Abstract — Active contour methods can be used to segment a 3D mesh into parts by iteratively moving the contour to the mesh region that minimizes the contour energy. However, as the contour moves, it often does not lie on the mesh surface. To address this problem, existing methods use either vertex/edge projection or mesh parameterization to obtain the corresponding contour on the mesh surface. Although vertex/edge projection methods are simple, they may create unwanted loops along the projected contour due to irregular mesh connectivity or modeling noise. Extra operations, which are often complex, are needed to remove such loops. On the other hand, mesh parameterization suffers from distortion and out-of-range problems, which are not trivial to solve. In this paper, we propose a face projection method to address the above problems. Our experiments show that the proposed method produces much smoother, more consistent and accurate projected contours than existing methods. At the end of the paper, we also show some multimedia applications of our method.

Index Terms — 3D active contours, contour projection, mesh parameterization, mesh segmentation.

I. INTRODUCTION

Digital entertainment exploits multimedia, input & output and communication technologies to provide people various forms of amusement, where computer games [Li09] and movies are typical examples. Technically, the production of any form of digital entertainment generally involves the creation, processing and presentation of content of different types. Among these types, 3D content plays an increasingly significant role during last decade, as it widens the product range of digital entertainment by allowing the creation of artificial objects that are hard to be practically found available. Specifically, artificial objects are typically modeled by 3D geometry primitives, namely vertices and edges, and a popular option for adding realistic object poses and animations to artificial objects is by motion capture [Mena99]. However, creating and manipulating 3D content is generally complicated and time consuming. In light of this, mesh segmentation [Sham08] and mesh retrieval [Tam07] techniques have been developed to split an object into meaningful parts and to identify objects with desired features, respectively, which forms underlying technologies to support the reuse of 3D content and hence help reduce the burden in 3D content creation and editing. Unfortunately, such techniques generally suffer from the accuracy problem, particularly when they need to process 3D content with noise or various resolutions. This is due to the fact that the ability of these techniques in identifying features or critical boundaries under such unfavorable situations is reduced considerably. To improve the situation, we have developed an active contour projection method to tackle such a problem.

Recently, the active contour model (ACM), which was originally developed from image segmentation [Kass88], has been applied to 3D mesh segmentation [Lee02], as it is found that the concept of energy terms in ACM can carry over to support boundary feature detection of 3D meshes. In an ACM segmentation process, an initial boundary is firstly constructed and then the contour is allowed to evolve iteratively over the mesh until it reaches a segmentation boundary, which minimizes the contour energy. However, unlike 2D images, 3D meshes have an additional dimension, i.e., height, and most of them have irregular connectivity. As such, a contour may not lie on the mesh surface during its evolution, if only the energy terms are considered in the contour evolution process. Hence, the accuracy of the segmentation can be significantly affected. To address this problem, [Jung04, Bisc05, Ji06] propose to project snaxels, which are the vertices that constitute an active contour, such that they are constrained to lie on either mesh vertices or edges. However, these methods may cause a snaxel to be projected on a mesh location that does not really minimize the contour energy, or even generate jagged segmentation boundaries, especially when the meshes have low resolution. Alternatively, [Lee02, Lee05] allow snaxels to be projected on any arbitrary mesh face points by performing parameterization on local mesh regions, which basically project each local mesh region onto a 2D space for the contour to evolve. Unfortunately, the distortion incurred by the parameterization process may lead to inaccurate results, especially when a 3D mesh comprises highly curved regions. In addition, if the desired minimum contour energy is outside the local parameterized mesh region, extra parameterization operations must be carried out to handle such situation, which is time-consuming.

Our research interest is on mesh segmentation, which requires generating accurate segmentation boundaries in order to obtain the desired segmentation result. To achieve this goal, we propose a new projection method...
for active contour based mesh segmentation. The main
correspondence of this paper is that we propose a contour
projection method based on two operations, motion
displacement projection and the projection face selection.
The new method avoids generating jagged segmentation
boundaries, even for meshes with low resolution, and
solves the loopin problem. Hence, the resulting
segmentation boundaries would be smoother. It also
prevents the use of the parameterization process, which
suffers from distortion and the out-of-range problem. In
addition, we depict some applications of our new method.

The rest of this paper is organized as follows. Section
II. Related Work

3D content is an important media to digital
entertainment applications. However, creating such
content is generally labor-intensive. A handy solution to
address this problem is to reuse existing objects or their
meaningful segments, where proper feature detection and
object segmentation algorithm should be in place to
support object retrieval or object segment extraction.
Consider these operations may perform poorly when
handling objects with noise or insufficient resolution, we
propose to apply active contour model (ACM) providing
a better alternative to implement such operations. This is
to exploit the capability of ACM in using energy items
for identifying object features and retaining the integrity
of boundary features obtained as a whole. In addition, we
also review the limitations of existing work in ACM.

A. 3D Content

During last decade, 3D content becomes a very
important media to digital entertainment applications, as
it allows people creating artificial objects that are hard to
be found available in reality, or capturing and modifying
real-life objects into different forms. The construction of
3D content broadly involves object modeling and object
animation. Object modeling concerns how an object is
acquired or created and how the object is represented
digitally in an application, while object animation
concerns how object shape can be changed over time and
following certain behaviors, such as reactive motion
[Komu05].

To create a 3D object, it is typical to use some
modeling tools, such as Maya or 3D Studio Max, to
construct the object using a polygon or a parametric
[LI07, LI09] representation. However, creating objects in
this way generally requires considerable experience or
technical skill from a user. Alternatively, with the
availability of 3D scanner technology, a 3D object can be
acquired from a real-life object by collecting its sample
points followed by performing an object reconstruction
process [Ber02]. Unfortunately, objects obtained in this
way are prone to noise. To change the shape of an object,
we may modify the internal data representation of the
object, which may be complicated. Instead, an object can
be deformed through virtual sculpting [Li03], skeletal
animation [Zhen10] or applying animation data from
motion capture [Men99], which is more intuitive to a
user. Despite the availability of tools and techniques,
creating and manipulating 3D content is still generally
complicated and time consuming. To alleviate such a
burden, a favorable solution is to reuse existing objects.
This relates to how relevant objects or their meaningful
segments can be retrieved or extracted, which may
demand substantial support from feature detection and
object segmentation algorithms.

On the other hand, multi-resolution modeling [To99,
To01] serves as a popular technique to allow real-time
rendering of complicated objects or many objects
together at the same time, by selectively adjusting the
resolution of each object on the fly according to certain
run-time criteria such as hardware capability or object
importance. This is critical to many digital entertainment
applications, particularly those supporting user
interactions, such as online gaming [Li10]. This
technique can also be applied to objects acquired from 3D
scanning to reduce their resolutions, as their resolutions
are generally too high resolutions to support real-time
rendering. Hence, multi-resolution modeling is widely
adopted by digital entertainment applications. Despite
reducing object resolution can lead to a better rendering
performance, it affects the accuracy of feature detection
and object segmentation algorithms considerably.

B. Active Contour Model

Active contour model (ACM) is a decent technique for
identifying boundary feature. When applying ACM to 3D
objects, mesh features of an object is formulated by
energy terms and an iteration process is carried out to
identify segmentation boundaries by evolving the contour
towards mesh regions that generate minimum contour
ergories. Mathematically, an ACM is modeled by an
ergory that comprises an internal energy \( E_{\text{int}}(s) \) and an external energy \( E_{\text{ext}}(s) \) as follows:

\[
E = \int \left( E_{\text{int}}(s) + E_{\text{ext}}(s) \right) ds
\]

(1)

The internal energy keeps the shape of a contour
regular and smooth during its evolution. It also prevents
the contour from leaking out of gaps in the boundary
of a mesh feature, e.g., due to modeling noise, or sticking
to spurious mesh features. On the other hand, the external
ergory behaves like an attractor, which attracts a contour
to mesh features. Such energy measures how strong each
mesh vertex can attract the contour.

The ACM was originally proposed for image
segmentation [Kass88] and has recently been applied to
3D mesh segmentation [Lee02]. However, unlike 2D
images, 3D meshes have an additional dimension, i.e.,
height, and most of them have irregular connectivity. If
only the energy terms are considered in the contour
evolution process, a contour may move away from the
mesh surface. Hence, the external contour energy, which
formulates the mesh features, can no longer be evaluated.
This leads to a failure in performing mesh segmentation.
Common options to address such problem are mesh parameterization and contour projection.

Mesh parameterization methods for 3D geometry models [Zhan05] project a mesh onto a 2D parametric space. [Lee02, Lee05] adopt such an idea to parameterize local regions of a mesh onto 2D spaces and let a contour evolve over the 2D space based on energy minimization, while the contour energy is still defined on the 3D mesh space. The final contour is obtained by projecting the resultant contour in the 2D space back to the 3D mesh space. This essentially keeps the contour lying on the mesh surface. In addition, this method performs parameterization only on local mesh regions to avoid potentially large distortion to occur. However, it may still generate unstable results, as the results depend greatly on the parameterization technique used. The distortion problem may also be unavoidable if a 3D mesh contains highly curved regions. Further, it is possible that the location for the desired minimum contour energy lies outside of the local parameterized mesh region. In such case, extra parameterization operations must be performed, which are time-consuming.

Contour projection methods project a contour onto the mesh surface every time when the contour is updated during the contour evolution process. To ensure that the contour will be lying on the mesh surface, [Bisc05, Ji06, Jung04] restrict the projection of the contour only onto mesh vertices or edges. In particular, [Jung04] evolves a contour by moving its snaxels to the neighbor mesh vertices that produce the lowest contour energy. Alternatively, [Bisc05] constrains snaxels to map on supporting edges only. It uses oblique projection to project the normal vector of the contour to the supporting edge of a snaxel in the 3D mesh space. It uses the geodesic normal to form its motion displacement, and projects it to the tangent plane along its supporting edge. However, as the tangent plane may not lie on the mesh surface, this method needs to flatten the neighbor faces of the snaxel in order to project the motion displacement on the flattened 2D plane along its supporting edge. As this edge projection method constrains snaxels to be projected on mesh edges, it limits the accuracy of the contour evolution process. In addition, the flattening process may also lead to distortion as it changes the relative lengths of the mesh edges. In [Ji06], snaxels are constrained to lie on either vertices or edges only. In order to refine a contour, a snaxel moves along its supporting edge based on the parameterization free active contour model as in [Bisc05]. However, [Ji06] applies a splitting scheme to smooth the contour. With such a scheme, when a snaxel runs into a vertex, it splits into several snaxels at the outgoing edges.

III. OUR FACE PROJECTION METHOD

Similar to the parameterization methods, our method supports accurate contour evolution by allowing the contour snaxels to be projected onto any arbitrary mesh surface locations, without restricting to mesh vertices or edges. However, unlike the parameterization methods, our method evolves a contour directly over the mesh surface along its natural direction based on evaluating the contour energy function. This avoids producing unstable results and the out-of-range problem that the parameterization methods suffer from. Our method involves the following major steps:

Step 1: Given an active contour with n snaxels, and an arbitrary snaxel of the contour, si, we compute si’s motion displacement D as follows (Ref. Figure 1(a)):

$$D(s_i) = D_{int}(s_i) + D_{ext}(s_i)$$  \hspace{1cm} (2)

$$= \sum_{j \in N_i} \left( \vec{F} - \frac{1}{|N_i|} \sum_{j \in N_i} \langle s_j - si \rangle \right) + Grad(s_i)$$

where $\vec{F} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|N_i|} \sum_{j \in N_i} \langle s_j - si \rangle$, which is the global mean square distance between $s_i$ and the midpoint of its two neighbor snaxels. $N_i$ denotes the neighbor snaxels of $s_i$. The motion displacement comprises an internal displacement $D_{int}$ produced by the internal force and an external displacement $D_{ext}$ produced by the external force. The first two items on the RHS of Eq. (2) constitute the internal displacement, while the external displacement is defined by the gradient $Grad(s_i)$ at $s_i$.

![Figure 1. The projection of motion displacement.](image)

Figure 1. The projection of motion displacement.

Step 2: We identify the projection face of snaxel $s_i$ (Ref. Figure 1(b)).

Step 3: We project motion displacement $D$ to the projection face $F$. As shown in Figure 1(c), $N$ is the normal direction of face $F$, $\alpha$ is the angle between motion displacement $D$ and projected motion displacement $D_1$, and $\beta$ is the angle between $N$ and $D$. We consider the tangent component of $D$ on the projection face $F$ as the real motion displacement of $s_i$. Then, the projected motion displacement $D_1$ can be computed as follows:

$$D_1 = D \cos \alpha = D - N \|D\| \sin \alpha$$ \hspace{1cm} (3)
where \( \cos \beta = \frac{D^T N}{\| D \| \| N \|} \).

**Step 4:** If the motion displacement of \( s_i \) is beyond the scale of the projection face, we take the intersection point between the projected motion displacement and the edge of projection face as a new snaxel and the section of the displacement exceeding the current projection face as the new motion displacement. We then repeat step 2 to step 4, until the motion displacement of \( s_i \) no longer exceeds the current projection face (Ref. Figure 1(d)). Note that point \( S \) in Figures 1-5 represents the location of snaxel \( s_i \).

In the following subsections, we discuss the implementation of our face projection method in detail.

### A. Computation of Motion Displacement

In practice, the motion displacement \( D \) of a snaxel \( s \) is computed in a different way depending on the type of mesh primitive as follows:

- **Vertex:** When snaxel \( s \) is projected onto a mesh vertex \( v \), its motion displacement \( D \) is set equal to the displacement of the mesh vertex \( v \). As \( D(v) \) is computed according to Eq. (2), \( D(s) \) becomes:

  \[
  D(s) = D(v) \quad (4)
  \]

- **Edge:** When snaxel \( s \) is projected onto a mesh edge \( E \), its motion displacement \( D \) is computed by taking linear interpolation among the motion displacements of the two end vertices \( v_i \) and \( v_j \) of \( E \). Again, \( D_{ext}(v_i) \) and \( D_{ext}(v_j) \) are computed by Eq. (2). Hence, \( D(s) \) becomes:

  \[
  D(s) = D_{ext}(s) + \delta_D \cdot D_{ext}(v_i) + \delta_D \cdot D_{ext}(v_j) \quad (5)
  \]

  where \( \delta_D = \left| \frac{s - v_j}{v_i - v_j} \right| \), \( \delta_D = \left| \frac{s - v_i}{v_j - v_i} \right| \).

**Face:** When snaxel \( s \) is projected onto a mesh face \( F \), its motion displacement \( D \) is computed by taking linear interpolation among the motion displacements of the three corner vertices \( v_i \), \( v_j \) and \( v_k \) of mesh face \( F \). As \( D_{ext}(v_i) \), \( D_{ext}(v_j) \) and \( D_{ext}(v_k) \) are computed by Eq. (2), \( D(s) \) becomes:

  \[
  D(s) = D_{ext}(s) + \delta_D \cdot D_{ext}(v_i) + \delta_D \cdot D_{ext}(v_j) + \delta_D \cdot D_{ext}(v_k) \quad (6)
  \]

  where \( \delta_D = \left| \frac{s - v_j}{\text{sum}} \right| \), \( \delta_D = \left| \frac{s - v_i}{\text{sum}} \right| \), \( \delta_D = \left| \frac{s - v_k}{\text{sum}} \right| \), and \( \text{sum} = |s - v_i| + |s - v_j| + |s - v_k| \).

### B. Projection Face Selection

When snaxel \( s \) is projected on a vertex \( S \), its projection face \( F \) can be chosen from the set of adjacent mesh faces to \( S \), which has the largest angle \( \beta \) between the motion displacement \( D \) and the surface normal \( N \). As shown in Figure 2(a), the motion displacement \( D_1 \) is a projection of \( D \) on the face highlighted. When snaxel \( s \) lies on an edge, there will be only two candidate projection faces available (Ref. Figure 2(b)), the two adjacent faces which share the same edge. Again, the face which normal \( N \) has a larger angle \( \beta \) with \( D \) will be chosen. When snaxel \( s \) lies on a face, the face will be selected as the projection face (Ref. Figure 2(c)).

### C. Handling of Large Motion Displacements

When the motion displacement projects beyond the span of the projection face, we need to select an additional neighbor mesh face so that the part of the motion displacement that goes beyond the current projection face can be projected. Figure 3 describes such situation when the snaxel is projected onto a mesh vertex, a mesh edge and a mesh face. Regardless of the type of mesh primitive that the snaxel is projected onto, we project the motion displacement \( D \) onto AMB, which will
produce an intersection point $G$ at edge $AB$. We then select $G$ as a new snaxel, and use $GH$ as a new motion displacement for projecting to the neighbor face that shares edge $AB$ with $AMB$. The projection process carries on until the entire motion displacement is projected onto the mesh surface.

Sometimes, there may not be a projection face for a motion displacement. The treatment depends on the type of mesh primitive that a snaxel is projected onto.

**Vertex:** As shown in Figure 4(a), motion displacement $D$ does not project onto a mesh face. In such a case, we select the mesh edge that forms the minimum angle with $D$ among all neighbor mesh edges connecting to vertex $S$ for $D$ to project onto.

**Edge:** Motion displacement $D$ may not always find a suitable projection face when the snaxel lies on a mesh edge. As shown in Figure 4(b), along the mesh edge $BM$ that snaxel $S$ is projected onto, there are only two candidate faces, $ABM$ and $QBM$, for $D$ to project onto. However, a mesh face cannot be chosen as the projection face if $D$ is not projected onto its span. For example, if the projection $SC_2$ of $D$ is outside $ABM$’s span, $ABM$ cannot be the projection face. Likewise, $QBM$ cannot be the projection face if the projection $SC_1$ of $D$ is outside of $QBM$’s span. In addition, even if $D$ is projected within the span of a mesh face, this face cannot become the projection face if the angle between $D$ and this face is larger than 90°. Eventually, if a suitable projection face cannot be found, in order to keep the snaxel lying on the mesh surface, we simply project $D$ onto edge $BM$.

**Face:** This situation would not happen if a snaxel lies on a mesh face.

**D. Comparison between Projection Methods**

Existing projection methods do not have a proper treatment for the large motion displacement problem. When a motion displacement is projected beyond the span of a projection face, we may choose either to keep projecting the residual of the motion displacement to an adjacent mesh edge (Ref. Figure 5(b)), or to stop projecting this motion displacement (Ref. Figure 5(c)). The first option leads to a change in contour direction, while the second option modifies the contour position. Both results are undesirable.

To avoid producing undesirable results, as described in Section III.C, we project a large motion displacement in parts to an adjacent mesh face (Ref. Figure 5(a)). This is important for maintaining the accuracy of the contour evolution. It also simplifies the projection algorithm in two aspects. First, existing methods need to determine the start and the end points of a supporting edge, such that a snaxel can move along the direction formed by these two points. However, we let the snaxel move along its natural direction according to the contour energy evaluated; this leads to a more accurate result. Second, we do not need to apply a cleaning conquest [5] to remove invalid contour segments, which is required by existing methods when more than one consecutive contour segment is lying on the same mesh face (Ref. Figure 6). This is done by substituting invalid contour segments by a single segment. In our method, although there may be more than one contour segment on the same mesh face, they are only generated to improve the contour accuracy.

As discussed above, with the edge or vertex projection methods, a snaxel can only move to the minimal energy position along a mesh edge. This limitation leads to the creation of multiple loops along the contour due to irregular mesh connectivity or modeling noise. Figure 7 shows an example of this problem. In Figure 7(a), the curve segment $ABCD$ is the initial contour for evolution. If the vertex projection method is applied, evolved snaxels can only stop at vertices. After contour evolution, the contour becomes $ABCD'$ (Ref. Figure 7(b)). If the edge projection method is used, evolved snaxels can only move along mesh edges. The contour after evolution becomes $A'B'C'D'$ (Ref. Figure 7(c)). Both the edge projection and the vertex projection methods produce a
loop in the output contour. Our method, on the other hand, does not produce loops. The output contour becomes \(A"B"C"D"\) (Ref. Figure 7(d)).

We further illustrate the looping problem through segmentation of a mushroom model as shown in Figure 8. We segment the mushroom model by our method (Ref. Figure 8(a)), the vertex projection method (Ref. Figure 8(b)) and the edge projection method (Ref. Figure 8(c)). Figures 9(a-c) show the enlarged images of the segmentation boundary generated by these methods, respectively. As shown in Figures 9(b) and 9(c), the looping problem occurs when either the vertex or the edge projection method is applied. On the contrary, our method does not have this problem (Ref. Figure 9(a)).

IV. RESULTS

To evaluate the performance of our method, we have compared it with the vertex projection method, the edge projection method and the parameterization method. Given a human model as shown in Figure 10, we have

<table>
<thead>
<tr>
<th>Model location</th>
<th>Model size (vertices) / contour size (snauxels)</th>
<th>Average distance errors</th>
<th>Our method</th>
<th>Edge projection</th>
<th>Vertex projection</th>
<th>Parameterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elbow</td>
<td>2752/30</td>
<td>0.00090</td>
<td>0.00129</td>
<td>0.00575</td>
<td>0.01331</td>
<td>0.00915</td>
</tr>
<tr>
<td></td>
<td>5002/30</td>
<td>0.00049</td>
<td>0.00095</td>
<td>0.00385</td>
<td>0.00848</td>
<td>0.00848</td>
</tr>
<tr>
<td></td>
<td>7502/42</td>
<td>0.00073</td>
<td>0.00095</td>
<td>0.00205</td>
<td>0.00477</td>
<td>0.00447</td>
</tr>
<tr>
<td>Knee</td>
<td>2752/29</td>
<td>0.00183</td>
<td>0.00219</td>
<td>0.00715</td>
<td>0.01331</td>
<td>0.01331</td>
</tr>
<tr>
<td></td>
<td>5002/35</td>
<td>0.00111</td>
<td>0.00121</td>
<td>0.00525</td>
<td>0.01002</td>
<td>0.01002</td>
</tr>
<tr>
<td></td>
<td>7502/41</td>
<td>0.00075</td>
<td>0.00133</td>
<td>0.00363</td>
<td>0.00686</td>
<td>0.00686</td>
</tr>
<tr>
<td>Ankle</td>
<td>2752/26</td>
<td>0.00083</td>
<td>0.00116</td>
<td>0.00684</td>
<td>0.02174</td>
<td>0.02174</td>
</tr>
<tr>
<td></td>
<td>5002/30</td>
<td>0.00071</td>
<td>0.00112</td>
<td>0.00458</td>
<td>0.00779</td>
<td>0.00779</td>
</tr>
<tr>
<td></td>
<td>7502/53</td>
<td>0.00047</td>
<td>0.00095</td>
<td>0.00216</td>
<td>0.00590</td>
<td>0.00590</td>
</tr>
</tbody>
</table>
selected the initial contours, which are labeled by different colors, to locate the elbow, the knee, and the ankle. We compare the average distance error $E_{\text{distance}}$, as shown in Eq. (7), between the ideal contour and the contours produced by different methods, and the error of the same contour generated under different model resolutions. The comparison is shown in Table 1. Note that the smaller the distance error means that the actual contour is much closer to the ideal one.

$$E_{\text{distance}} = \frac{1}{N} \sum_{i=1}^{N} | s_{\text{actual}}(i) - s_{\text{ideal}}(i) |$$

In addition, as shown in Table 2, we have compared the time costs among the four methods when processing a horse model (Ref. Figure 11) at two different resolutions. Moreover, we also present the segmentation results on a horse model produced by these methods, as shown in Figure 11. Finally, we apply our method to map tattoo patterns on models at different resolutions to demonstrate the ability of our method in producing consistent results, as shown in Figure 12.

Table 2. Time costs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Small model (486 vertices)/ 5 contours</th>
<th>Large model (4850 vertices)/ 4 contours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex Projection</td>
<td>0.313s</td>
<td>2.047s</td>
</tr>
<tr>
<td>Edge Projection</td>
<td>0.219s</td>
<td>2.219s</td>
</tr>
<tr>
<td>Our method</td>
<td>0.781s</td>
<td>4.688s</td>
</tr>
<tr>
<td>Parameterization</td>
<td>1.9850s</td>
<td>8.5000s</td>
</tr>
</tbody>
</table>

A. Accuracy

This experiment compared the average distance error between the ideal and the actual contours, where the actual contours are projected onto the mesh surface using
different projection methods. Figure 11 shows the ideal contour and the actual contours produced by different projection methods. By comparing our method with the others, we can see that our method produced the closest contour to the ideal one. Also, the other methods produced contours with irregular shapes to some extent, where the parameterization method performed the worst.

To study the accuracy of our method, we have measured the average distance errors incurred by different projection methods using the human model at three different resolution levels, i.e., 2752 vertices, 5002 vertices, and 7502 vertices. The average edge lengths of these models are 0.0113, 0.0080, and 0.005, respectively. As illustrated in Table 1, the average distance errors produced by other methods are much larger than our method with respect to the same model location and model resolution. In addition, although the average distance error increases when the model resolution becomes lower, such an error increases with a much smaller scale when our method is applied by comparing with the other methods. Actually, if the average edge lengths of the horse model are taken into account, the errors produced by our method can be ignored. On the other hand, the parameterization method produces the worst results. In summary, results show that our method can produce the most accurate contour.

B. Performance

We compare the computation time between our method and the others with respect to a small model (486 vertices) and a large model (4850 vertices). The small model contains 5 contours (134 snaxels in total) while the large model contains 4 contours (227 snaxels in total). All experiments were run by a PC equipped with an Intel core 2 Duo CPU and 2GByte memory under Matlab 7.0. The result is shown in Table 2. Note that the time cost by edge projection does not include the time spent on the cleaning conquest, which has been discussed in section 3.4. Our method performs worse than the vertex and the edge projection methods, as we need to find the intersect point between the projection face and the motion displacement during the contour evolution. However, our method still runs much faster the parameterization method.

C. Consistency

Our face projection method can keep consistency independent of the resolution of the mesh model or the active contour. To further illustrate this advantage, we show the results of applying our method in a tattoo projection application. Such application is very useful. For example, horses need to be distinguished in a racing game by labeling different numbers; china is often wrapped with fancy pattern; an enterprise needs to print their logo on their products, etc. Figure 12 gives the projection results of different tattoos. It shows that our method works well with different tattoo shapes and resolutions.

Consistency is also proved when different resolutions of mesh models are used. The results are shown in figure 13-16. Figure 13 shows the result of projecting a low resolution tattoo ‘Mickey Mouse’ (22 snaxels) to the high resolution model (24955 vertices). Figure 14 shows the result of projecting a high resolution tattoo ‘Mickey Mouse’ (508 snaxels) to the low resolution model (2491 vertices). Figure 15 shows the result of projecting a low resolution tattoo ‘Mickey Mouse’ (22 snaxels) to the low resolution model (2491 vertices); and Figure 16 shows the result of projecting a high resolution tattoo ‘Mickey Mouse’ (508 snaxels) to the high resolution model (24955 vertices). We discuss the results as follows:

Edge projection & vertex projection: Because these projection methods constrain snaxels to edges or vertices, so they change the direction of motion displacement (Ref. Figure 13-16(b, c)). The detailed reasons were explained in section 3.4. As a result, the tattoo shapes are distorted. Another problem is that part of the contour inter-crosses itself (Ref. Figure 13-16(c) & Figure 14(b)) when the actual motion displacement is longer than the theoretical one.

Parameterization: Topological information loses significantly when a local mesh region is flattened to a 2D plane. There are two conditions for parameterization method to obtain a reasonable result: i) the local mesh region is planar enough or ii) the resolution of the mesh model is higher than the contour resolution. Figure 14(d) and 15(d) shows the situations that the above conditions are failed to meet, and results generated are not acceptable. When the resolution of contour is much higher than that of mesh model, the relative position between adjacent snaxels of a contour on the 2D plane is distorted. Hence, when tattoo is mapped back to 3D mesh, its basic shape could be roughly kept but detail information is distorted as we see in Figure 15(d) and 16(d) where contours become noisy.

Our method: As shown in Figure 13-16(a), results show that our method can generate consistent contour under different model and contour resolutions. With the vertex or the edge projection methods, even one may attempt generating better results by using higher resolution contours, the results are still undesirable (Ref. Figure 15(b) and 15(c)) when model resolution is limited.

V. Applications

Our proposed method makes improvement to the ACM over existing methods in terms of accuracy, performance and consistency. This does not only natively benefit mesh segmentation, but also makes contributions to different multimedia applications. Here, we discuss a number of them, which include object retrieval, virtual sculpting, and motion data recognition.

A. Object Retrieval

Object retrieval is a process to identify suitable 3D objects from a repository. Generally, two main processes are involved: feature extraction and feature matching, which determine critical characteristics from an object and computes the similarity of the extracted features among objects, respectively. To extract object features, it is typical to consider either the geometry, which deals with the shape and size, or the topology, which analyzes...
the skeleton or structure, of an object. Particularly, topology information may serve as a better feature for retrieving deformable objects [Tam07]. Also, it is common to segment an object into meaningful parts and use these parts for feature matching to improve both the performance and the accuracy of object retrieval. Practically, object retrieval may not be straightforward due to the existence of model noise and there exists models with different resolutions. To address this problem, the ACM can be deployed to obtain better object features from object parts and topology through object segmentation [Lee04] and skeleton extraction [Ma03], respectively, as the ACM can generally tolerate model noise based on its internal and external energy formulation. In addition, by integrating our proposed method to [Lee04, Ma03], we may generate more consistent object features, even though the objects in the repository are of considerably different resolutions.

B. Virtual Sculpting

Virtual sculpting [Li03] is a human-computer interaction technique, which allows a user using CyberGlove to perform direct object shape manipulation using hand gesture. Technically, it is done by casting rays that pass through both the virtual hand and the object under editing to compute and establish the relation between the virtual hand and the object. Any change in hand gesture can then be translated to become editing parameters for modifying the object shape. To extend such an interaction technique, we may apply the ACM to allow a user manipulating the shapes of meaningful object parts using hand gesture, rather than performing editing merely based on the virtual hand projection, which may sometimes lead to less meaningful sculpting results. To do this, we consider the silhouette of the virtual hand projection on the object as an initial active contour and use it to identify nearby object parts. Consequently, a user can manipulate these object parts through hand gesture, which is more intuitive. Note that we prefer using the ACM to determine object parts rather than presuming that there is a prior knowledge of these parts, as in virtual sculpting, object parts may be created on the fly.

The above extension can natively carry over to improve the concurrency control in collaborative

![Figure 13. Local enlarged figure: low resolution contour (22 snaxels) on high resolution model (24955 vertices). (a) our method; (b) edge projection method; (c) vertex projection method; (d) parameterization method.](image1)

![Figure 14. Local enlarged figure: high resolution contour (508 snaxels) on low resolution model (2491 vertices). (a) our method; (b) edge projection method; (c) vertex projection method; (d) parameterization method.](image2)

![Figure 15. Local enlarged figure: low resolution contour (22 snaxels) on low resolution model (2491 vertices). (a) our method; (b) edge projection method; (c) vertex projection method; (d) parameterization method.](image3)

![Figure 16. Local enlarged figure: high resolution contour (508 snaxels) on high resolution model (24955 vertices). (a) our method; (b) edge projection method; (c) vertex projection method; (d) parameterization method.](image4)
sculpting, where multiple remote users are editing a shared object at the same time. Typically, a server [Chim03] (or multiple servers [Ng02] in case of a large scale system) should be in place to enforce a concurrency control across these users, such that only a single user is allowed to edit certain region of the object at any time instance. Such a control is generally realized by a locking mechanism [Li03]. To define a locking region, it may be more intuitive to use object parts, which can be identified by the ACM, rather than merely based on the virtual hand projection. On the other hand, motion prediction techniques [Chan08, Chan05, Chan01] may be incorporated to enhance the quality of the object part determination by providing a more accurate estimation of hand motion in a network environment, where there exists network latency to avoid hand motion data to reach the server and relevant remote users in real-time.

C. Motion Data Recognition

Motion data comprises streams of data, in which each stream contains geometry information describing the position and orientation of each joint of an articulated character over the time. As motion data is typically large in data size, it is both hard and inefficient to perform any comparison or recognition directly against the raw motion data. Instead, an active contour based descriptor can be constructed for each motion data stream by running an iterative optimization process to minimize the energy of the active contour against an incrementally fetching in the virtual hand motion data. Based on such a construction, a repository of motion capture data can be set up for efficient retrieval and reuse, as the descriptors obtained are compact and feature preserving. In addition, the performance and accuracy of motion data recognition can be further increased by adopting our active contour projection method, as our method can generate a consistent active contour even if the resolution of the active contour itself or the input 3D data is reduced significantly.

VI. CONCLUSION

In this paper, we have introduced a face projection method for active contour model that significantly improves the previous methods in terms of accuracy and consistency. Our method avoids creating loops and thus produces smoother segmentation boundaries. Our method can benefit mesh segmentation and a number of multimedia applications by increasing their performance and accuracy. We have also presented a performance comparison among our work and existing methods.

As a future work, in order to locate the segmentation boundary more accurately, we need to try not to use the normal component of the motion displacement, such that the active contour will not deviate from the mesh surface too much. We need to amend the motion displacement by incorporating more appropriate mesh feature information.

ACKNOWLEDGMENT

The work described in this paper was partially supported by an UK EPSRC grant (Project Number: EP/G009635/1).

REFERENCES

Fan Yang received a Bachelor degree in Communication Engineering from Northwestern Polytechnical University, and has studied a MSc. in Information and Communication Engineering in National University of Defense Technology. She is currently a PhD student in Computer Science at University of Durham. Her research interests include computer graphics and e-learning.

Frederick W. B. Li received both a Bachelor of Arts (Honors) in Computing Studies and a Master of Philosophy from The Hong Kong Polytechnic University in 1994 and 1998, respectively, and a Ph.D. degree in Computer Science from the City University of Hong Kong in 2001. He is currently a Lecturer (Assistant Professor) at the University of Durham. Prior to the current appointment, he was an Assistant Professor at The Hong Kong Polytechnic University from 2003 to 2006. From 2001 to 2003, he was the project manager of a Hong Kong Government Innovation and Technology Fund (ITF) funded project. He serves as an Associate Editor of International Journal of Distance Education Technologies on Communications Technologies (Distributed and Collaborative Learning). He has been a Co-guest Editor of two special issues of JMM, and has served in the conference committee of a number of conferences, including Program Co-chair of ICWL 2007, ICWL 2008, IDET 2008 and IDET 2009, Workshop Co-chair of ICWL 2009 and U-Media 2009, and Publicity Co-Chair of ACM MTDL 2010 and U-Media 2010. Frederick Li's research interests include distributed virtual environments, computer graphics and multimedia systems.