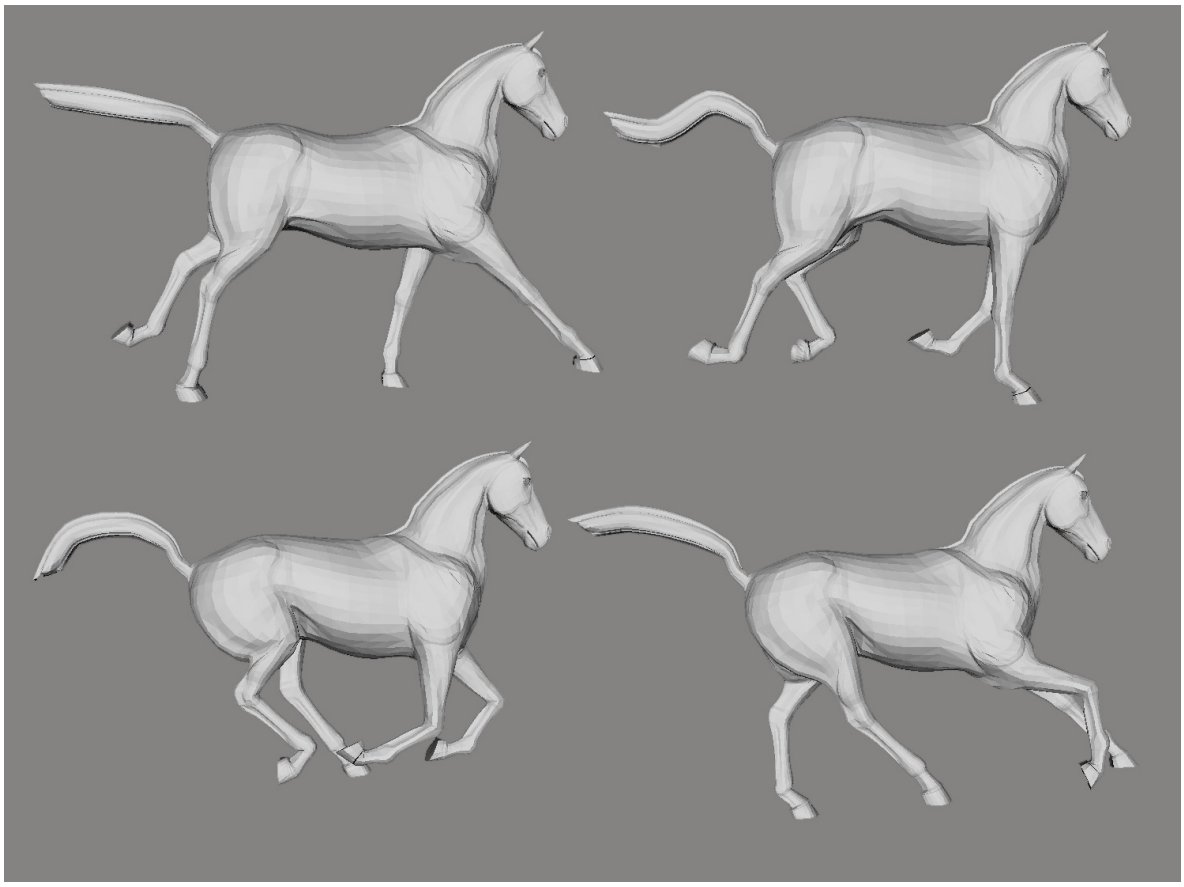

Figure 3. 1 Four frames of Horse Gallop sequence depicting the motion of the model.

We test our segmentation algorithm under six different settings of the parameters. These settings are:

1) $\mu = 2.5$ and *frame-cut=10*

2) $\mu = 2.5$, *frame-cut =10* and average angular distances are used.

3) $\mu = 2.5$ and *frame-cut =20*

4) $\mu = 3.5$ and *frame-cut =10*

5) $\mu = 3.5$, *frame-cut =10* and average angular distances are used.

6) $\mu = 3.5$ and *frame-cut =20*

In Figure 3.2, we plot the segmentation errors obtained using different metrics and different parameter settings for varying values of β, i.e., the motion information blending factor (with 0.05 increments in the interval $0 \le \beta \le 1$), The segmentation error in each case is computed as the sum of the two errors, one obtained with K-means clustering and the other obtained using merge-cluster algorithm.

The number of segments resulting from each parameter setting is given in Table 3.1. Moreover, average error values of each parameter set for each error type calculated over β are provided in Table 3.2.

β	**Set 1**	**Set 2**	**Set 3**	**Set 4**	**Set 5**	**Set 6**
0	10	10	10	21	24	21
0.05	10	15	11	23	22	22
0.10	12	10	11	23	18	20
0.15	16	15	11	20	21	21
0.20	10	12	10	18	21	19
0.25	12	10	13	18	18	17
0.30	12	11	13	18	21	20
0.35	11	12	12	20	19	19
0.40	12	12	10	19	18	19
0.45	12	10	12	19	16	20
0.50	12	12	12	18	18	19
0.55	10	11	11	17	18	17
0.60	12	11	11	16	15	16
0.65	11	10	10	16	15	17
0.70	11	10	11	16	14	15
0.75	11	11	10	14	16	14
0.80	10	11	11	15	14	14
0.85	11	10	11	13	14	15
0.90	9	11	10	16	14	15
0.95	11	11	11	16	16	16
1.00	12	12	10	16	16	16

Table 3. 1 Number of segments resulting from each parameter set for the *Horse Gallop* sequence.

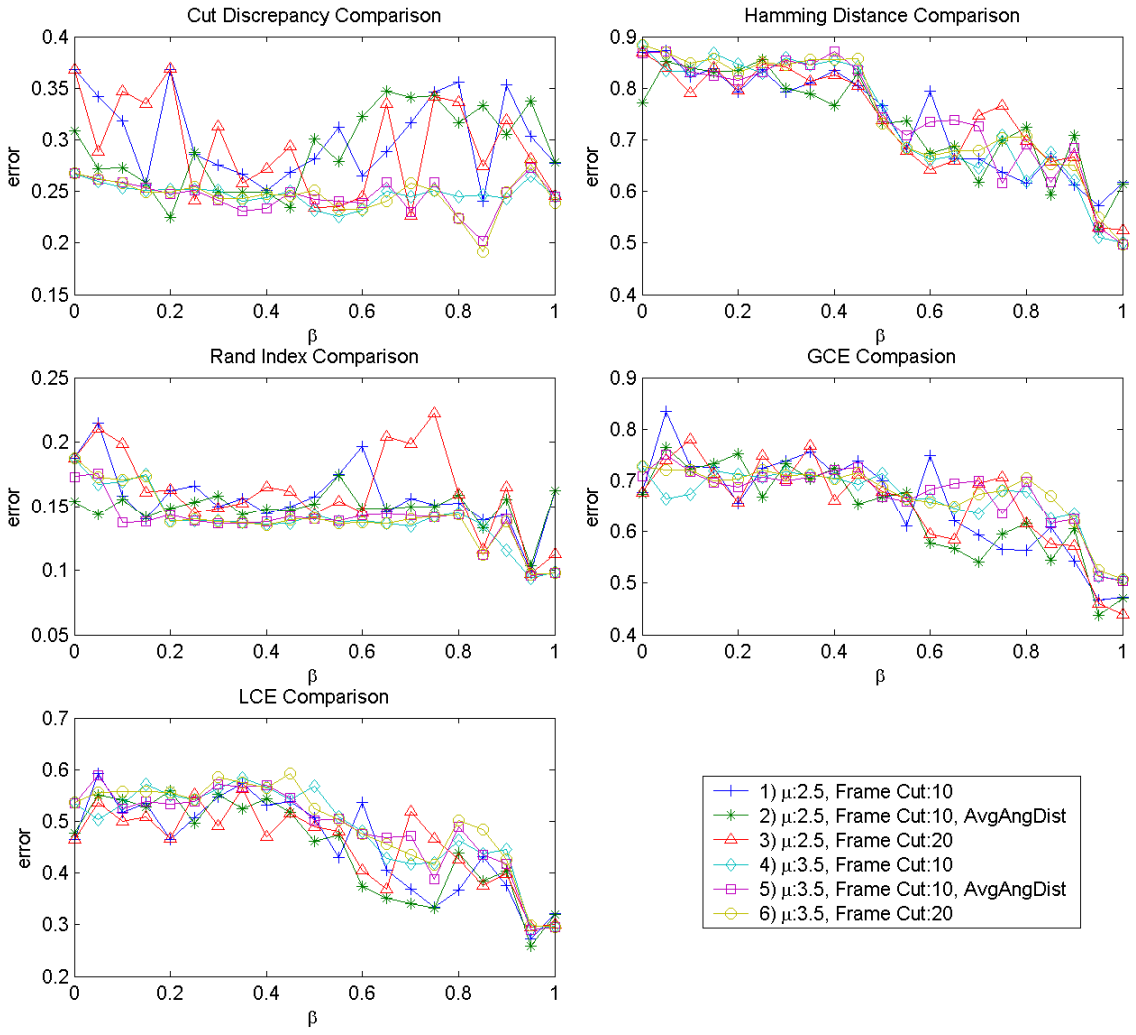

Figure 3. 2 Segmentation errors obtained for the *Horse Gallop* sequence under different parameter settings.

	CD	HD	RI	GCE	LCE
Set 1	0.30197	0.74112	0.15738	0.65637	0.45827
Set 2	0.29086	0.73716	0.14876	0.63949	0.44912
Set 3	0.29337	0.7416	0.16248	0.65477	0.45683
Set 4	0.24758	0.74029	0.14015	0.66612	0.48772
Set 5	0.24564	0.74917	0.13789	0.67229	0.48805
Set 6	0.24602	0.75033	0.14054	0.67496	0.49709

Table 3. 2 Average total error values calculated over β for the *Horse Gallop* sequence

3.2.1.1 Metric Results

Since merge-cluster algorithm may produce different number of segments, we favor the Rand Index the most as this error metric does not need to find segment correspondences. With this assumption, when we look at the plots in Figure 3.2, and the values given in Table 3.2 we see that the best result is produced by the segmentation using the parameter set 5. In order to provide a more detailed look to the results of this parameter set, in Figures 3.3, 3.4, 3.5 and 3.6, we display the plots of each error metric both for K-means and merge-cluster algorithms.

In Figure 3.3, *Cut Discrepancy* error results are plotted. Since *Cut Discrepancy* measures the distances between cuts, we expect K-means to produce better results than merge-cluster due to the fact that merge-cluster algorithm does not produce the exact number of segments as defined in the ground truth segmentation whereas K-means algorithm can be forced to do that as verified by the given plot.

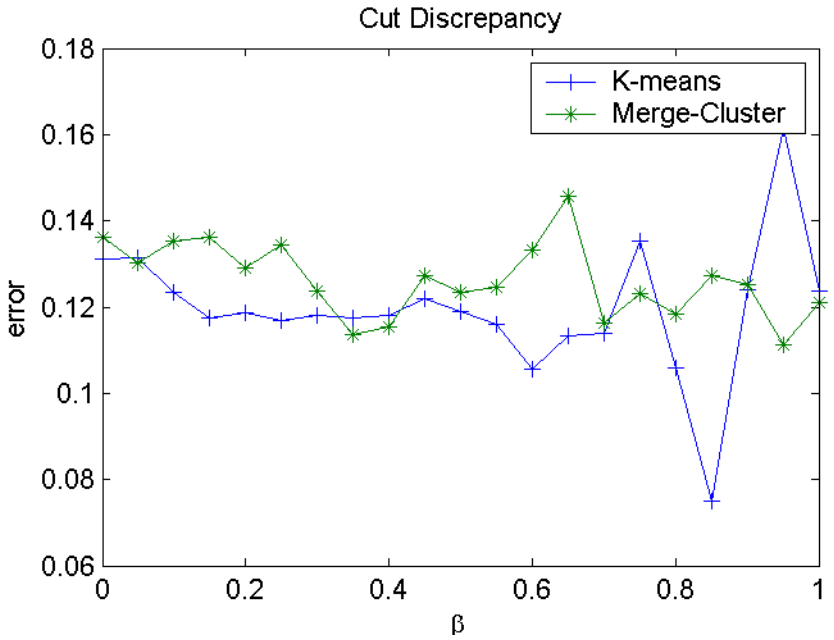

Figure 3. 3 *Horse Gallop* Cut Discrepancy error plot for the parameter set 5.

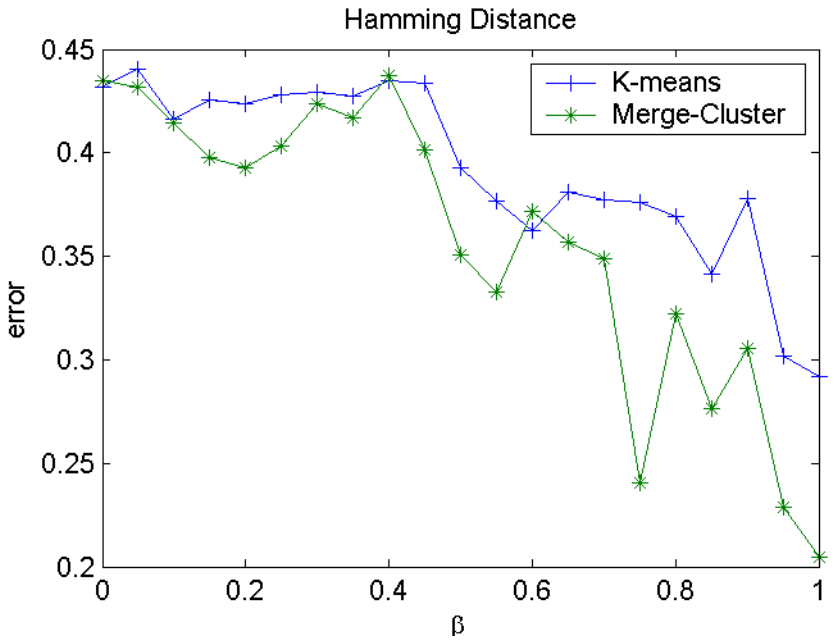

Figure 3. 4 *Horse Gallop* Hamming Distance error plot for the parameter set 5.

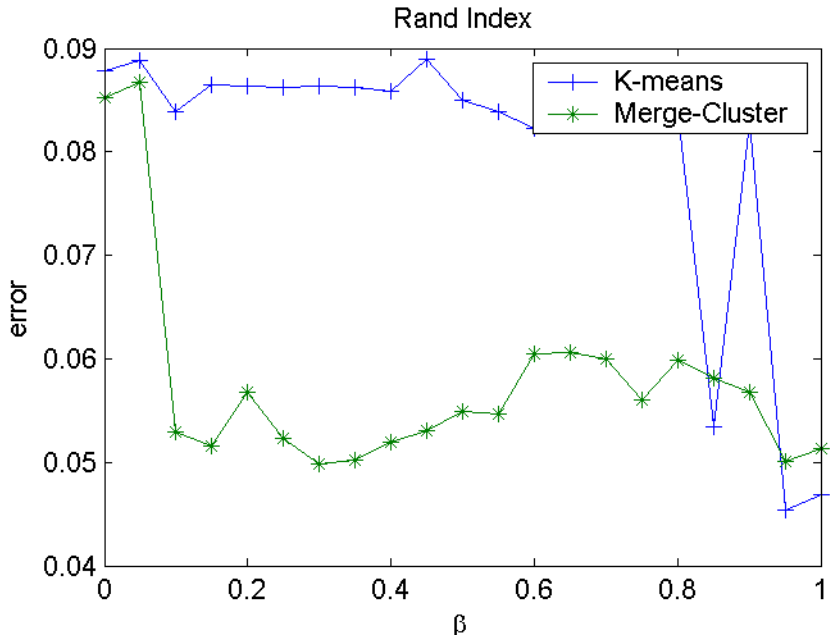

Figure 3. 5 *Horse Gallop* Rand Index error plot for the parameter set 5.

We expect the merge-cluster algorithm to produce better results than K-means in terms of *Hamming Distance* and *Rand Index*, which are plotted in Figure 3.4 and Figure 3.5 respectively, as they measure the regional differences. The difference between the results of K-means and merge-cluster algorithms is anticipated to be larger for *Rand Index,* since *Rand Index* algorithm does not need to find segment correspondences as opposed to *Hamming Distance* algorithm. Again, our expectation is confirmed by these two plots.

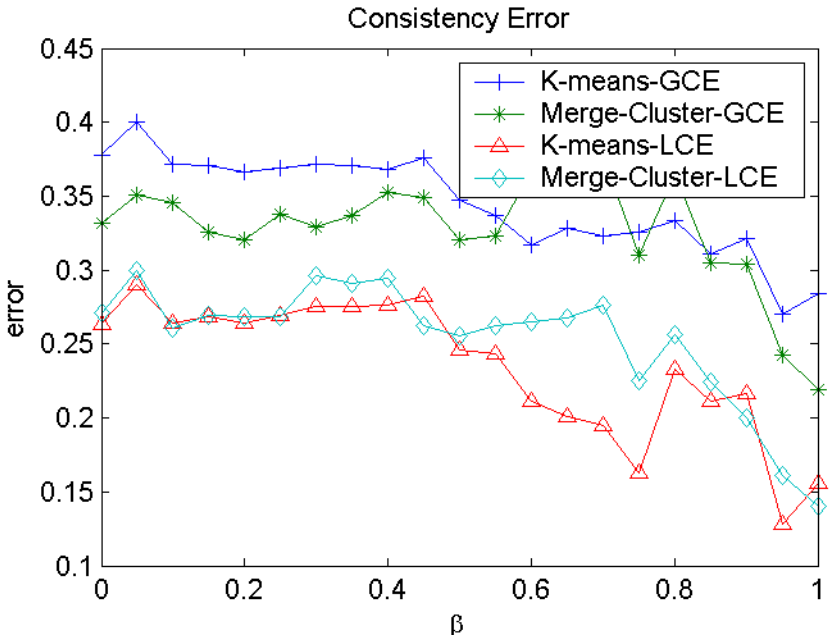

Figure 3. 6 *Horse Gallop* Consistency Error plot for the parameter set 5.

3.2.1.2 Visual Segmentation Results

In Figure 3.7, we display the initial ground truth segmentation and the ground truth segmentation after the faces are distributed to their respective patches, the latter of which is used for metric evaluation.

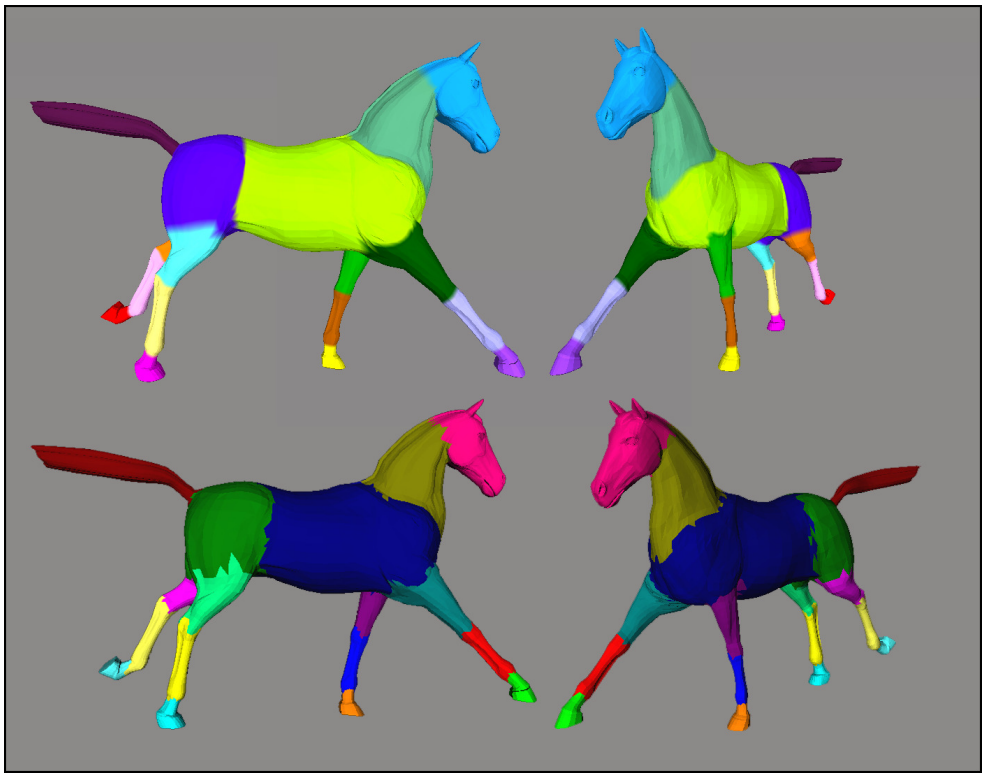

Figure 3. 7 Ground truth segmentations. Top row: The initial ground truth segmentation. Bottom row: The ground truth segmentation used for error calculation.

We expect the metric results and the visual results to match, i.e., the lower the error is, the better the visual result must be. According to the plots shown above, the lowest error is received when $\beta = 0.95$ and the error tends to be higher when β gets closer to zero. β=0 means that only spatial information is used. When β=1, spatial information is almost not

used and all information comes from temporal term. In Figure 3.8 and in Figure 3.9, we display the resulting segmentations for β=0 and β=1, respectively. Looking at these figures, we can deduce that the best visual results, in other words the results closest to the ground truth segmentation, are the ones obtained by setting β to 0.95. The visual segmentation results obtained at β=0.95 are shown in Figure 3.10. As also can be predicted looking at the error plots previously given, merge-clustering produces a better segmentation for this set of parameters. Metric error results with β values, which we have displayed the visual results for, are given in Table 3.3.

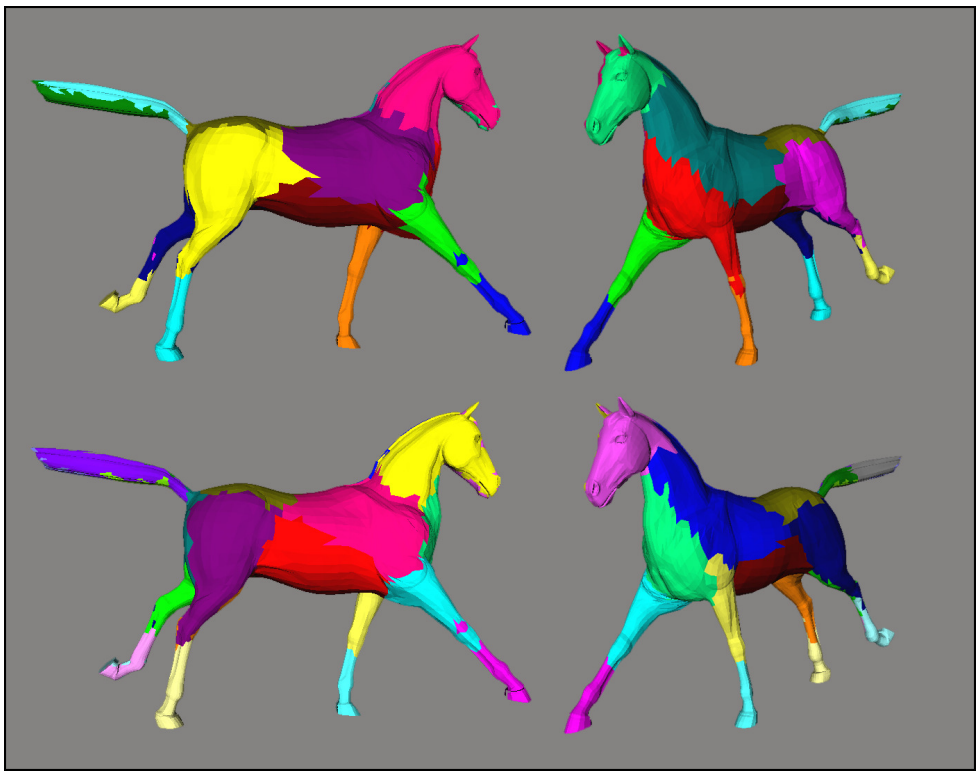

Figure 3. 8 Segmentation results for β=0. Top row: K-means. Bottom row: Merge-cluster

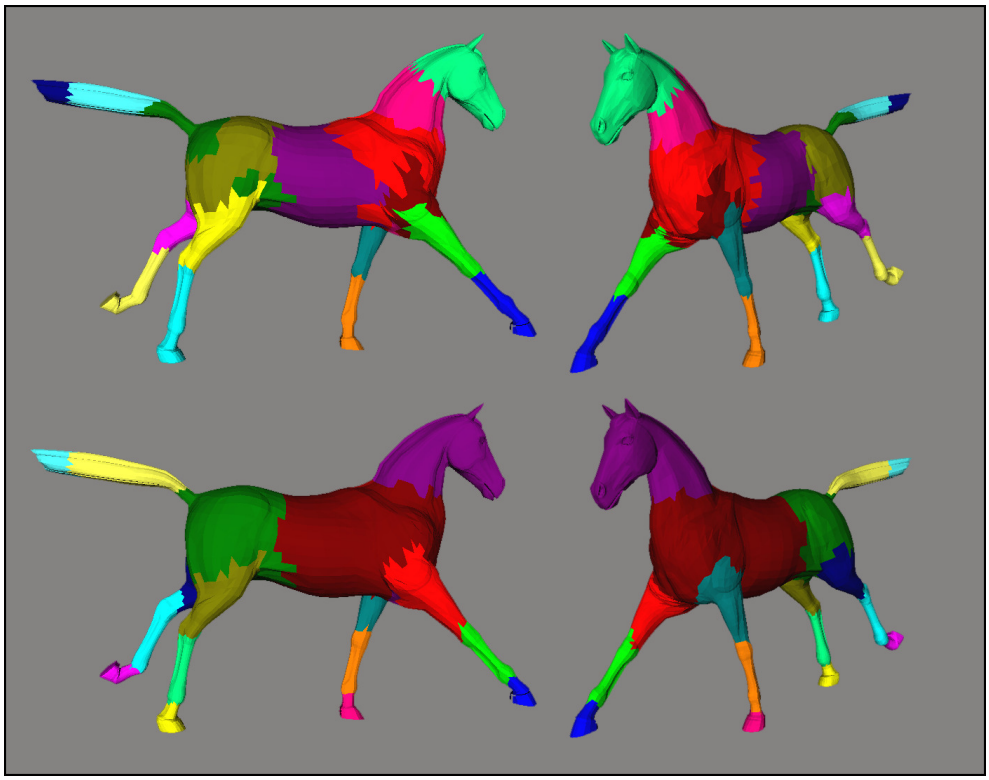

Figure 3. 9 Segmentation results for β=1. Top row: K-means. Bottom row: Merge-cluster

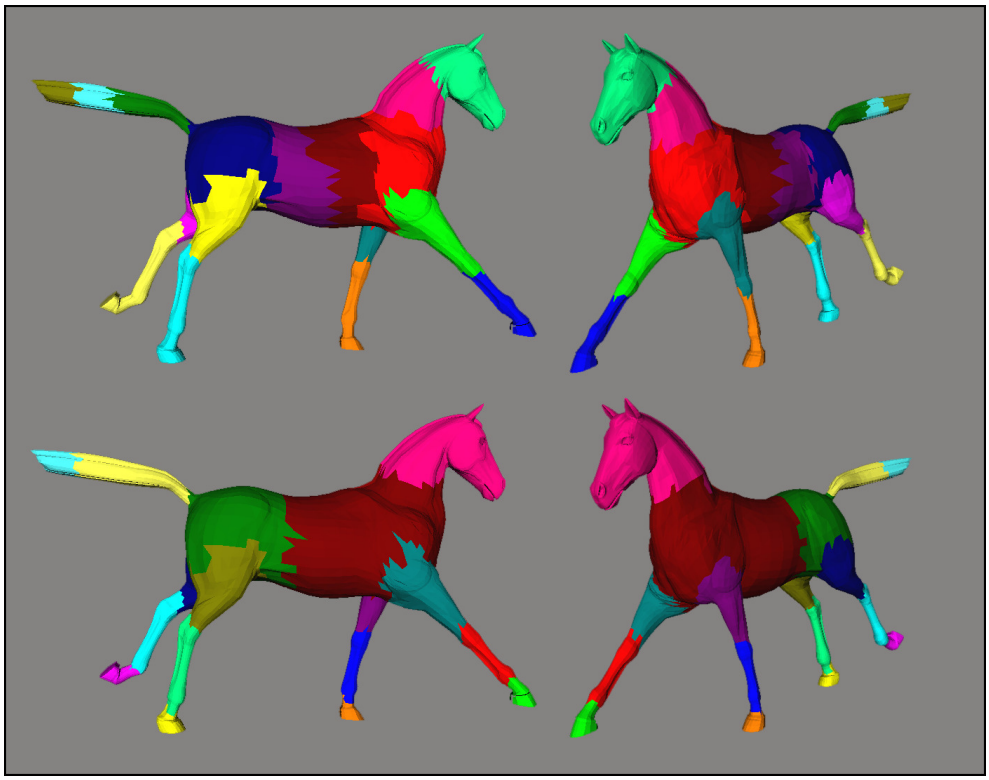

Figure 3. 10 Segmentation results for β=0.95. Top row: K-means. Bottom row: Merge-cluster

	β	CD	HD	RI	GCE	LCE
K-means	0	0.13103	0.43201	0.087819	0.37788	0.26339
	1	0.1237	0.29174	0.046893	0.28449	0.15592
	0.95	0.1614	0.30191	0.04541	0.27013	0.12832
Merge-cluster	0	0.13621	0.43509	0.085252	0.33151	0.27096
	1	0.12118	0.20485	0.051298	0.21912	0.13994
	0.95	0.11131	0.22894	0.050094	0.24223	0.16122

Table 3. 3 Metric error results for β values, which the visual results are shown for.

3.2.2 Sequence 2

Our second mesh sequence, *Jumping Man*, was acquired by Peter Sand, Leonard McMillan and Jovan Popović [13], and is the reconstruction of a real scene in which an actor jumps forward and back. The length of the sequence is 221 frames and the mesh is formed of 15830 vertices and 31660 faces. Sample frames demonstrating the motion of the model are shown in Figure 3.11. We have reduced number of vertices to 1000 base vertices by setting $\dot{r} = 0.000819$. The ground truth segmentation contains 13 segments and is displayed in visual results section.

Likewise the other sequences, we have performed segmentations using six different parameter sets on this sequence. These sets are:

1) $\mu = 2.5$ and *frameCut =10*
2) $\mu = 2.5$, *frameCut =10* and average angular distances are used.
3) $\mu = 2.5$ and *frameCut =20*
4) $\mu = 3$ and *frameCut =10*
5) $\mu = 3$, *frameCut =10* and average angular distances are used.
6) $\mu = 3$ and *frameCut =20*

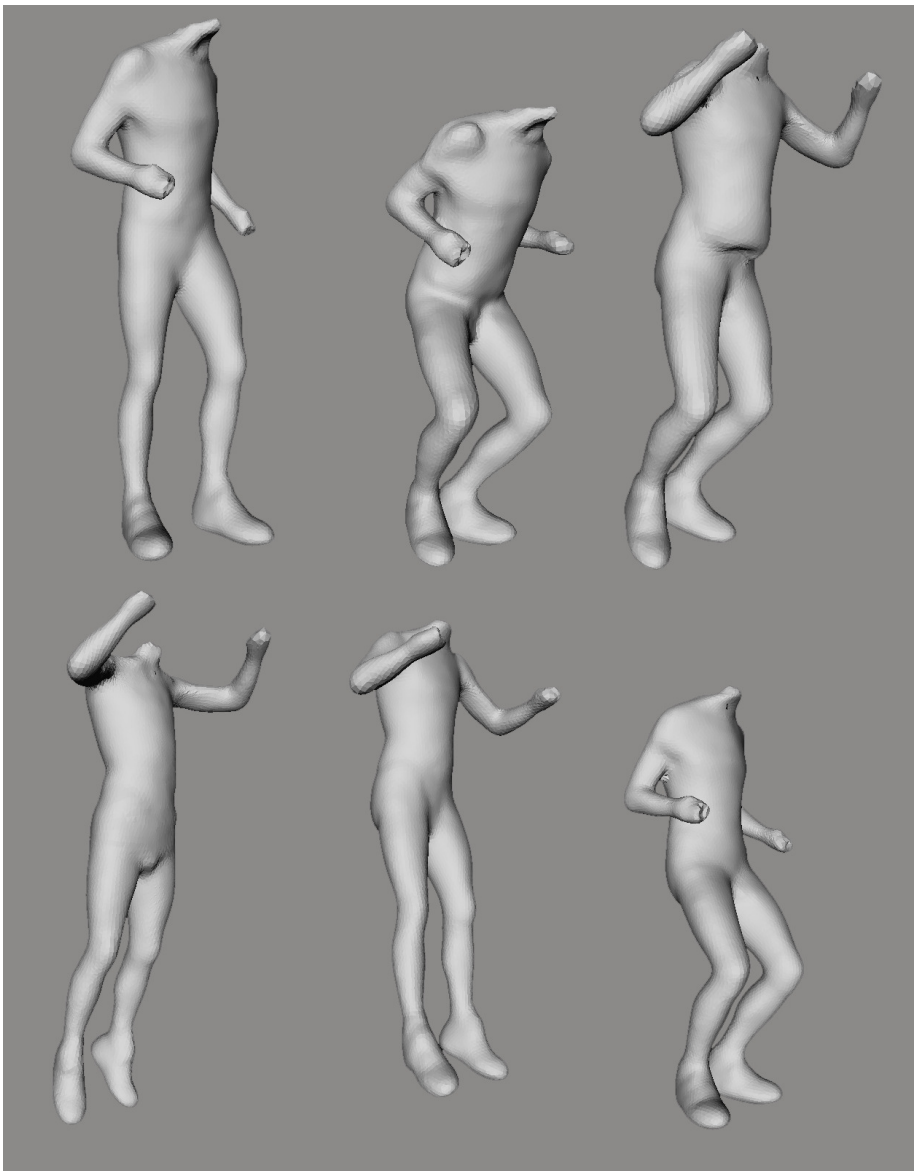

Figure 3. 11 Six frames of the *Jumping Man* sequence depicting the motion of the model.

3.2.2.1 Metric Results

In Figure 3.12, we plot the segmentation errors for each parameter set and for each error metric for varying values of β, with 0.05 increments in the interval $0 \leq \beta \leq 1$. Recall that the segmentation error in each case is computed as the sum of the two errors, one obtained with K-means clustering and the other obtained using merge-cluster algorithm. In Table 3.3, we provide the number of segments produced by our merge-cluster algorithm for each parameter set and for each β value. Recall that the number of clusters is fixed for K-means.

We observe that low metric error values do not always necessarily imply good visual segmentation results. For example, one can clearly see that, for parameter sets 4,5 and 6, the resulting numbers of segments are unacceptably high. However, the corresponding error values do not indicate that the segmentation results for these sets are poor. Although Hamming Distance values stand as an evidence of these poor results, other error metric values fail to do so. The main reason of these unsuccessful results is that clusters cannot simply merge because of the low threshold value we set as $\mu=3$. Therefore, we can discard the parameter sets 4,5 and 6 from further analysis. In Table 3.4, we provide the average error values computed over β for the rest of the parameter sets.

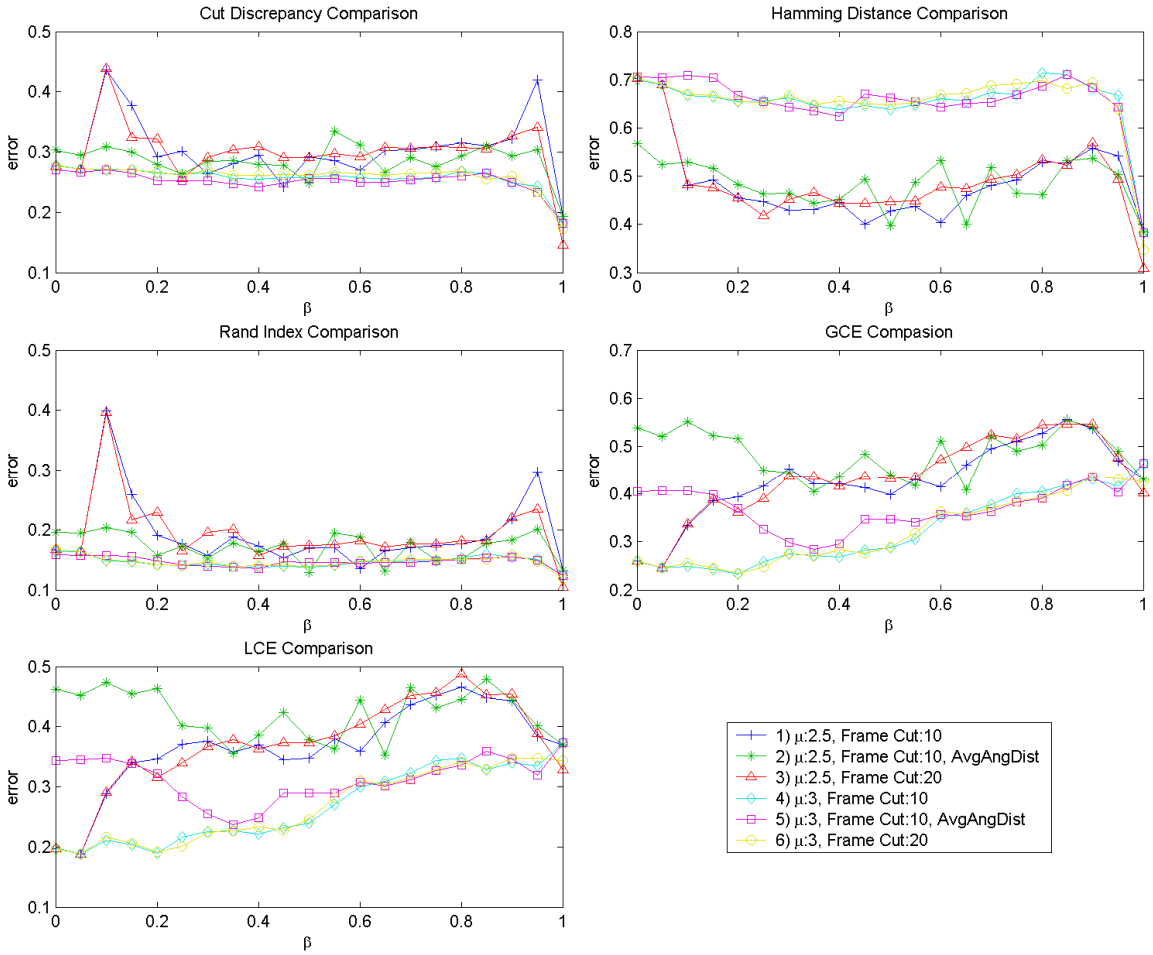

Figure 3. 12 Segmentation errors obtained for the *Jumping Man* sequence under different parameter settings.

β	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
0	999	9	999	999	250	999
0.05	999	9	999	999	232	999
0.10	9	10	9	999	251	999
0.15	9	9	9	999	246	999
0.20	9	10	8	999	226	999
0.25	9	9	9	691	293	739
0.30	12	9	9	593	308	613
0.35	10	9	9	515	328	556
0.40	9	9	9	461	264	545
0.45	11	9	10	450	252	498
0.50	9	11	10	377	251	411
0.55	10	8	10	332	256	318
0.60	9	9	10	245	190	226
0.65	9	10	10	203	204	261
0.70	9	9	10	230	201	266
0.75	9	10	9	183	200	246
0.80	9	10	9	213	218	234
0.85	10	10	10	206	191	173
0.90	8	9	8	147	148	163
0.95	8	9	9	148	145	108
1.00	8	8	10	10	10	10

Table 3. 4 Number of segments resulting from each parameter set for the *Jumping Man* sequence.

	CD	HD	RI	GCE	LCE
Set 1	0.30286	0.48676	0.19142	0.42753	0.36523
Set 2	0.28581	0.48361	0.17185	0.48454	0.42113
Set 3	0.30066	0.49074	0.19235	0.43387	0.36997

Table 3. 5 Average total error values calculated over β for the *Jumping Man* sequence.

For all three parameter sets, average error values are very close to each other. We choose to proceed with set 2, which is the winner of *Cut Discrepancy, Hamming Distance* and *Rand Index*, and which produces reasonable numbers of segments for β=0 and β=0.05, unlike sets 1 and 3.

In Figure 3.13, we plot the *Cut Discrepancy* error results. We observe that, since *merge-cluster* algorithm may not generate the exact number of segments given by the ground truth segmentation, *Cut Discrepancy* values come out to be significantly higher for merge-clustering results than they are for K-means clustering.

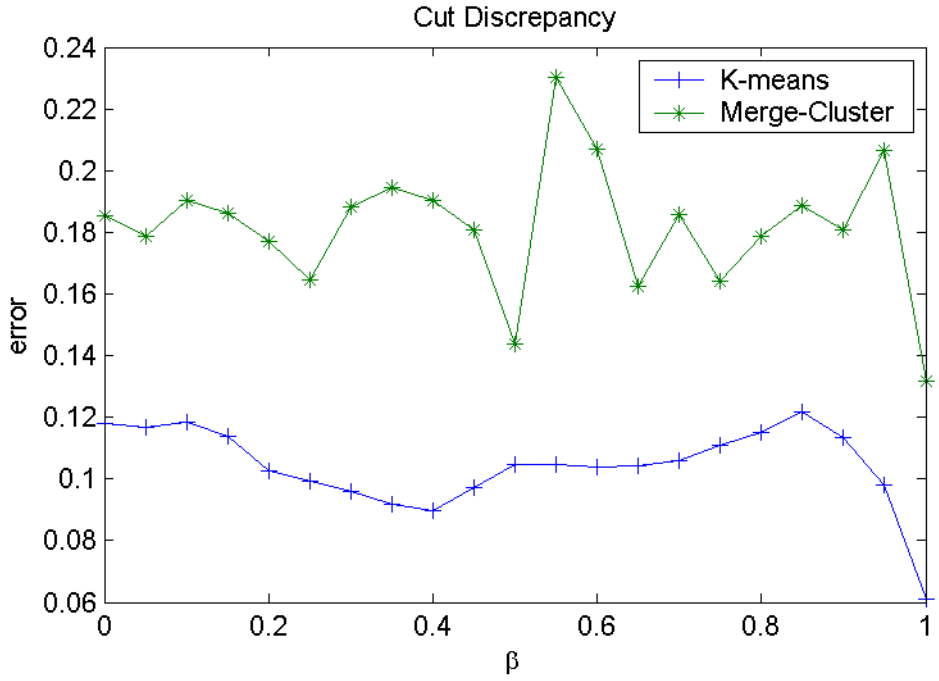

Figure 3. 13 *Jumping Man* Cut Discrepancy error plot for the parameter set 2.

We show *Hamming Distance* and *Rand Index* error values in Figure 3.14 and in Figure 3.15, respectively. As we have already discussed in Chapter 2, K-means clustering is sensitive to the choice of initial cluster centers. When each initial center is placed in proximity of a different segment, K-means clustering tends to produce better results. In the case of *Jumping Man*, our initial center selection is successful enough to distribute the centers in a semantically reasonable way as shown in Figure 3.16, and thus, K-means clustering yields better segmentations in terms of both *Hamming Distance* and *Rand Index* metrics.

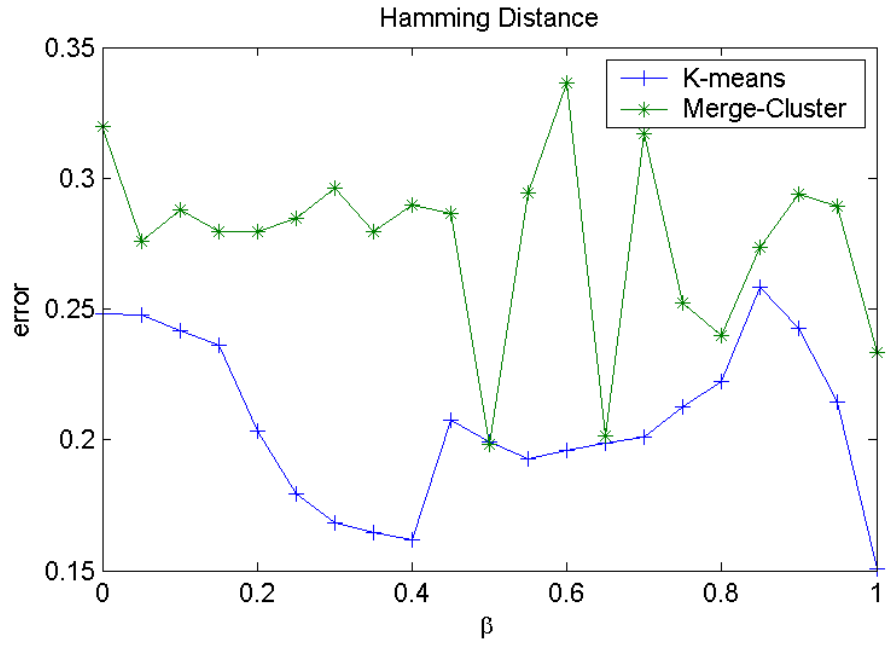

Figure 3. 14 *Jumping Man* Hamming Distance error plot for the parameter set 2.

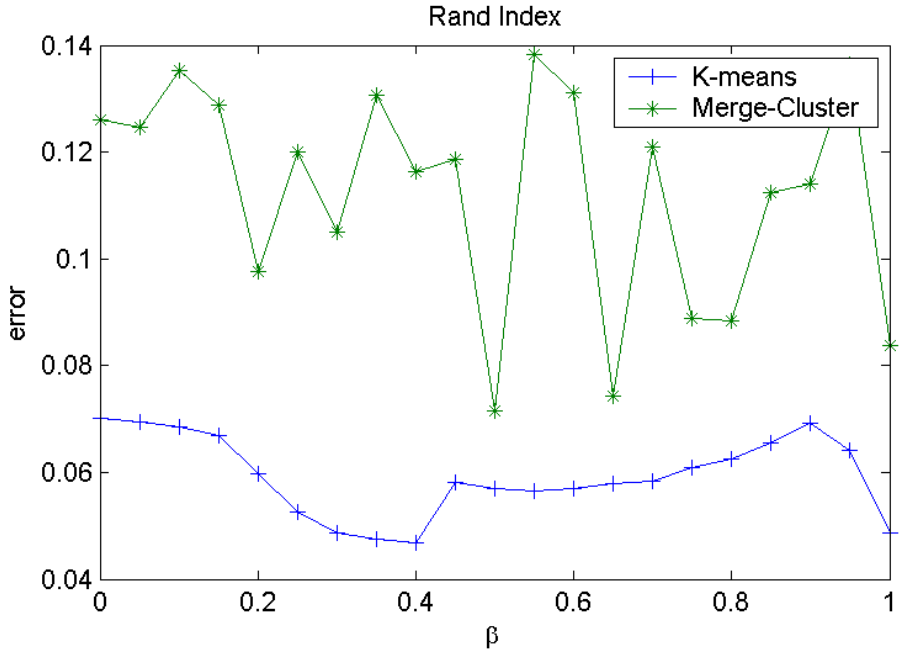

Figure 3. 15 *Jumping Man* Rand Index error plot for the parameter set 2.

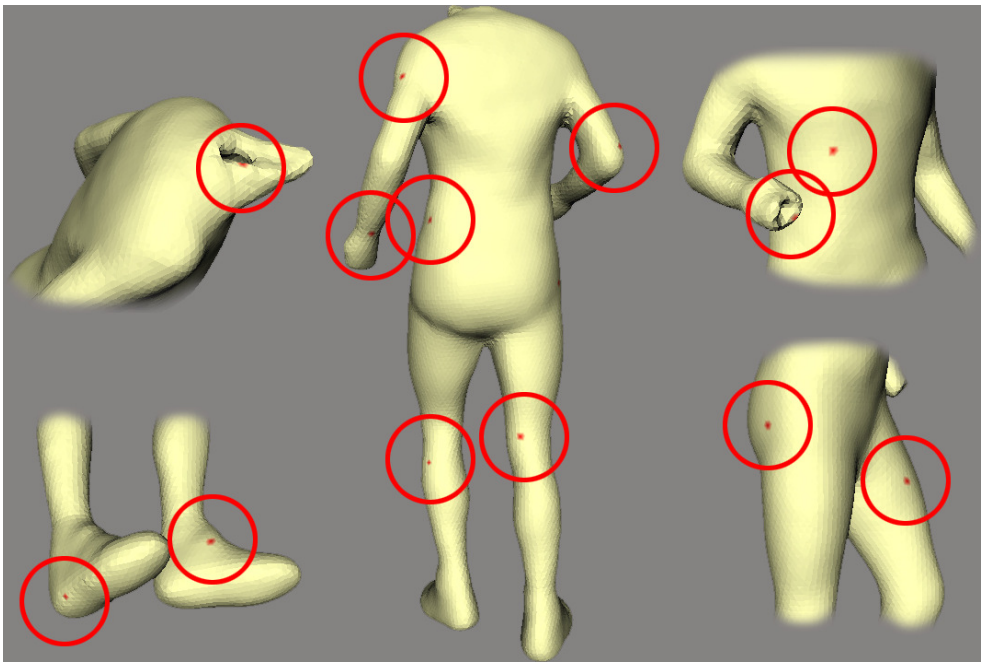

Figure 3. 16 *Jumping Man.* Distribution of initial centers used by K-means.

Figure 3.17 shows the *Consistency Error* plots. This metric is sensitive to the number of segments and is expected to produce better results when the numbers of segments of the compared segmentations are different. Table 3.3 shows that the number of segments yielded by merge-cluster is always different than the number of ground truth segments. This is the reason why merge-cluster seems to have done a better job than K-means. Therefore, we will not use this metric to compare K-means and merge-clustering but to compare segmentation results for different β values.

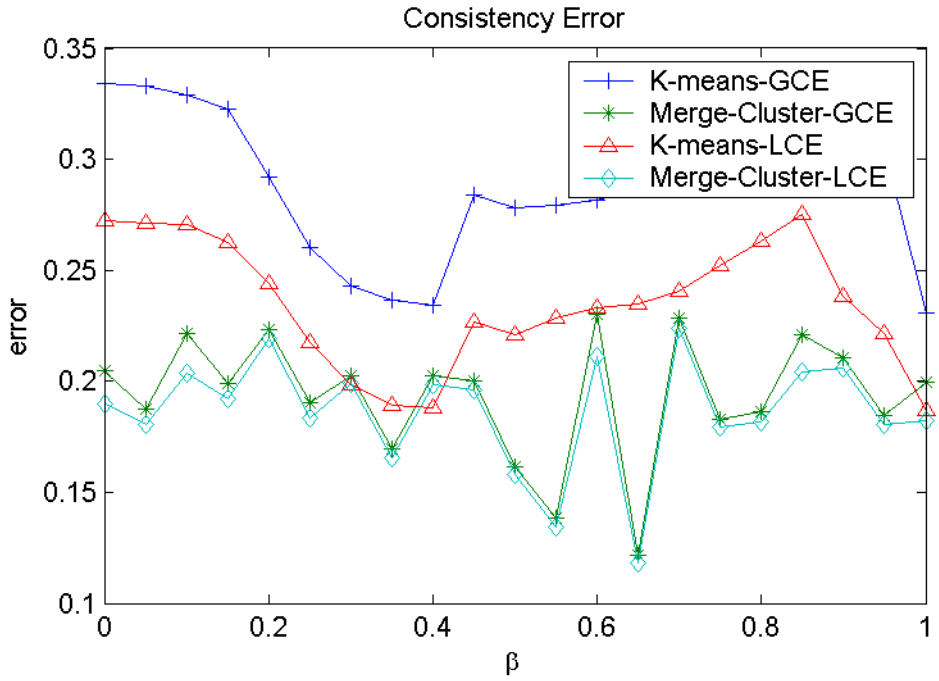

Figure 3. 17 *Jumping Man* Consistency Error plot for the parameter set 2.

3.2.2.2 Visual Segmentation Results

In Figure 3.18, we exhibit the initial ground truth segmentation along with the ground truth segmentation which is used for error metric calculation. Recall that the ground truth segmentation used for metric evaluation is obtained by distributing faces to their respective patches.

In Figure 3.19 and 3.20, we display the results for two extreme cases where β=0 and β=1, i.e., when we make use of only spatial information and when we make use of only temporal information, respectively. Error plots indicate that, for β=0.4 and β=1, K-means clustering yields very close error values, which are also the lowest error values achieved. For merge-clustering, the lowest error values on average are received when β=0.5. For the

mentioned β values, the visual results are displayed in Figure 3.21 and the metric error results for β=0, β=1, β=0.4 and β=0.5 are given in Table 3.5.

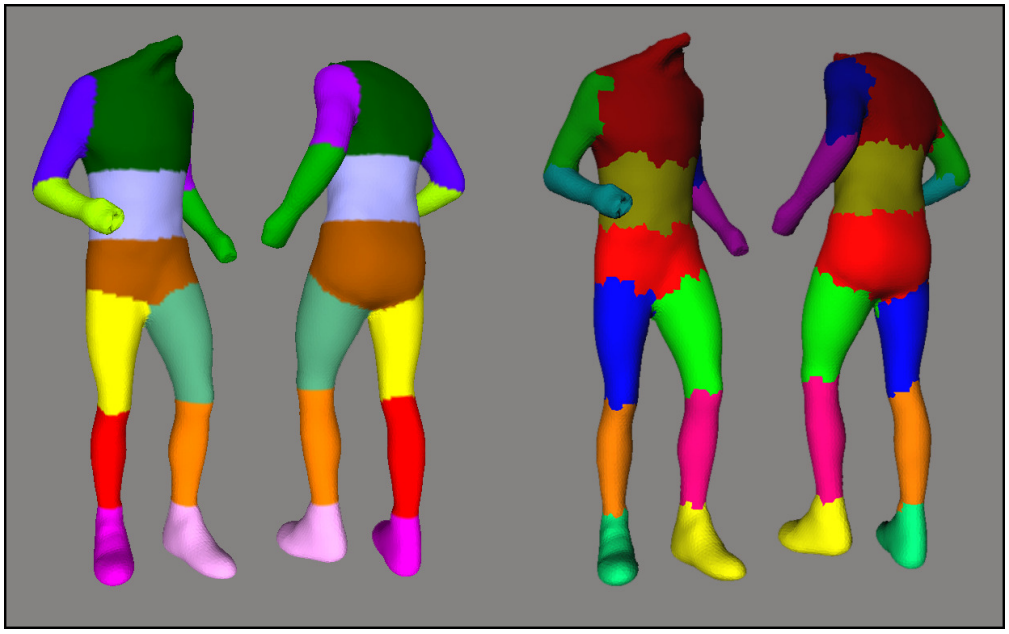

Figure 3. 18 Ground truth segmentations. Left: The initial ground truth segmentation. Right: The ground truth segmentation used for error calculation.

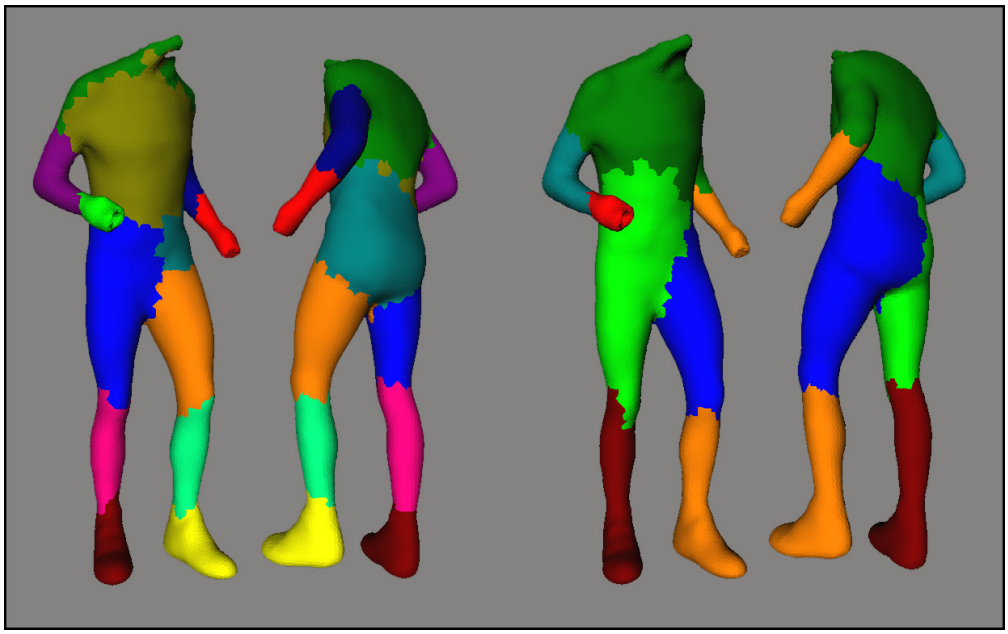

Figure 3. 19 Segmentation results for β=0. Left: K-means. Right: Merge-cluster.

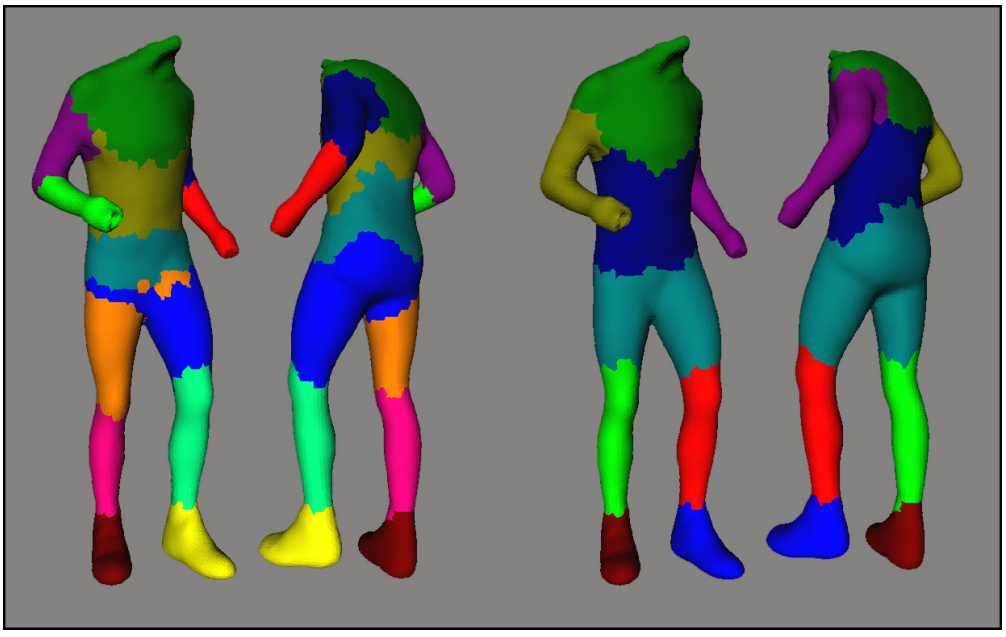

Figure 3. 20 Segmentation results for β=1. Left: K-means. Right: Merge-cluster.

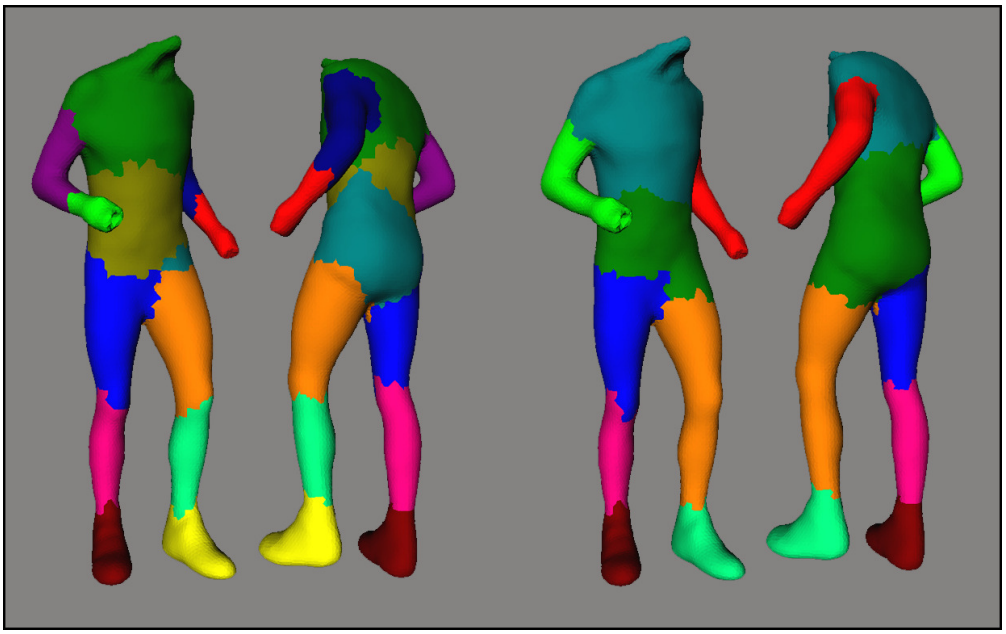

Figure 3. 21 Segmentation results. Left: K-means with β=0.4. Right: Merge-cluster with β=0.5.

	β	CD	HD	RI	GCE	LCE
K-means	0	0.1179	0.24823	0.070067	0.33402	0.2723
	1	0.061222	0.1507	0.048577	0.23058	0.18714
	0.4	0.089517	0.16173	0.046754	0.23419	0.1878
Merge cluster	0	0.18534	0.31956	0.12602	0.20467	0.19006
	1	0.13179	0.19826	0.083807	0.19986	0.18209
	0.5	0.1439	0.23343	0.071478	0.16162	0.1578

Table 3. 6 Metric error results for β values, which the visual results are shown for.

3.2.3 Sequence 3

Our third sequence, *Dancing Woman*, is reconstructed by the authors of [29] from a real scene showing a woman exhibiting dance figures. A number of sample frames depicting the motion of the woman are displayed in Figure 3.22. The sequence has 217 frames. Throughout the frames, 15002 vertices and 30000 faces are in fixed and known connectivity. The number of vertices is reduced to 996 base vertices using $\dot{r} = 0.000719$. The mesh is divided into 15 segments manually to form the ground truth segmentation. The particularity of this sequence is that it does not have a smooth surface and possesses many concavities due to its extremely realistic surface details. Without motion information incorporated into the segmentation process, we expect these cavities to aggravate the segmentation process.

We have tested our algorithm using six different parameter sets, which are:

1) $\mu = 2.5$ and *frameCut=10*

2) $\mu = 2.5$, *frameCut =10* and average angular distances are used.

3) $\mu = 2.5$ and *frameCut =20*

4) $\mu = 3$ and *frameCut =10*

5) $\mu = 3$, *frameCut =10* and average angular distances are used.

6) $\mu = 3$ and *frameCut =20*

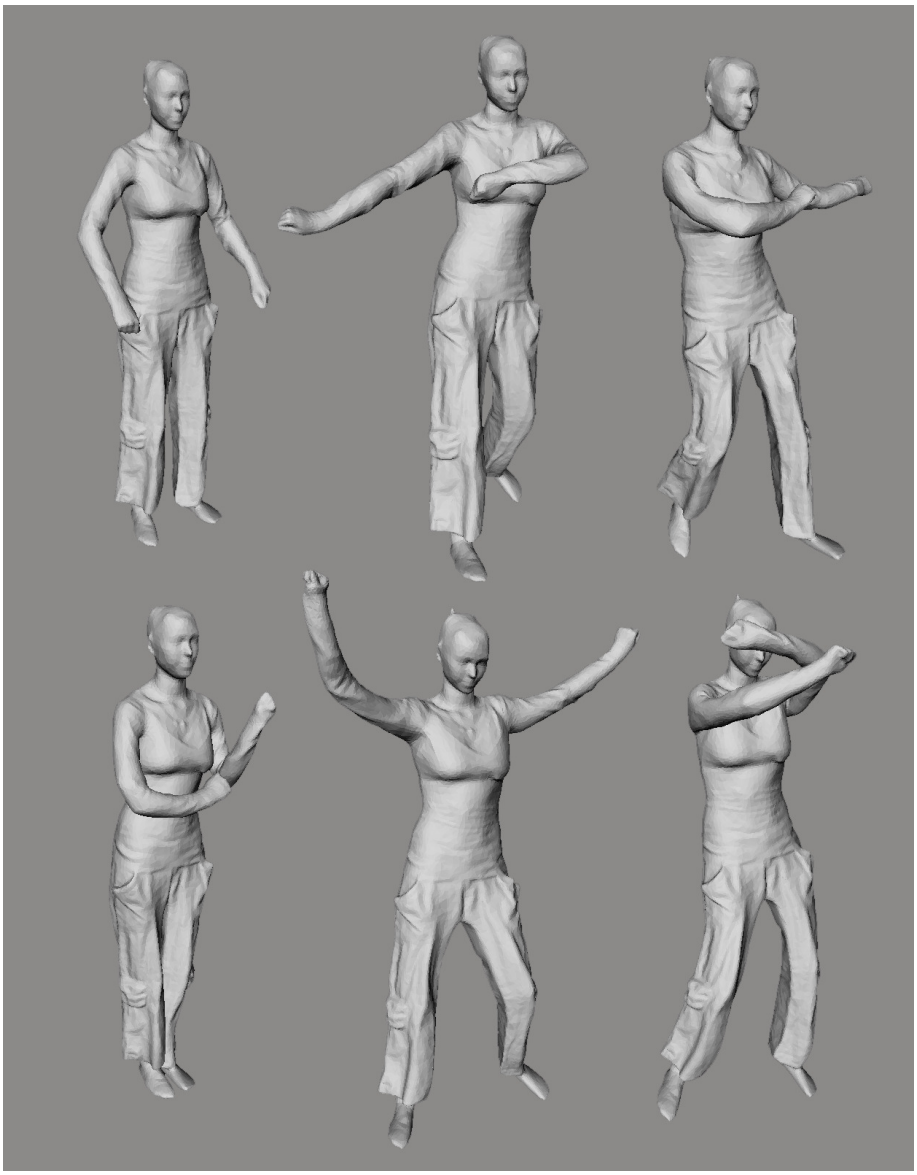

Figure 3. 22 Six frames of the Dancing Woman sequence depicting the motion of the model.

3.2.3.1 Metric Results

In Figure 3.23, we display the plots of segmentation errors obtained using different parameter sets and error metrics for varying β with 0.05 increments in the interval $0 \le \beta \le 1$. As observed from Table 3.7 the merge-cluster algorithm produced very reasonable results in terms of resulting segment numbers, therefore we will consider all parameter sets for further evaluation. We present the average error values calculated over β for each parameter set and each error metric in Table 3.8. Looking at the error plots and the average error values, we can conclude that the best results are obtained using the parameter set 5.

β	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
0	11	14	11	19	18	19
0.05	12	14	14	19	21	21
0.10	14	14	14	22	14	21
0.15	16	9	13	16	16	16
0.20	10	8	8	13	12	13
0.25	9	12	13	15	13	14
0.30	12	13	12	14	16	17
0.35	12	13	10	13	14	15
0.40	11	10	12	13	13	14
0.45	11	9	12	13	14	13
0.50	12	13	12	14	14	14
0.55	12	9	11	15	14	14
0.60	12	9	9	13	15	15
0.65	10	10	13	13	14	13
0.70	13	11	13	14	14	14
0.75	13	9	14	14	14	13
0.80	12	10	11	12	13	13
0.85	11	10	12	13	13	12
0.90	10	10	9	11	11	12
0.95	11	9	12	13	12	14
1.00	10	10	10	12	12	13

Table 3. 7 Number of segments yielded by each parameter set for the *Dancing Woman* sequence.

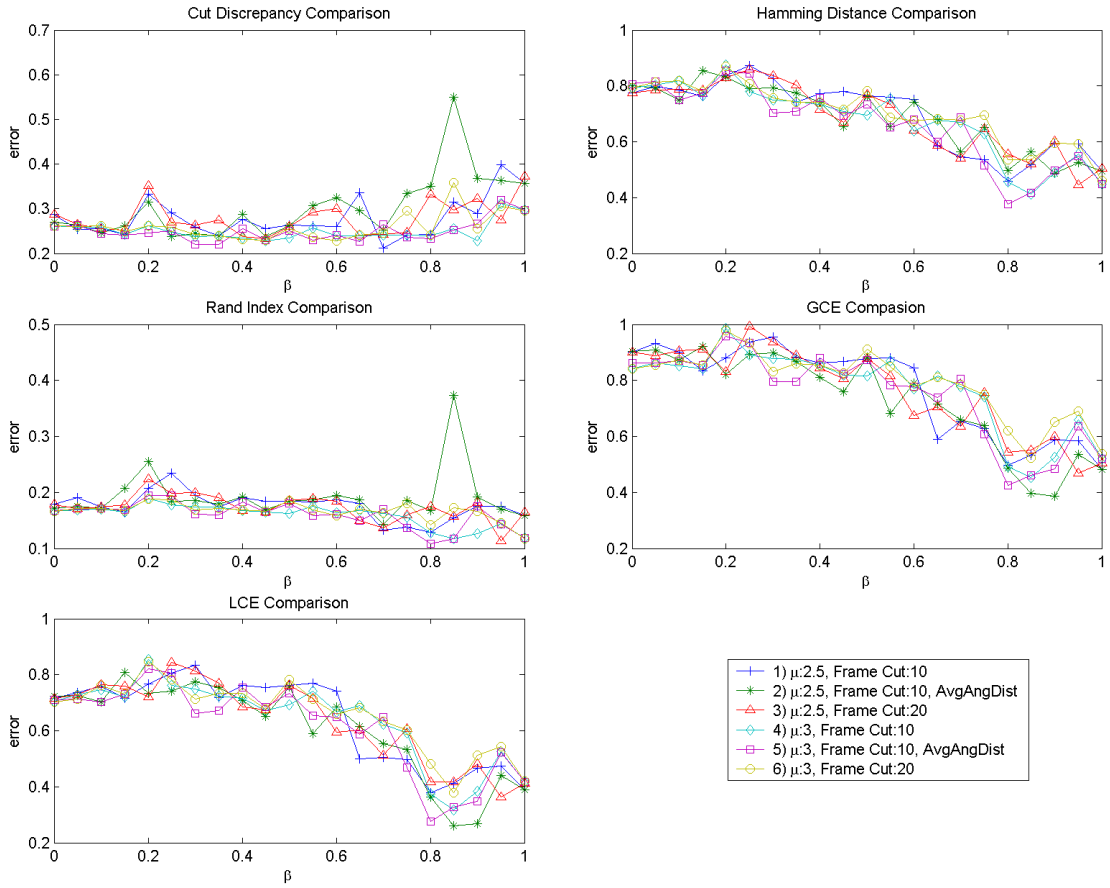

Figure 3. 23 Segmentation errors obtained for the *Dancing Woman* sequence under different parameter settings.

	CD	HD	RI	GCE	LCE
Set 1	0.27946	0.69255	0.17575	0.76718	0.64133
Set 2	0.30322	0.68578	0.19207	0.72919	0.60855
Set 3	0.27989	0.68547	0.1735	0.76427	0.63629
Set 4	0.25039	0.67611	0.159	0.76874	0.63832
Set 5	0.24999	0.65985	0.15995	0.74976	0.61376
Set 6	0.25972	0.70278	0.16642	0.79115	0.65758

Table 3. 8 Average error values calculated over β for the *Dancing Woman* sequence.

In Figure 3.24, *Cut Discrepancy* error plot for parameter set 5 is given. For this model and parameter set, the performance of merge-cluster algorithm is observed to be successful in terms of the produced cluster number, hence the error values for both K-means and merge-cluster stay in close proximity.

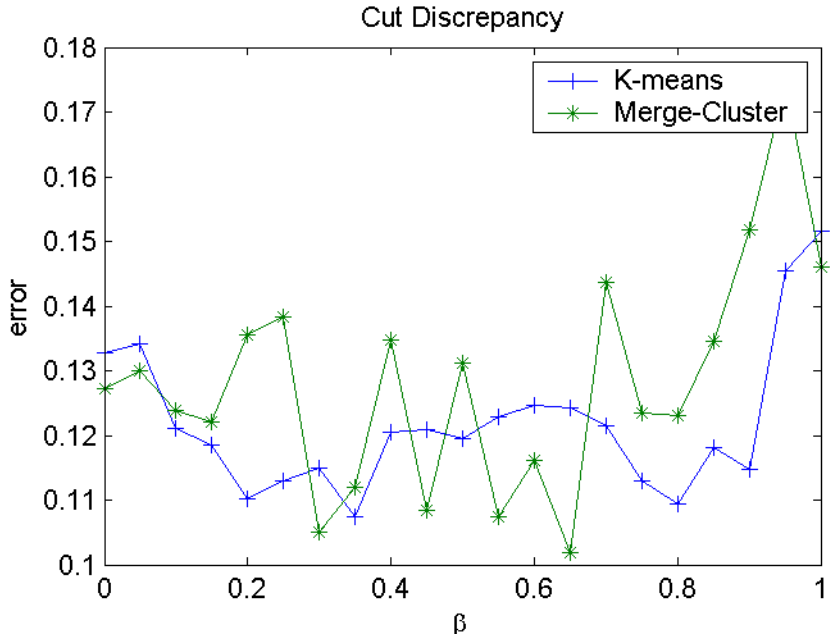

Figure 3. 24 *Dancing Woman* Cut Discrepancy error plot for the parameter set 5.

Hamming Distance and *Rand Index* error plots are shown in Figure 3.25 and Figure 3.26, respectively. Looking at these plots, we observe an unstable behavior for the performance of the merge-cluster algorithm in terms of error values. In Figure 3.27, we display the *Consistency Error* plot, which shows a similar behavior as the plots of the other metrics..

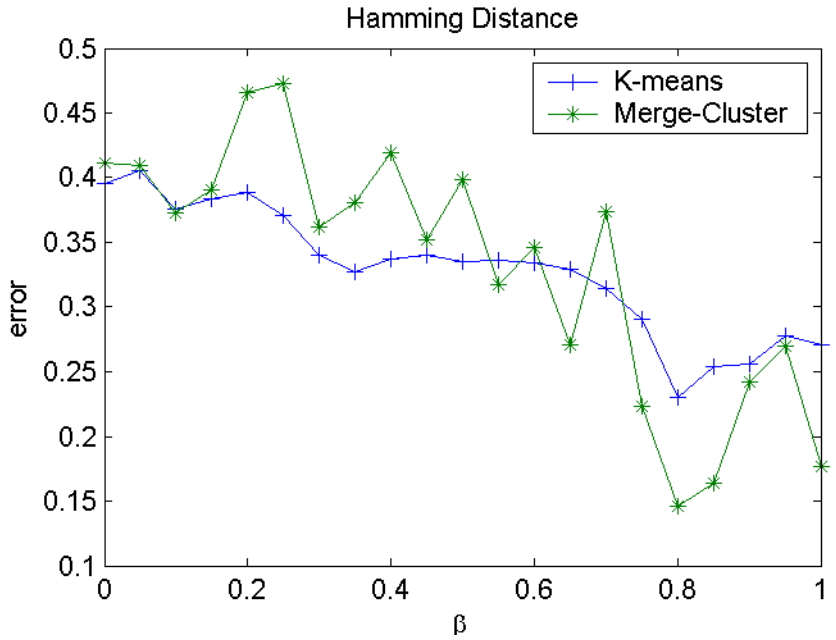

Figure 3. 25 *Dancing Woman* Hamming Distance error plot for the parameter set 5.

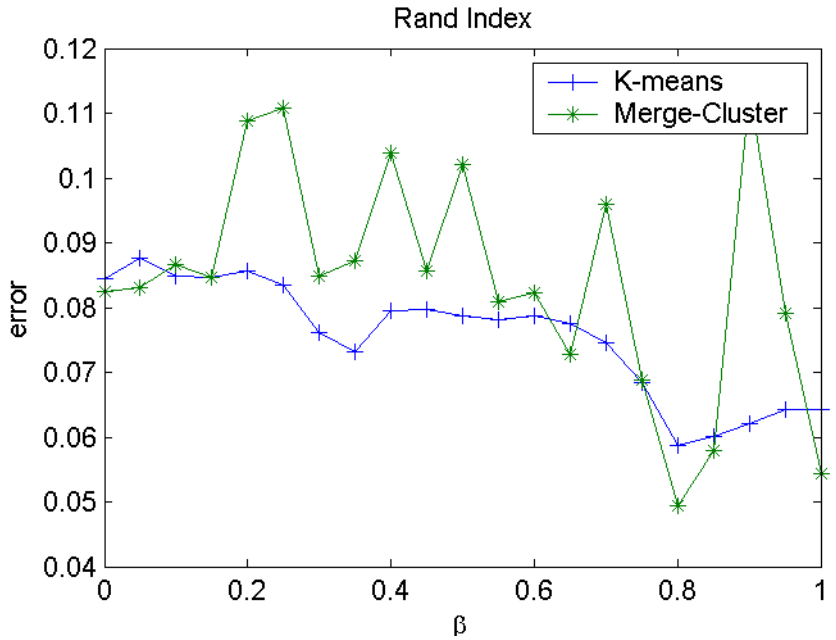

Figure 3. 26 *Dancing Woman* Rand Index error plot for the parameter set 5.

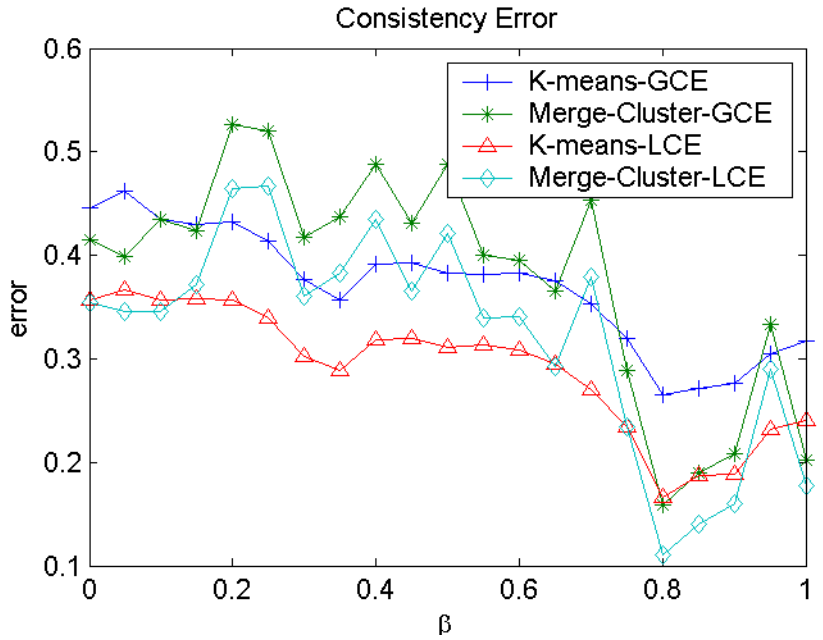

Figure 3. 27 *Dancing Woman* Consistency Error plot for the parameter set 5.

3.2.3.2 Visual Segmentation Results

We first display, in Figure 3.28, the initial ground truth segmentation and the ground truth segmentation obtained after the faces are distributed according to their patches.

Two extreme cases for β=0 and β=1 are shown Figure 3.29 and Figure 3.30. Looking at the *Hamming Distance, Rand Index* and *Consistency Error* plots, we see that the minimum error values are achieved at β=0.8. For this β value we show the visual results in Figure 3.31. In order to provide a better sight, in Table 3.9, the error values for β=0, β=1 and β=0.8 are given.

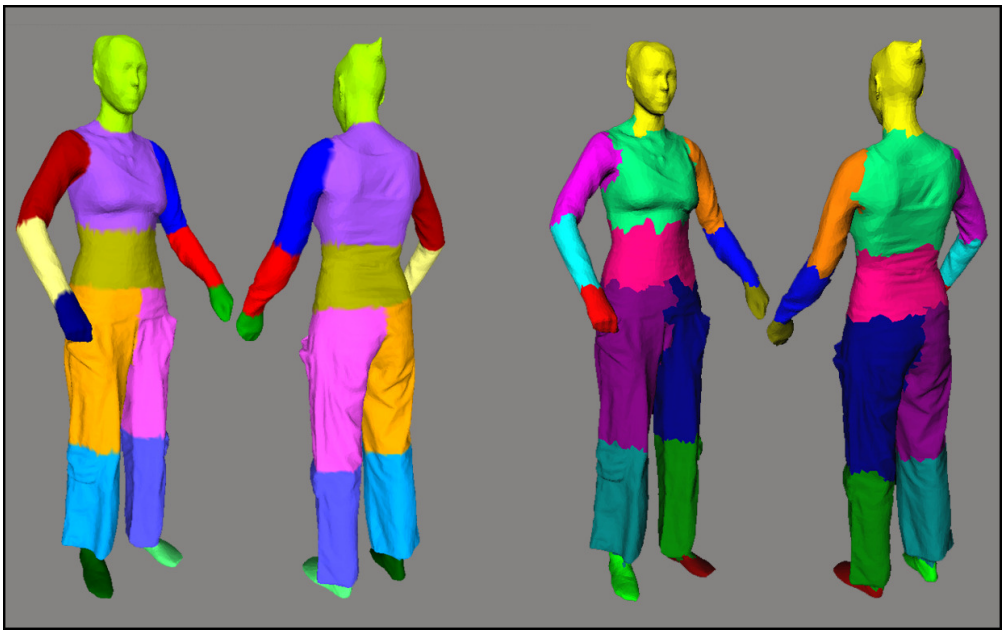

Figure 3. 28 Ground truth segmentations. Left: The initial ground truth segmentation. Right: The ground truth segmentation used for error calculation.

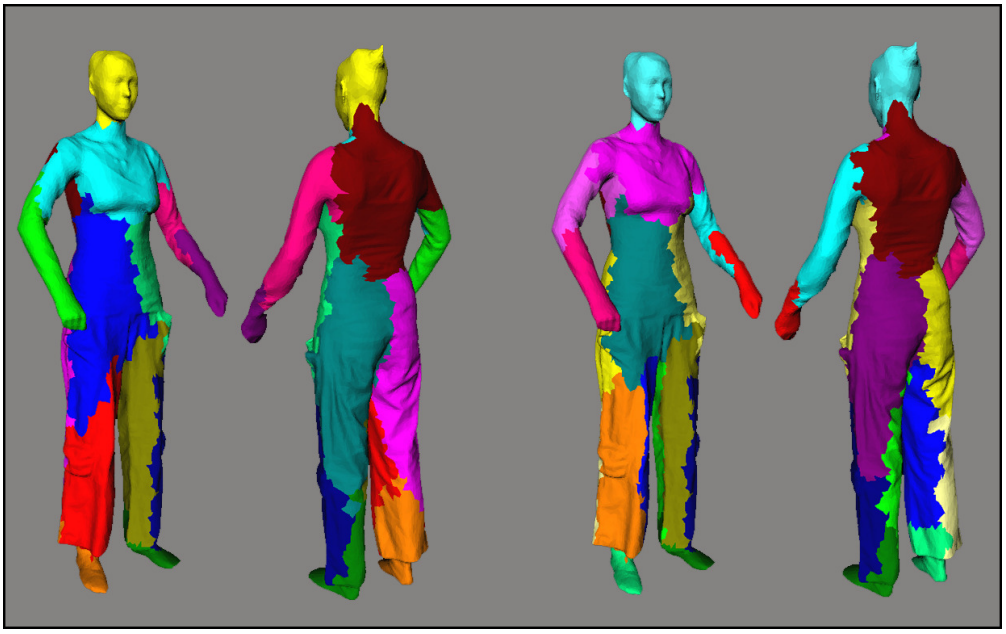

Figure 3. 29 Segmentation results for β=0. Left: K-means. Right: Merge-cluster.

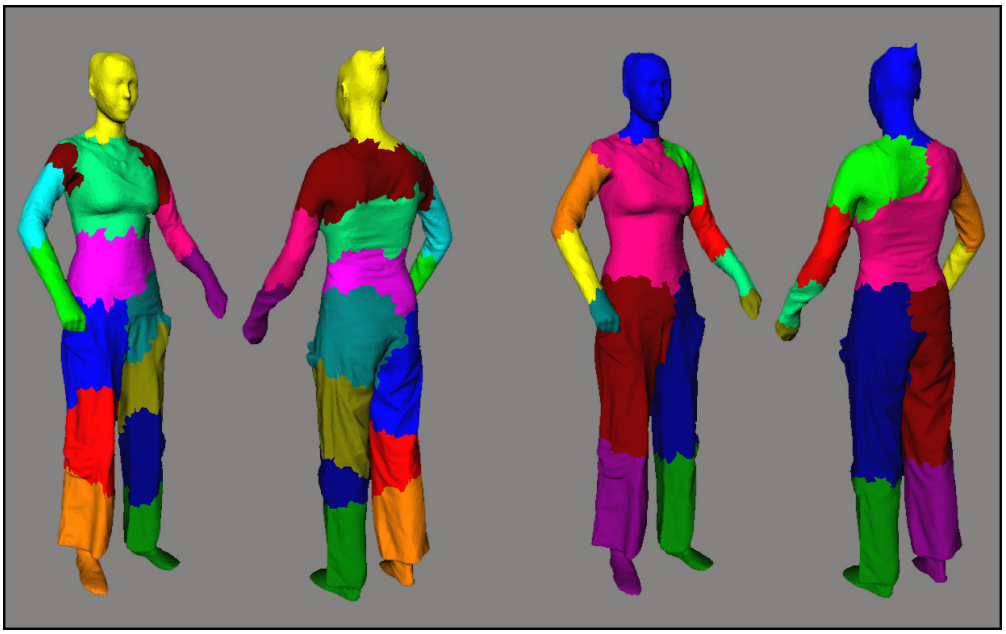

Figure 3. 30 Segmentation results for β=1. Left: K-means. Right: Merge-cluster.

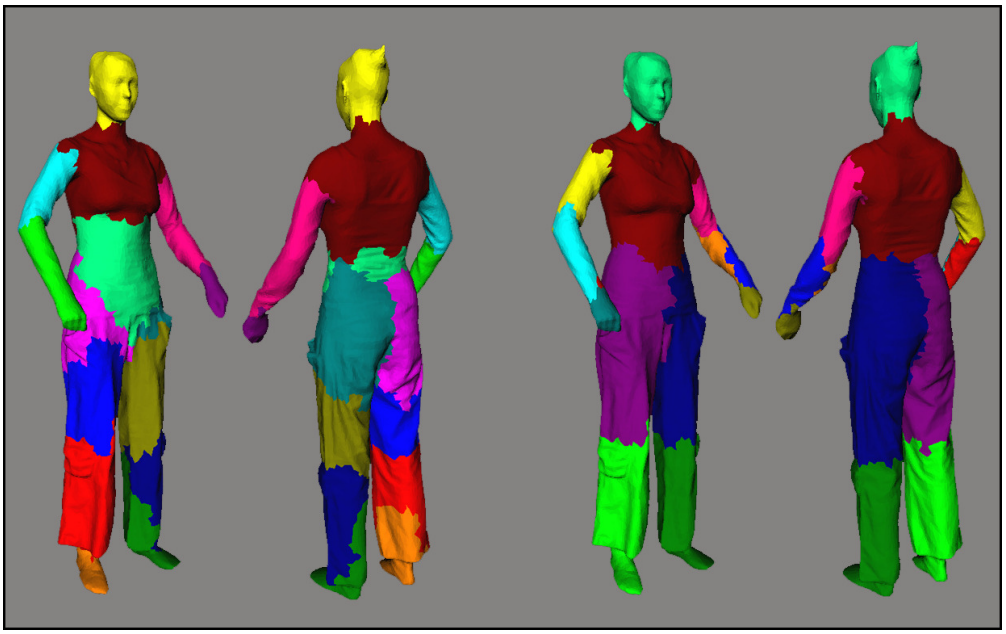

Figure 3. 31 Segmentation results for β=0.8. Left: K-means. Right: Merge-cluster.

	β	CD	HD	RI	GCE	LCE
K-means	0	0.13272	0.39543	0.084432	0.44584	0.35639
	1	0.1515	0.27068	0.064216	0.31723	0.24025
	0.8	0.10948	0.23013	0.058634	0.2657	0.16628
Merge-cluster	0	0.12727	0.41155	0.082503	0.41529	0.35425
	1	0.14597	0.17689	0.054315	0.20188	0.17713
	0.8	0.12313	0.1459	0.049475	0.15925	0.11045

Table 3. 9 Metric error results for β values, which the visual results are shown for.

3.2.4 Sequence 4

Our fourth and last sequence, *Dancing Man*, was made available by the authors of [29] from a real scene, showing the dancing moves of an actor. A number of sample frames out of 250 frames demonstrating the motion of the actor are shown in Figure 3.32. The mesh is composed of 19988 vertices and 39972 faces and the model is of fixed and known connectivity. We have downsampled the model to 1000 base vertices by setting $\dot{r} = 0.00078$. The number of segments in the ground truth segmentation is 16. *Dancing man* and *dancing* woman sequences have quite similar features in terms of surface structure. The wavy nature of the surface of the mesh is expected to cause the segmentations performed with only spatial information to yield unsuccessful results.

We tested our algorithm using six different parameter sets, which are:

1) $\mu = 2.5$ and *frameCut=10*

2) $\mu = 2.5$, *frameCut =10* and average angular distances are used.

3) $\mu = 2.5$ and *frameCut =20*

4) $\mu = 3$ and *frameCut =10*

5) $\mu = 3$, *frameCut =10* and average angular distances are used.

6) $\mu = 3$ and *frameCut =20*

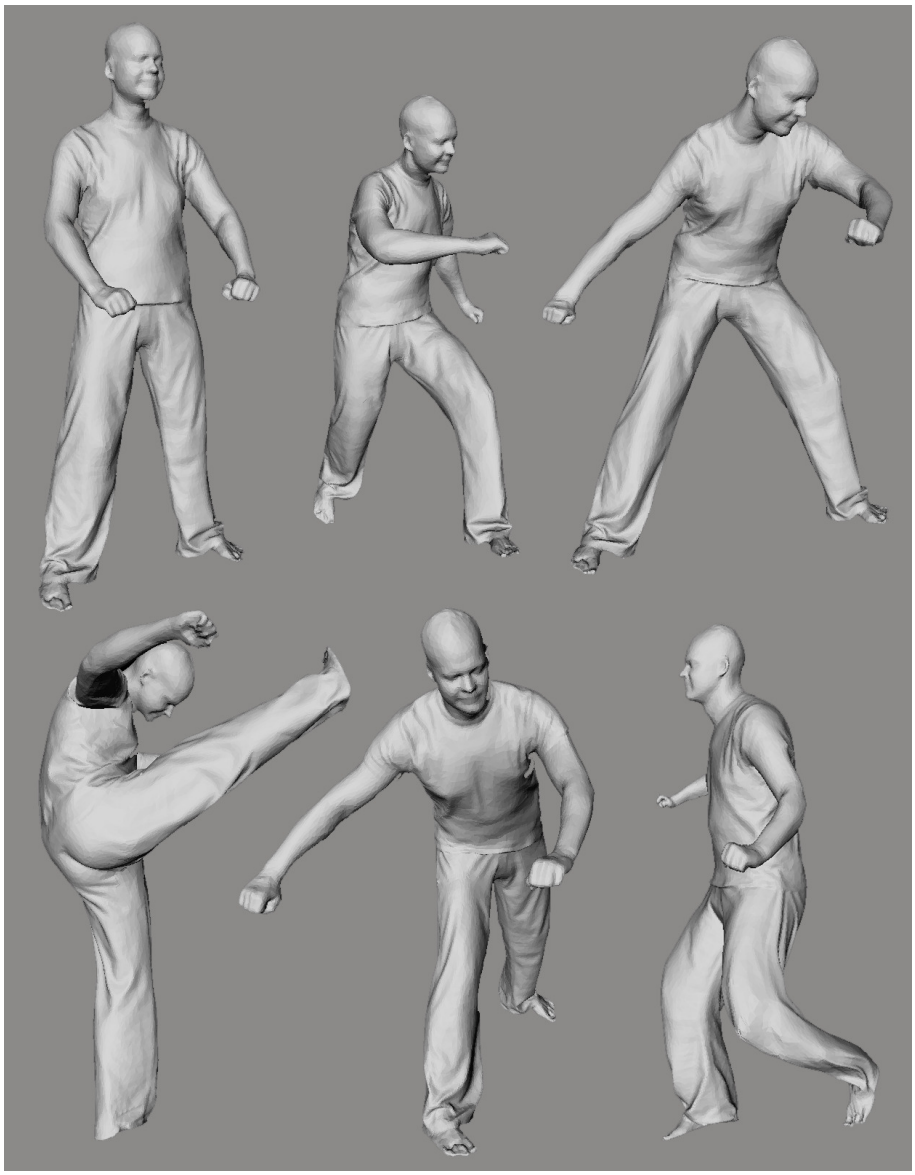

Figure 3. 32 Six frames of Dancing Man sequence depicting the motion of the model.

3.2.4.1 Metric Results

In Figure 3.33, we display the segmentation error plots for different error metrics and for different parameter sets with varying β in the interval $0 \le \beta \le 1$, where the error values are the sum of K-means and merge-cluster error values. We provide the resulting segment number for each parameter set in Table 3.10. For this specific sequence, it is difficult choose a winner parameter set by looking at the error values and the average error values given in Table 3.11, since the error values are very close to each other. Therefore, we will continue to our evaluation using parameter set 2 since the best visual results are produced by this set.

β	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
0	14	8	14	18	15	18
0.05	13	9	13	16	15	15
0.10	9	12	14	15	14	15
0.15	14	14	9	16	14	16
0.20	12	9	14	15	14	15
0.25	10	15	10	16	14	15
0.30	14	14	10	16	14	15
0.35	9	14	9	15	14	14
0.40	9	14	10	15	14	15
0.45	9	14	13	14	14	13
0.50	8	13	14	14	14	12
0.55	14	14	9	13	14	13
0.60	10	14	9	14	16	14
0.65	10	13	11	14	16	13
0.70	9	10	10	14	14	13
0.75	10	9	10	14	15	12
0.80	11	9	11	13	17	12
0.85	10	12	11	13	16	13
0.90	10	9	8	14	14	18
0.95	11	10	9	16	16	17
1.00	12	12	11	15	15	15

Table 3. 10 Number of segments yielded by each parameter set for *Dancing Man* sequence.

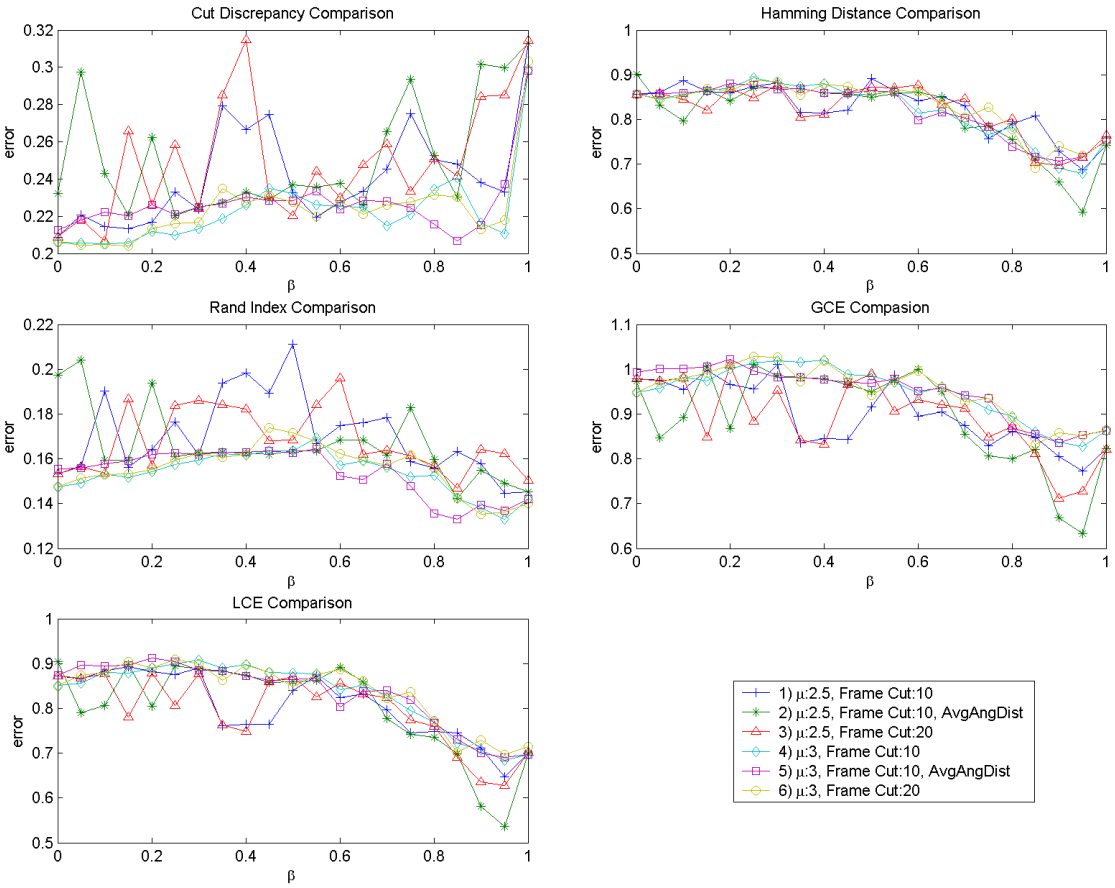

Figure 3. 33 Segmentation errors obtained for the *Dancing Man* sequence under different parameter settings.

	CD	HD	RI	GCE	LCE
Set 1	0.24126	0.82489	0.17016	0.89883	0.8055
Set 2	0.25157	0.80998	0.16591	0.89453	0.80202
Set 3	0.24994	0.81962	0.16799	0.89152	0.79651
Set 4	0.22294	0.82036	0.15358	0.94815	0.83252
Set 5	0.22712	0.8193	0.15366	0.94973	0.83317
Set 6	0.22403	0.82823	0.15565	0.95032	0.83759

Table 3. 11 Average error values calculated over β for Dancing Man sequence.

In Figure 3.34, 3.35, 3.36 and 3.37, we display *Cut Discrepancy*, *Hamming* Distance, *Rand Index* and *Consistency Error* plots for the parameter set 2, respectively. In terms of *Cut Discrepancy* and *Rand Index*, K-means results in better segmentations on the average while for *Hamming Distance* error values are close to each other.

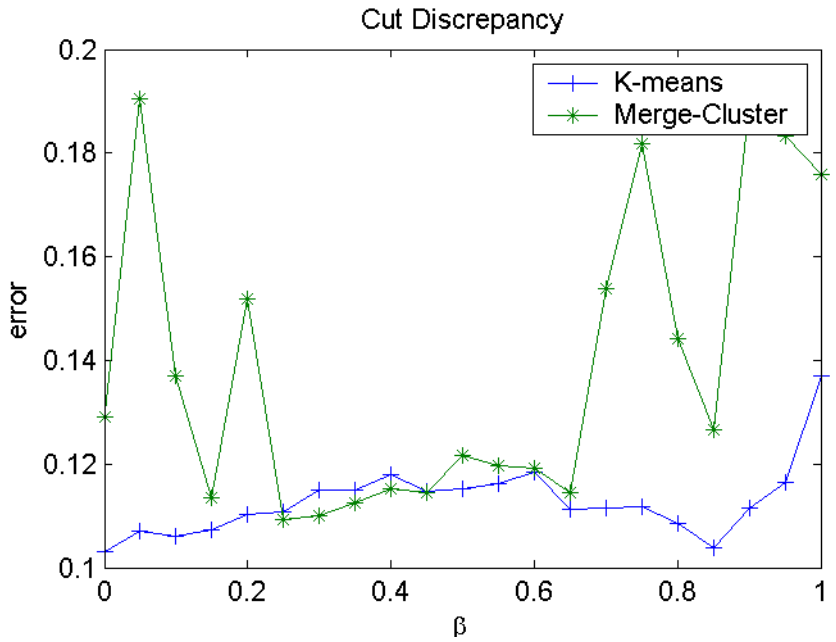

Figure 3. 34 *Dancing Man* Cut Discrepancy error plot for the parameter set 2.

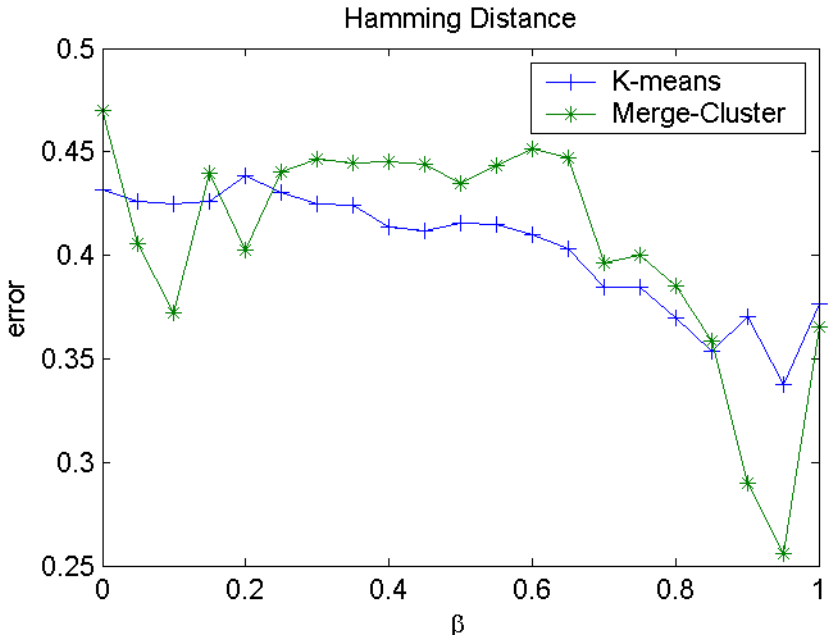

Figure 3. 35 *Dancing Man* Hamming Distance error plot for the parameter set 2.

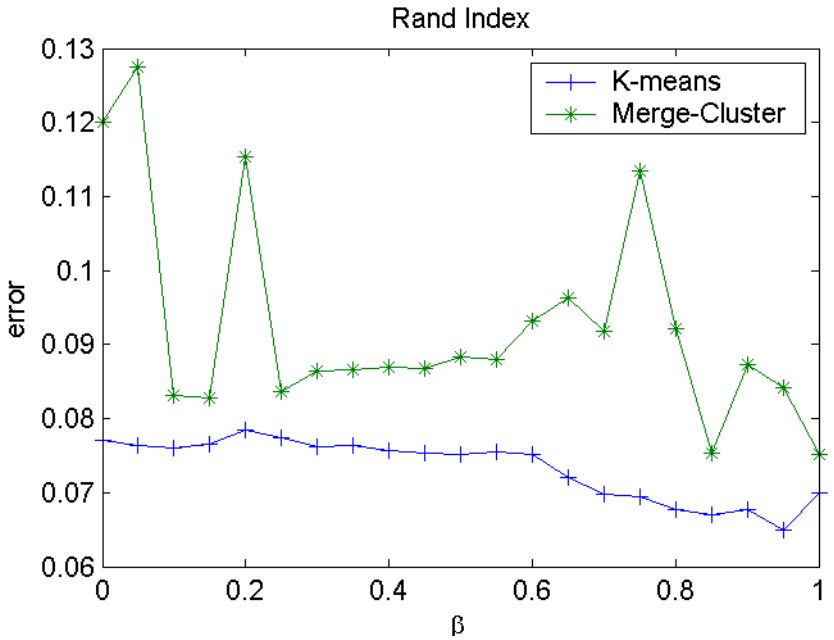

Figure 3. 36 *Dancing Man* Rand Index error plot for the parameter set 2.

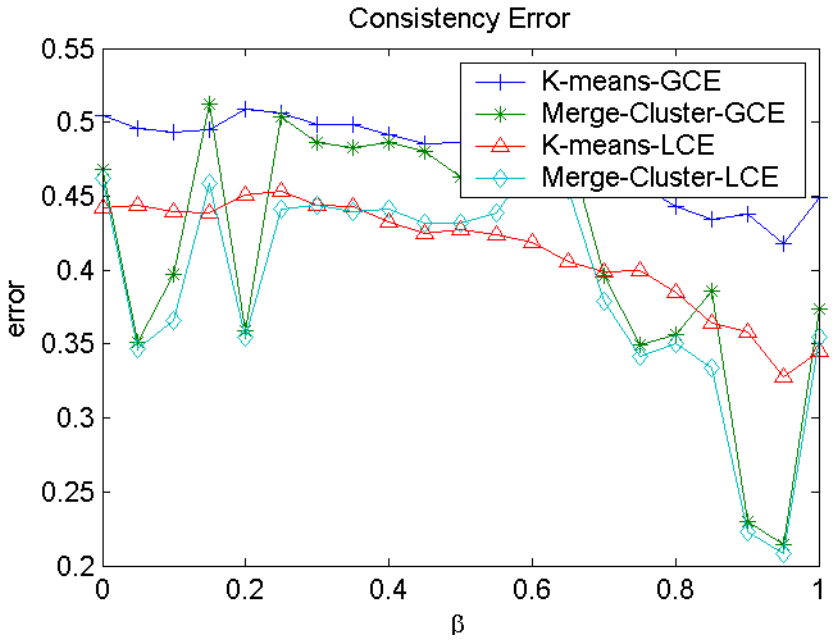

Figure 3. 37 *Dancing Man* Consistency Error plot for the parameter set 2.

3.2.4.1 Visual Segmentation Results

The segmentation maps obtained at two extreme cases with β=0, which means only spatial information is used, and β=1, which means only temporal information is used, are displayed in Figure 3.39 and Figure 3.40. Looking at the error plots, we see that the best results are obtained by using β=0.95, which are as shown in Figure 3.41. The corresponding error values for these β values are given in Table 3.12.

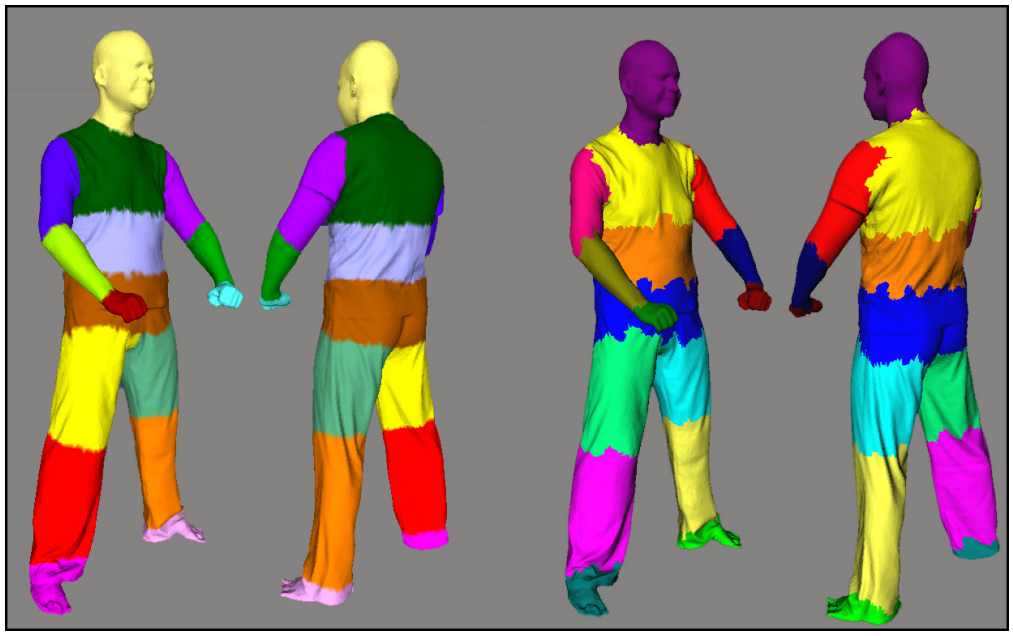

Figure 3. 38 *Dancing Man* ground truth segmentations. Left: The initial ground truth segmentation. Right: The ground truth segmentation used for error calculation.

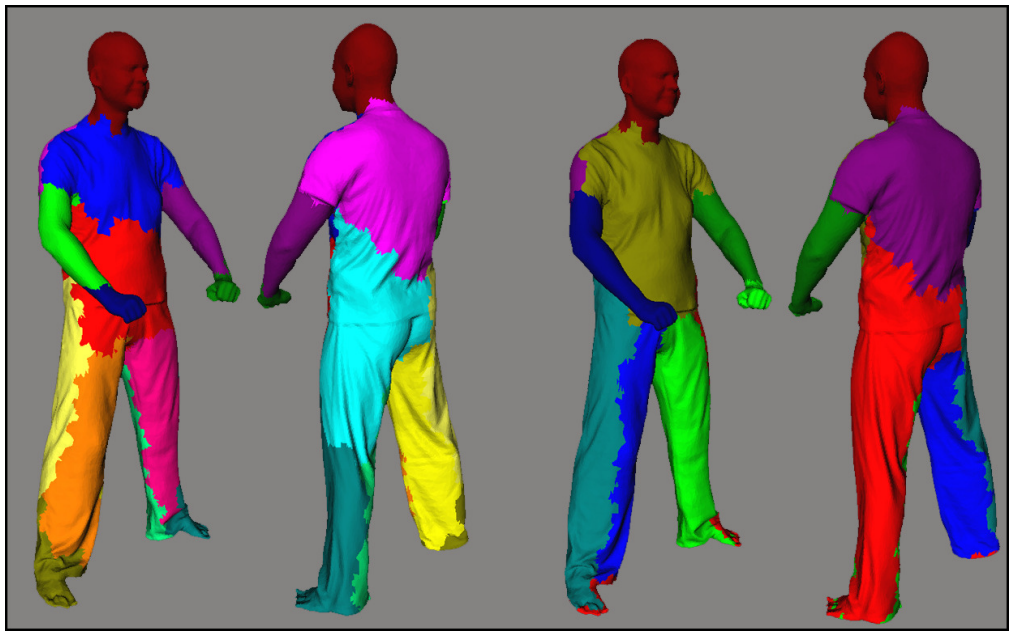

Figure 3. 39 *Dancing Man* segmentation results for β=0. Left: K-means. Right: Merge-cluster.

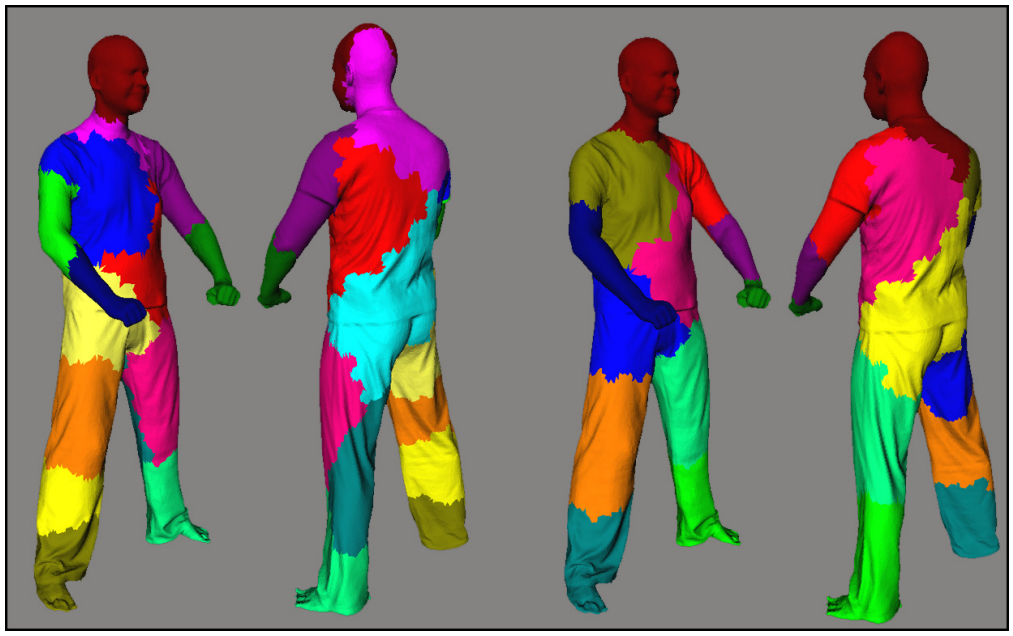

Figure 3. 40 *Dancing Man* segmentation results for β=1. Left: K-means. Right: Merge-cluster.

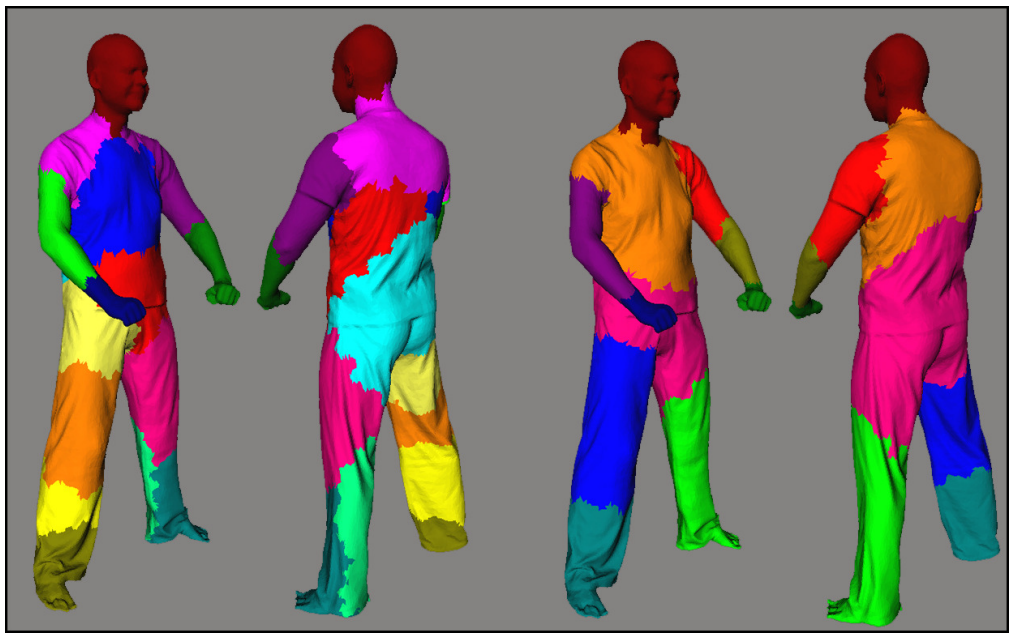

Figure 3. 41 *Dancing Man* segmentation results for β=0.95. Left: K-means. Right: Merge-cluster.

	β	CD	HD	RI	GCE	LCE
K-means	0	0.1031	0.43177	0.07711	0.50472	0.44214
	1	0.13701	0.37658	0.070047	0.44931	0.3449
	0.95	0.11651	0.33727	0.064895	0.41803	0.32804
Merge-cluster	0	0.12898	0.47003	0.12012	0.46844	0.46246
	1	0.17595	0.3652	0.075123	0.37364	0.35452
	0.95	0.18332	0.25558	0.084202	0.2144	0.20782

Table 3. 12 Metric error results for β values, which the visual results are shown for.

3.3 Discussion

In this section we will discuss and analyze the results presented in the previous section. In Table 3.13, a summary of the metric results for the used parameter sets is given.

At the beginning of our work, we were expecting to improve the segmentation results by introducing a *motion distance* term, instead of using spatial information single handedly. Looking at the error plots, for all parameter sets used and for almost all error metrics and for each sequence, we managed to find a β value where the error is smaller than the error received by using only spatial information. We also anticipated the spatial information and the temporal information to compensate each other so that the best result will not be received with either β=0 or β=1 but with a β value between. *Jumping Man* is a fine example of this situation. In this sequence the relative motion of right and left thighs is very small that the merge-cluster algorithm fails to keep both limbs separated when temporal information is used alone. However, with β=0.5, in other words, when both temporal and spatial information are used, merge-cluster manages to keep both limbs apart. In this particular case, since initial centers are distributed well enough, K-means divides right and left thigh into different clusters for both β=0.4 and β=1. However, it fails to separate the hip from the left leg when β =1 as a result of the faint relative motion between, although

the separation of the two is accomplished by K-means with β=0.4. Furthermore, segmentation using only spatial distance information in the form of angular distance is prone to fail when the cut regions and the regions with concavities do not match. For example, our segmentation results for *Jumping Man* are semantically good even with β=0 due to the fact that the surface of the mesh is smooth on average and concave regions do match with the cut regions. Nonetheless, the surface of *Dancing Woman* mesh is messy and has many concavities leading to semantically poor segmentation results far from the ground truth segmentation when only spatial information is used.

We have chosen each sequence for a specific reason. *Jumping Man* is an easy model to segment using only spatial information due to its smooth surface and cut regions placed at sharp edges generally. On the contrary *Dancing Woman* and *Dancing Man* sequences are difficult to segment using only spatial information due to their jaggy and wavy surfaces. *Horse Gallop* has an average difficulty in terms of surface structure. We showed that in all four cases our algorithm generates improved results with the incorporation of temporal information.

Sequence	μ	Average angular distantce used	Frame-cut	Clustering Algorithm	β	CD	HD	RI	GCE	LCE	Number of segments
Horse Gallop	3	Yes	10	K-means	0	0.13103	0.43201	0.087819	0.37788	0.26339	17
					0.95	0.1614	0.30191	0.04541	0.27013	0.12832	17
					1	0.1237	0.29174	0.046893	0.28449	0.15592	17
				Merge-cluster	0	0.13621	0.43509	0.085252	0.33151	0.27096	24
					0.95	0.11131	0.22894	0.050094	0.24223	0.16122	16
					1	0.12118	0.20485	0.051298	0.21912	0.13994	16
Jumping Man	2.5	Yes	10	K-means	0	0.1179	0.24823	0.070067	0.33402	0.2723	13
					0.4	0.089517	0.16173	0.046754	0.23419	0.1878	13
					1	0.061222	0.1507	0.048577	0.23058	0.18714	13
				Merge-cluster	0	0.18534	0.31956	0.12602	0.20467	0.19006	9
					0.5	0.1439	0.23343	0.071478	0.16162	0.1578	9
					1	0.13179	0.19826	0.083807	0.19986	0.18209	8
Dancing Woman	2.5	Yes	10	K-means	0	0.13272	0.39543	0.084432	0.44584	0.35639	15
					0.8	0.10948	0.23013	0.058634	0.2657	0.16628	15
					1	0.1515	0.27068	0.064216	0.31723	0.24025	15
				Merge-cluster	0	0.12727	0.41155	0.082503	0.41529	0.35425	18
					0.8	0.12313	0.1459	0.049475	0.15925	0.11045	13
					1	0.14597	0.17689	0.054315	0.20188	0.17713	12
Dancing Man	2.5	Yes	10	K-means	0	0.1031	0.43177	0.07711	0.50472	0.44214	16
					0.95	0.11651	0.33727	0.064895	0.41803	0.32804	16
					1	0.13701	0.37658	0.070047	0.44931	0.3449	16
				Merge-cluster	0	0.12898	0.47003	0.12012	0.46844	0.46246	8
					0.95	0.18332	0.25558	0.084202	0.2144	0.20782	12
					1	0.17595	0.3652	0.075123	0.37364	0.35452	10

Table 3. 13 Summary of metric errors

Recall that the number of segments *n* must be fed into the K-means clustering as well as *n* initial centers. K-means' sensitivity to the choice of initial centers affects its reliability when these centers are not distributed well enough. In order to overcome this problem we also introduced a way to choose initial centers so that they are distributed evenly on the surface of the mesh and therefore they are expected to be placed at desired locations which will yield the best clustering. Nevertheless, when sizes of limbs differ notably, this method fails to locate an initial center at each limb. *Horse Gallop* and *Jumping Man* are the two opposite examples demonstrating this fact. The huge size difference between the hooves and the torso of the horse model, K-means clustering fails to separate hooves as it divides the torso to many pieces. However, the ground truth segmentation of *Jumping Man* sequence has almost equal sized segments in terms of surface area. Thus, K-means clustering performs better than merge-cluster algorithm for *Jumping Man*.

The main advantage of merge-cluster algorithm against K-means clustering is that it removes the necessity of knowing the number of segments beforehand. Discarding the necessity of manual tuning of one hyper-parameter is an important step towards a more automated system. However, looking at the error plots, we conclude that merge-cluster algorithm is unstable and although it results in very promising segmentations for specific β values, it is difficult to give an interval which will guarantee its success all the time. The reason behind the instability of merge-cluster algorithm in terms of metric error values is the unpredictable number of clusters it produces. On the contrary, although K-means clustering yields higher error values for many β, it is more stable.

When we use K-means, for each of the four sequences, we managed to find an interval, $0.5 \leq \beta \leq 0.95$, in which, we guarantee that almost all of the error values for each error metric is smaller than the error values of the segmentation results obtained with β=0, with the exceptions of *Cut Discrepancy* values for *Dancing Man* and *Consistency Error* values of *Jumping Man*. In other words, in terms of the sequences we have experienced on, we

have found an interval for β so that incorporating the motion information enhances the segmentation results against the case only spatial information is used. In Table 3.14, maximum and average decline ratios for error values calculated over $0.5 \le \beta \le 0.95$ when K-means clustering is used are given. Note that the negative values belong to the mentioned exceptions. Looking at Table 3.14, we see that, for this interval of β, incorporation of temporal information enhances the results by up to %27 on average and up to %52 at maximum in terms of metric error results.

	Horse Gallop		**Jumping Man**		**Dancing Woman**		**Dancing Man**	
	avg	max	avg	max	avg	max	avg	max
CD	%10.76	%42.72	%3.11	%12.37	%7.72	%16.77	%-8.69	%-0.29
HD	%15.33	%30.11	%-0.71	%9.12	%20.69	%38.29	%9.77	%20.82
RI	%12.51	%48.29	%17.73	%23.53	%14.33	%28.34	%5.24	%12.68
GCE	%22.28	%51.28	%-21.89	%-11.80	%27.60	%51.98	%7.97	%22.80
LCE	%14.96	%28.51	%-15.87	%-7.15	%22.29	%37.69	%5.41	%13.48

Table 3. 14 Average and maximum declines in K-means error values within the interval $0.5 \le \beta \le 0.95$

The improvements obtained are also partly due to the use of average angular distances instead of using the angular distances obtained using a single frame. This is because, most of the time, a single frame does not carry enough information as we mentioned in the introduction chapter and showed in Figure 1.1.

The most important drawback of our algorithm is that it is not fully automated and adaptive. We have failed to find a value for β where the algorithm's performance is at its peak for all cases, although we can give an interval where the results are always improved. In order to find the best resulting β value, we still need to perform many segmentations for $0 \le \beta \le 1$. Setting of μ is another problem. The best results are obtained when $2.5 \le \mu \le 3.5$. However, as in the example of *Jumping Man* with μ=3, even when μ is this interval, unfortunately there is no guarantee that the outcome will be acceptable. In the

beginning, the use of static angular distances or average angular distances was a hyper-parameter setting decision but fortunately after the experiments we have learnt that using average angular distances yields better results. *Frame-cut*, which is the number of frames with smallest and largest Euclidian distances used in variance calculation in order to boost the effect of motion information, was our last manually set parameter. Although, in our initial experiments we have observed that using 10 smallest and largest Euclidian distance values is always sufficient to boost variances, we demonstrated the effect of increasing this number as well. In Table 3.15, we provide the average error values for *frame-cut*=10 and *frame-cut*=20, where the values for other hyper-parameters are set to the values with the best segmentation results. These results indicate that average errors generally increased as we increased the *frame-cut*.

	Horse Gallop		**Jumping Man**		**Dancing Woman**		**Dancing Man**	
Frame-cut	10	20	10	20	10	20	10	20
CD	0.24758	0.24602	0.30286	0.30066	0.25039	0.25972	0.25157	0.24994
HD	0.74029	0.75033	0.48676	0.49074	0.67611	0.70278	0.80998	0.81962
RI	0.14015	0.14054	0.19142	0.19235	0.159	0.16642	0.16591	0.16799
GCE	0.66612	0.67496	0.42753	0.43387	0.76874	0.79115	0.89453	0.89152
LCE	0.66612	0.67496	0.42753	0.43387	0.76874	0.79115	0.80202	0.79651

Table 3. 15 Average total error values calculated over β for different frame-cut values.

Chapter 4

CONCLUSIONS AND FUTURE WORK

In this work we have presented a segmentation method which is applicable to mesh sequences. We have shown that we can enhance the segmentation results by analyzing multiple postures in a time-varying mesh sequence. The proposed algorithm exploits the fact that semantically different parts of an articulated shape exhibit different rigid motions. In order to utilize this fact, a motion distance term, encoding the variances of Euclidian distances between vertices on the surface of a mesh, has been proposed.

We have tested our algorithm on four mesh sequences under different settings and have evaluated the segmentation performance both visually and quantitatively using different error metrics. By combining the motion term with spatial information, we have managed to benefit from strong sides of both spatial and temporal information, and to receive better results than the case using only spatial information.

The main drawback of our algorithm is that it lacks the ability to set all the parameters automatically. The choice of μ value for merge-clustering and setting the number of segments for K-means are important problems preventing our algorithm from being human independent. This limitation of our algorithm can be overcome by improving the clustering scheme by utilizing a method to choose the number of segments automatically.

BIBLIOGRAPHY

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

[2] M. Brand and K. Huang, "A unifying theorem for spectral embedding and clustering." *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

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

[4] A. Golovinskiy and T. Funkhouser, "Randomized cuts for 3d mesh analysis," *ACM Trans. Graph.*, vol. 27, no. 5, pages 1–12, 2008.

[5] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, "Topology matching for fully automatic similarity estimation of 3d shapes*," Proceedings of the 28th annual conference on computer graphics and interactive techniques*, pages 203–212, 2001.

[6] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," *Proceedings of the 27th annual conference on computer graphics and interactive techniques*, pages 279–286, 2000.

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

[8] S. Katz and A. Tal. "Hierarchical mesh decomposition using fuzzy clustering and cuts." *Proceedings of the SIGGRAPH*, pages 954–961, 2003.

[9] Y. K. Lai, S. M. Hu, R. R. Martin, and P. L. Rosin. "Fast mesh segmentation using random walks," *Proceedings of the* S*ymposium on Solid and Physical Modeling*, pages 183–191, 2008.

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

[11] R. Liu and H. Zhang, "Segmentation of 3d meshes through spectral clustering," *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 298–305, 2004.

[12] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing,* vol. 17, no. 4, pages 395–416, 2007.

[13] P. Sand, L. McMillan, and J. Popović, "Continuous capture of skin deformation," *ACM Trans. Graph.,* vol. 22, no. 3, pages 578–586, 2003.

[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, pages 249–259, 2008.

[15] S. Shlafman, A. Tal, and S. Katz, "Metamorphosis of polyhedral surfaces using decomposition," *In Computer Graphics Forum*, pages 219–228, 2002.

[16] L. Grady. "Random walks for image segmentation," *IEEE Trans. Pattern Analysis and Machine Inbtelligence,* vol. 28, no. 11, pages 1768–1783, 2006.

[17] M. Garland, A. Willmott, and P. S. Heckbert, "Hierarchical face clustering on polygonal surfaces," *Proceedings of the Symposium on interactive 3D Graphics* I3D, pages 49-58, 2001.

[18] R. Arcila, K. S. Buddha, F. Hétroy, F. Denis and F. Dupont, "A Framework for motion-based mesh sequence segmentation," *International Conference on Computer Graphics, Visualization and Computer Vision, WSCG*, 2010.

[19] T. Lee, Y. Wang, and T. Chen, "Segmenting a deforming mesh into near-rigid components," *The Visual Computer,* vol. 22, no. 9, pages 729-739, 2006.

[20] H. Yamauchi, S. Gumhold, R. Zayer and H.P. Seidel, "Mesh segmentation driven by Gaussian curvature," *The Visual Computer,* vol. 21, no. 8-10, pages 659-668.

[21] L. Chen and N. D. Georganas, "An efficient and robust algorithm for 3D mesh segmentation," *Multimedia Tools and Applications,* vol. 29, no. 2, pages 109-125.

[22] X. Ju , N. Werghi , J. P. Siebert, "Automatic Segmentation of 3D Human Body Scans," *Proc. IASTED Int. Conf. on Computer Graphics and Imaging (CGIM),* 2000.

[23] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, E. Boyer, "Articulated shape matching using Laplacian eigenfunctions and unsupervised point registration," *Computer Vision and Pattern Recognition, CVPR,* 2008.

[24] J. Varun, H. Zhang, "Robust 3D Shape Correspondence in the Spectral Domain," *Shape Modeling and Applications, SMI, IEEE International Conference,* 2006.

[25] A. Y. Ng, M. I. Jordan, Y. Weiss, "On Spectral Clustering: Analysis and An Algorithm." *In Advances in Neural Information Processing Systems,* vol. 14, pages 857–864, 2002.

[26] Y. Weiss, "Segmentation Using Eigenvectors: A Unifying View," *Proceedings IEEE International Conference on Computer Vision,* pages 975–982, 1999.

[27] J. C. Chen and Y. Nakamura, "The underlying principle of Dijkstra's shortest path algorithm," *Formalized Mathematics*, vol. 11, no. 2, pages 143–152, 2003.

[28] E. Kalafatlar, Y. Yemez, "3D articulated shape segmentation using motion information," *20th International Conference on Pattern Recognition, ICPR,* 2010.

[29] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H. P. Seidel and S. Thrun, "Performance capture from sparse multi-view video," *SIGGRAPH*, pages 1-10, 2008.

[30] Y. Sahillioğlu, Y. Yemez, "3D shape correspondence by isometry-driven greedy optimization," *Computer Vision and Pattern Recognition (CVPR),* pages 453-458, 2010.

[31] N. B. Venkateswarlu and P. S. Raju, "Fast ISODATA clustering algorithms," *Pattern Recogn.,* vol. 25, no. 3, pages 335-342, 1992.

VITA

Emre Kalafatlar was born in İstanbul, Turkey on August 23, 1983. He graduated from Kabataş Erkek Lisesi, İstanbul in 2001. He received his B.S. degree in Electrical and Electronics Engineering with a double-major in Computer Engineering from Koç University, İstanbul in 2006. After a year of work experience, in 2007, he joined the M.S. Program in Electrical and Computer Engineering at Koç University, as a research and teaching assistant. Having received the M.S. degree in 2010, he is now working in Nortel Netaş as a design engineer.