2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Interactive Cross Cutting

Jekeon Jack Cha and Shahram Payandeh

Abstract—In this paper, we use an enhanced surface mesh model to simulate virtual dissection by progressive subdivision and re-meshing. Enhanced novel algorithms to generate interior structures that show the result of a cut that is generated by the interaction between instrument and model is used as the underlining framework. The notion of a Cross Cutting is introduced and solved to offer a more realistic interactive environment for various virtual diagnostic and dissection tasks. Our simulator supports two types of cutting: "cut-into", an instrument penetrating simulated tissues, and "cut through", an instrument cutting through tissues. In either case, a groove is developed in the path where the cutting has taken place to reflect the depth of the cut. Generation of a groove introduces a challenging problem when two cuts cross in their paths. For example, when two cutting path intersects (i.e. an occurrence of cross cutting), the current structures involved in the proximity of the cuts do not simulate a realistic intersecting cross cut. Although many studies have represented solutions to surface mesh cutting, a solution to an interactive cross cutting has not been adequately addressed yet. This paper presents an algorithm that can be flexibly applied for the simulation of an interactive cross cutting on a surface-mesh. Such solution can be further applied to many applications involving surface mesh.

I. Introduction

Virtual surgical simulation can be an alternative option for creating an efficient and reproducible training environment that can reduce the time and the cost of surgical training. Our aim is to develop an interactive training to be used for mastering basic tasks such as dissection. Virtual dissection can also be used as a complementary educational tool in post-secondary institutions. As to deliver higher degree of effectiveness of training, simulators must be able to offer realism. Our focus is to simulate a realistic cross cutting, an essential procedure of a general dissection. Previous approaches to produce realistic models include volume models using Finite Element Machine (FEM). However, FEM requires model parameters of soft tissues in order to produce computationally accurate result in solving dictating equations [1]. Boundary Element Method (BEM) is another approach where accurate real time deformable objects [2] [3] were simulated by condensing solutions into a domain boundary. However, its condensation introduced potentially expensive computational costs as the whole stiffness-matrix of the method must be recomputed when surface elements are being cut because it is usually not sparse [1]. Also, image-based rendering approach [14] has been taken in order to alleviate the computational burden on rendering tissue deformation. In our work, an interactive soft tissue model is implemented by using a surface mesh. Although the surface mesh method is a gross simplification of tissue behavior, it can offer sufficient realism for training and low computational requirements. Also, this method can accurately model actual objects from by extracting surface mesh model from the medical images. This advantage allows the realism of our simulation to be enhanced greatly without being constricted by the soft tissue parameters[13]. Our water-tight surface meshes simulate realistic human tissues where the topological changes correspond to the dissection hence a hole or losing any face of tissue surface is not desired.

We also have developed modification to surface mesh models where in addition to representing the vertices as a mass point and the connecting surface edges as a spring, we have introduced home springs to stabilize the surface mesh. Vertex displacements are calculated through solution of differential equations that model the mass-spring system. With this framework, a tension of soft tissue is simulated by initially stretching every mesh spring, hence once a cut is made, the surface spring forces will simulate realistic opening of a cut. Also, we have joined other researchers in modeling the virtual cutting tool (i.e. scalpel) as a line segment [7] [8] [9] [10] [11].

Although solutions in simulating a cut on the 3D tetrahedral meshes [7] [8] and surface meshes [4] [9] have been proposed, a cross cutting situation has not been analyzed. With the existing framework of mass-spring modeling of surface mesh and underlying cutting algorithm developed by [4], we will explore solutions to simulate realistic cross-cuts that can be also extended into many applications of 3d models.

In this paper, a triangular topology is used for modeling surface meshes where the cross cutting situation is analyzed. In Section II.A, the method of surface cutting and triangular subdivision are discussed. It also introduces its deficiency in handling a cross-cut situation. In Section II.B, we identify the parameters that can be used in classifying and handling the cross cuts and we propose an algorithm for managing cross cuts in Section II.C. The results are discussed in Section III, followed by the concluding remarks and some future research work in Section IV.

II. Problem Definition

A. Cutting Operation

A realistic simulation of soft tissue dissection (i.e. progressively cutting through a tissue model represented by

J.Cha is with Experimental Robotics Laboratory, School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, CANADA jchaa@sfu.ca
S. Payandeh is with Experimental Robotics Laboratory, School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, CANADA shahram@ensc.sfu.ca

the patches of triangles) is implemented by a triangle subdivision method. The subdivision of each triangle is a function of the state of interaction with the instrument. A state machine is used to model the two types of progressive cutting through the usage of a blade (or a knife): cutting *into* an object, and cutting *through* an object [12].

For our simulation, the movement of a cutting instrument while penetrating an object defines a cutting operation. By representing the scalpel as a line segment, the detection of a collision between the line segments and mesh triangles constitutes the start of a cut.

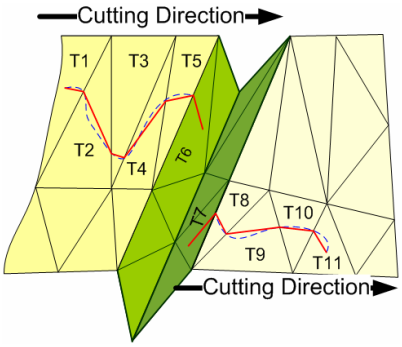


Fig. 1. Example of cutting operation

Fig. 1 depicts a situation where a normal cut (i.e. cutting from T1 to T5 and T8 to T11) is combined with a cut trough a groove triangle (i.e. T6 and T7). Such cutting path is crossing another cutting path and yields a cross-cut situation. Start of a normal cut is initiated once the instrument makes contact with the object, e.g. penetrating and moving across triangle T1 and T7 in Fig. 1. As the instrument moves, the underlying object mesh is modified by local subdivision, described in TABLE I. In Fig. 2, an example of progressive subdivision is shown. From inside the original triangle ABC, penetration is made and subdivision according to the start state of cut-through edge results (Fig. 2a). As the knife travels, further subdivision occurs as shown in Fig. 2b, hence progressive.

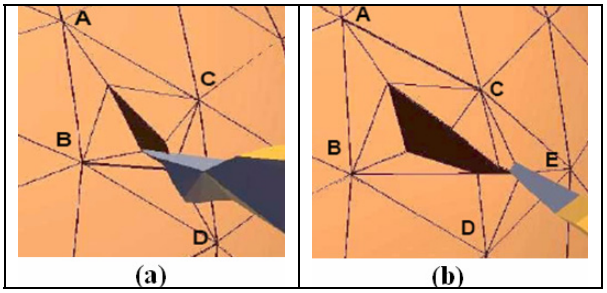
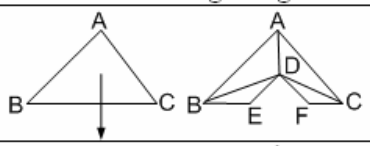
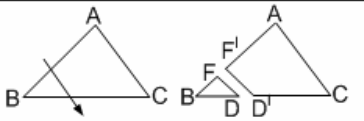
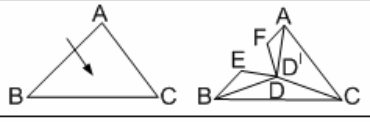
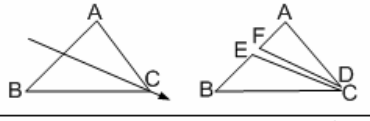
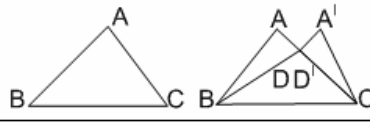


Fig. 2. Example of progressive subdivision a) Scalpel cuts out from edge BC to triangle BCD; b) Scalpel cuts out from edge CD

The instrument path inside a triangle may not be a straight line, e.g. the dashed line in Fig. 1. However we simplify any path to be a straight line connecting the first and last intersection points in a triangle (solid line). When the instrument is lifted up and contact with the object is lost, the algorithm enters a termination state. For example, triangle T6 and T11 are the last triangles that the instrument has contacted. Triangles like T1 and T7 are called *start state* triangles and T6 and T11 are called *termination* state triangles where as T2…T5 and T8 …T10 are referred to as *midway* triangles. In our study, we will focus on the subdivision algorithm of groove triangles like T6 and T7 where cutting path is the position of instrument's line segment existing or entering the groove.

TABLE I
SUBDIVISION METHOD OF CUTTING THROUGH EDGES AND VERTICES AND ITS DIFFERENT STATE

State Change	Cut-Through Edge
Start State	
Midway State	
Termination State	
State Change	**Cut-Through Vertex**
Start State	
Midway State	
Termination State	

Because inside of a cut on a surface mesh has no representation, it will appears empty. Therefore, a method of generating structures in the opening of a cut has been developed as a function of the depth of the instrument penetration as shown in Fig. 3a. Instrument tip positions are used at the start and end of the cut inside one surface triangle to define the bottom of a *groove* (B1 and B2 in Fig. 3a). A subdivision of surface triangles generates SL1 and SL2 (left side of the cut) and SR1 and SR2 (right side) respectively.

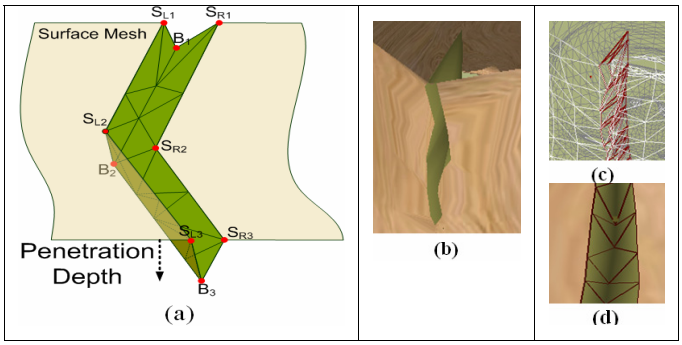


Fig. 3. a) Knife penetration into surface creating groove; b) Simulation of a cut; c) Wire frame display of groove triangles generated (red lines); d) Groove triangles generated viewed from surface angle

In case of a cross cut, two of such grooves get generated and intersect. The current drawback of most of the virtual cutting approaches is that they do not support the cross cutting and do not support proper correlation between two grooves, as shown in Fig. 4. In Fig. 4, We observe that the surface triangles do not 'open up' realistically as knife enters and leaves an existing groove trench at point #2 and #1 respectively as a result of improper handling of topological changes at the cross cutting intersection.

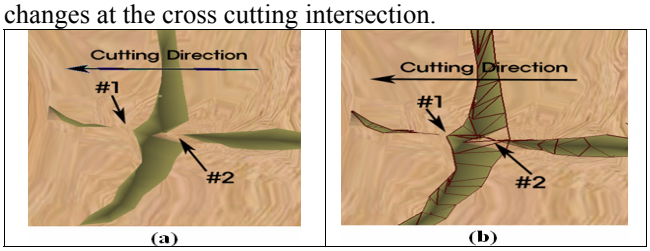


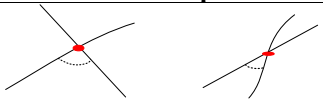
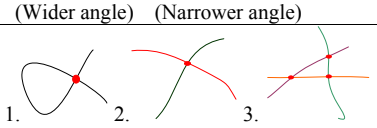
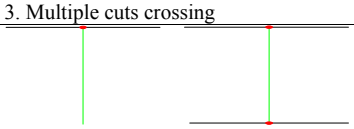
Fig. 4. a) Example of improper cross-cut, intersection of two grooves; b) Display of inappropriate groove vertices and edges interaction of two cuts

Without the handling of cross cuts, simulation of training in surgery and dissection will lose realism and accommodation for arbitrary cuts. For example, in [10], realistic I-cut along dotted guideline is desired for training of frog dissection. Also, improper topological structures generated at the intersection of the cross cut may become a potential source of instability of simulation system.

B. Cross Cuts and its Parameters

Solution to the cross cutting problem is developed preliminarily by experimentally studying various cross cuttings types and by classifying these into general templates of cross cutting. We have selected the following parameters to categorize parameters of a cross-cut; an angle measured between two planes that contains two cutting segment as depicted in Fig. 5; multiplicity of a cross cut (e.g. whether a single cut looped around to cross its own path); end points of each cut; and depth or penetration of two intersecting grooves. Examples of each parameter are shown in TABLE II.

TABLE II
EXAMPLES OF EACH PARAMETER

Parameters	Examples
Angle between two cuts	 (Wider angle) (Narrower angle)
Multiplicity	 1. 2. 3. 1. One cut crossing itself 2. 2nd cut crossing 1st cut 3. Multiple cuts crossing
End points of cut	 (T cut) (I cut)
Depth of grooves at intersection	Depth of first cut's penetration > Depth of second cut's penetration Depth of first cut's penetration < Depth of second cut's penetration

Multiplicity parameter can be omitted if cross cut is to be detected whenever a cutting tool (i.e. knife) comes in contact with a groove triangle (i.e. green triangles in Fig. 4b) Also, the end points of the second cut of "T" and "I" cuts can be generalized by this approach since "T" and "I" cut can be treated as one or two instances respectively of the cutting tool touching a groove triangle. The parameters can be narrowed to the remaining two in defining a cross cut. An angle between first and the second cut can be further defined as the angle between the top segment of the groove triangle where the knife hits and the plane projected from second cut's cutting path to the point where knife's cutting line segment hits the top line. The cutting path is projected from an interpolated line connected by the bottom vertices of produced trench of the second cut (i.e. $^2B_1$ to $^2B_3$ in Fig. 5 and Fig. 6). The angle is calculated by two vectors V1 and V2 at $P_{intersect}$ in Fig. 5.

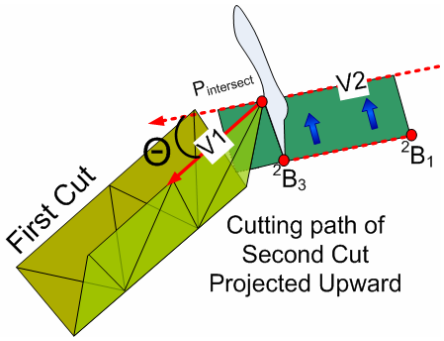


Fig. 5. Angle parameter

If the angle measured is calculated to be less than $\theta_{threshold}$, two groove trenches are merged by connecting last bottom vertex of the second cut (ex) $^2B_3$ in Fig. 6) and the bottom vertex of a first cut that is connected to $P_{intersect}$ (ex) $^1S_{L1}$ or $^2S_{L3}$ in Fig. 6**)** and connecting corresponding side vertices connected to these bottom vertices. As result, we generate new triangles in doing so as depicted in Fig. 6.

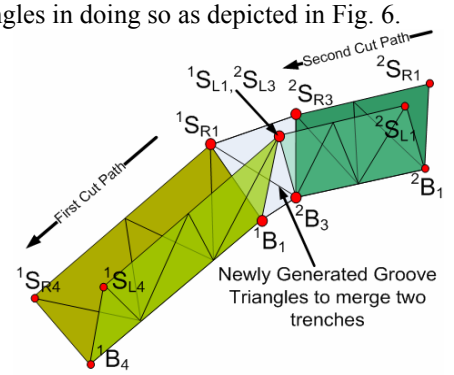


Fig. 6. Merging two trenches when angle between two cut is less than $\theta_{threshold}$

Also, the depth of each cut will affect how we correctly handle a cross cut. As shown in Fig. 7, when a second cut crosses the first cut, a cross cut occurs. Fig. 7 shows two possible cases where the second cut's penetration depth is smaller (Fig. 7a, b) and larger (Fig. 7c, d) than the first cut's penetration depth. In the first case, the points on second cut's grooves will be used in simulating opening of the cross cut as shown in Fig. 7a. In the second case where second cut has a

deeper depth, points on the first cut will be used to handle the cross cut as shown in Fig. 7c.

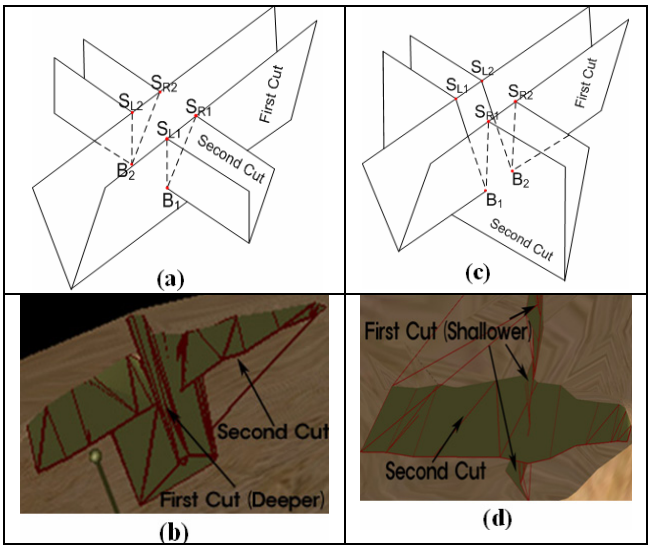


Fig. 7. a) Depth of second cut shallower than the first cut; b) Cutting path displayed for second cut being shallower in simulation; c) Depth of second cut deeper than the first, view from underneath the surface mesh; d) Cutting path displayed for second cut being deeper in simulation, view from underneath the surface

C. *Cross Cutting Method*

In order to detect occurrences of such cross cuts, we retain a supervisory run-time algorithm which records the motion of a cutting tool in the duration of cut penetration. Also, when new triangles are generated as a result of subdivision, each triangle is marked either as a surface triangle or as a groove triangle by a flag bit. Hence, if the triangle that the knife penetrating is a groove triangle, we determine that a cross cut is occurring. The recorded motion path is also used in determining the angle parameter as illustrated in Fig. 5. Another global tracker of the coordinates of cuts (i.e. coordinates of groove bottom and side vertices) and position of the cutting tool at the time of penetration is maintained in order to determine depth of a first cut at the time of intersection in comparison with a second cut.

Based on the pattern of the cut in comparison with TABLE II, each cross cut will be handled according to an algorithm represented in Fig. 10. First task is to detect a contact between a cutting tool and a side triangle belonging to a groove. Direction of the cutting tool approaching the groove triangle defines whether the contact was from outside or inside. For examples, in Fig. 1, cutting from penetrating T1 to T5 and hitting T6 would make a contact between the side triangle and the knife from "inside" the mesh structure. Conversely, cut from T7 to T11 originates from outside of the mesh structure hence makes a contact from "outside". In determining whether the cross cut will yield a circular cut or not, we have used 15° for the angle threshold value ($\Theta_{Threshold}$) in our implementation. To connect such circular cut, we connect two groove trenches together by joining respective vertices. In a crossing cut situation, if the first cut is deeper as in Fig. 7a, we can use normal subdivision on the groove triangle T1 in Fig. 8 as Start State.

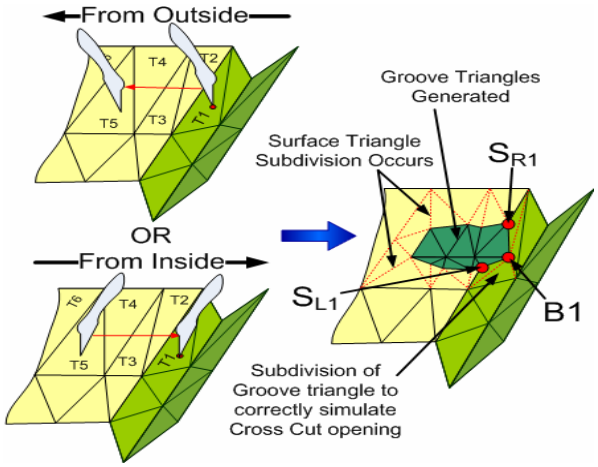


Fig. 8. Subdivision method when first cut is deeper than the second cut

In case first cut is shallower than the incoming second cut, normal subdivision method is insufficient. We must change the structure of the first cut's groove trench and treat it as if the second cut was made first. This is depicted in Fig. 9.

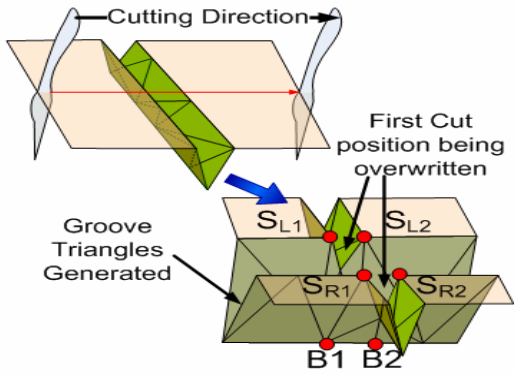


Fig. 9. Subdivision method when first cut is shallower than the second cut

Implementation of this algorithm is described as pseudo code in Fig. 10.

```
If (knife is in contact with a groove triangle)
    If (knife hitting groove triangle from outside)
        If ( Depth of First Cut >= Depth of Second Cut)
            Subdivide groove triangle in contact as start state of subdivision
            Start second cut's groove trench with SL1, SR1, and B1 as in Fig 8
        Else if ( Depth of First Cut < Depth of Second Cut)
            Generate start of groove triangle for second cut at SL2 to B2 to SR2 in Fig 9
            Delete First Cut's groove trench from SL2 to B2 to SR2 in Fig 9
    Else if (knife hitting groove triangle from inside)
        Angle = Calculate_Angle_Between_Two_Cuts_as_Fig 5 ( )
        If (Angle < θthreshold)
            Merge two groove trenches as one as shown in Fig 6
        Else
            If ( Depth of First Cut >= Depth of Second Cut)
                Subdivide groove triangle as normal
                Mark newly generated triangle as groove triangle
            Else If ( Depth of First Cut < Depth of Second Cut)
                Generate groove triangle for second cut
                Delete First Cut's groove trench from SL1 to B1 to SR1 in Fig 9
            End If
    End If
Else If (knife is in contact with a surface triangle)
        Surface Triangle Subdivision & Groove Trench Generation
End If
```

Fig. 10. Pseudo code of Cross-Cut handling

III. Results

Simulation results for handling different types of cross cuts are exhibited in TABLE III and third column of the table shows groove triangles generated as a result of crossing of two surface cuts. The groove triangles at the cross cut intersection are modified according to the algorithm we described in Fig. 10. Although some of the cross cut yielded realistic opening of intersection (i.e. correct stretch of the opening and smooth connection between two groove trenches, etc), some of the results also displays some of challenges we have faced. For example, some of the opening did not stretch enough. Also, when the user wanted to make a cut across an existing intersection of another cross cut, the topology of triangles at cross cut intersection was too complicate to handle. Another fundamental challenge we are facing not only in this study but in collective effort in tissue dissection simulation was generation of tiny triangles. A series of subdivision imposed upon a triangle produces number of tiny triangles that are beyond our triangle mesh resolution. When a knife crosses a considerably tiny triangle, computation error takes significant effect on our cross cutting algorithm as well as the general subdivision algorithm.

In order to measure performance of our algorithm in relation with the size of a groove triangle the cutting tool comes in contact with at both times of entering and leaving the groove trench, we have plotted the relative quality of a cross cut in Fig. 11 with respect to the size of the groove triangle. Quality is measured on scale out of ten, counting on how realistic cross cut is simulated (texture error rate, smoothness, opening of the cut, etc).

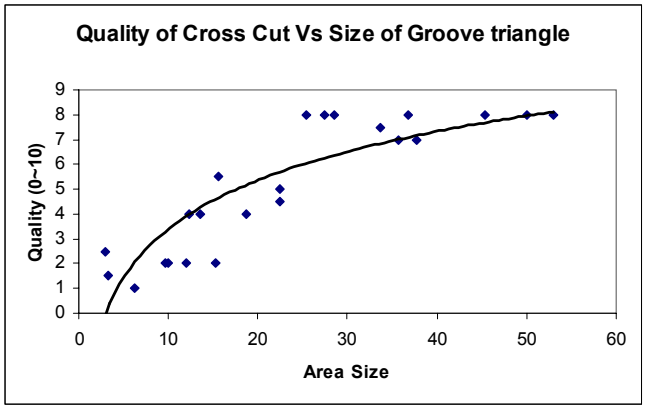
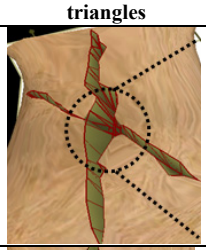
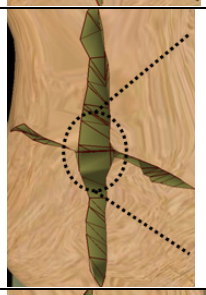
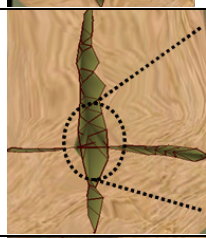
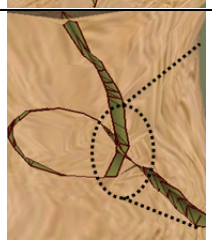
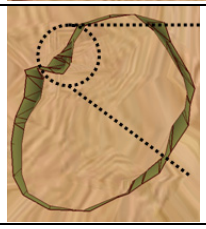
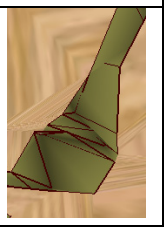


Fig. 11. Size of groove triangle and Failure rate of cross cutting

As you can see, performance of the algorithm significantly worsened when area of a triangle was below ~15 normalized units as topological errors and texture errors (perceiving groove triangles as surface triangle as the result of computational error). The quality scale was a subjective decision where 10 was a cut without any topological errors (e.g. texture, no missing triangle, a hole, etc).

TABLE III

BEST SIMULATION RESULTS OF VARIOUS TYPES OF CROSS CUT

Simulation & Types	Display of groove triangles	Cross cutting Intersection
Angle ~ 70°		
Angle ~ 80°		
Angle ~ 90°		
Multiplicity		
End Point Cut; I cut		
End Point Cut; T cut		
Circular Cut		

Also, haptic force feedback to the user is an important component in inducing realism. Fig. 12 shows the feedback force magnitude that was given in response to general cases of tissue dissection. Graph (a) of Fig. 12 illustrates the force fed back to the user when a virtual knife has touched the tissue and penetrates into the skin. As you can see, there is a threshold force needed to penetrate the tissue. After the penetration, a knife continues cutting the tissue. Graph (b) corresponds to the time where a knife is cutting along as in Fig. 2. The final force feedback is composed of three kinds - resistance, friction and side force. The resistance is proportional to the depth of the penetration of a cut. The friction force is also proportional to the resistance magnitude, but in the direction of the knife. Finally, the side force is a multiple of the displacement vector between current knife tip position and the previous position. Hence in Graph (b), the user is cutting along a tissue with increasing penetration depth in the first two third of a way, and decreasing the depth in the last third of the cutting path.

Finally, when a crosscut situation occurs, the cut-along movement of the knife will meet a groove face (for example, T6 of Fig. 1) and will exit the mesh. When the knife has exited the mesh, force feedback will be reduced to zero as shown in the sudden drop of force magnitude in Graph (c). However, when the knife crosses the groove and re-enters the tissue mesh (for example, T7 of Fig 1), the force feedback will be present again (i.e. increasing force magnitude latter half of Graph (c)).

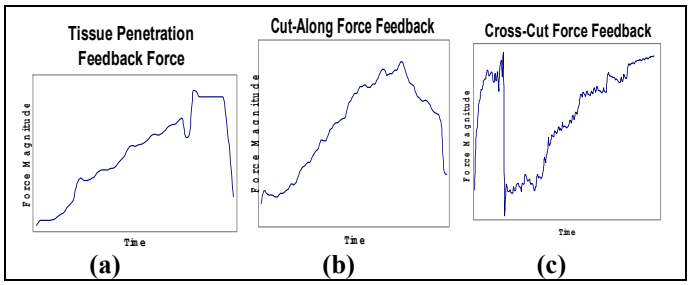


Fig. 12. Feedback Force Magnitude vs. Tine

IV. Conclusion and Future Work

Simulation of realistic cross-cuts is essential component in dissection training. In this paper, we have proposed a method for handling of various types of cross cuts with respect to surface meshing in real time. We have defined an occurrence of a cross cut as an instance when a cutting tool comes in contact with a groove triangle (for example, T6 and T7 in Fig. 1). Also, we have identified three factors or parameters of a cross cut that characterize a cross cut so we can use this information to decide how to handle each cross cuts. Angle between the initial and the second cut, depth of initial and second cut, and whether the cutting tool (i.e. knife) enters or exits the existing groove trench of the first cut were the factors used in how to treat each cross cut and an algorithm developed is shown in Fig. 10. Although this algorithm, used in real time with progressive subdivision to simulate tissue dissecting, produced encouraging result shown in TABLE III, our algorithm displayed limited flexibility to the size of the groove triangle as plotted in Fig. 11. But even for real tissue dissection, when tissue area becomes too small compared to the knife being used, dissection operation becomes a 'scraping' situation. We recognized triangle size, in other words, triangle mesh resolution, as the next immediate technical challenge in the realization of cross-cut and general tissue dissection application. For future work, we are currently developing more versatile solutions that response well to wider range of triangle size as we work to find a better solution in dealing with tiny triangle generated. To sophisticate our cross cutting even more, we are to store extensive cross cut information (i.e. detailed topological information at the cross cut) to increase level of data input. In doing so, we are hoping to keep track of locations of each cross cut and be able to handle even multiple cross cuts at one intersection.

References

[1] D. Bielser, P. Glardon, M. Teschner, and M. Gross. A state machine for real-time cutting of tetrahedral meshes. In *Proc. of Paci_c Graphics*, pages 377ñ386, 2003.

[2] D. Bielser, V.A. Maiwald, and M.H. Gross. Interactive cuts through 3-dimensional soft tissue. In *Proceedings of the Eurographics '99*, volume 18, pages C31ñC38, 1999.

[3] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(8):437ñ452, 2000.

[4] A.R. Mor and T. Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *CVRMed-Proceedings*, volume 19, pages 598ñ607, 2000.

[5] C. Basdogan, Chih-Hao, and M. A. Srinivasan. ì Simulation of tissue cutting and bleeding for laparoscopic surgery using auxiliary surfaces.î In *Medicine Meets Virtual Reality*, pages 38ñ44. IOS Press, 1999.

[6] D. Bielser and M. H. Gross. ì Interactive simulation of surgical cut procedures. In *Proceedings of the Pacific Graphics 2000*,pages 116ñ125, 2000.

[7] A. B. Mor and T. Kanade. ìModifying soft tissue models: Progressive cutting with minimal new element creation.î In *CVRMed-Proceedings*, volume 19, pages 598ñ607, 2000.

[8] H.-W. Nienhuys and A. F. van der Stappen. ìCombining finite element deformation with cutting for surgery simulations.î In *Proceedings of the Eurographics i00*, pages 274ñ277, 2000.

[9] D. Serby, M. Harders, and G. SzÈkely. ì A new approach to cutting into finite element models.î In *Procs. of the Fourth International Conference on Medical Image*, pages 425ñ433, 2001.

[10] N. Vafai and S. Payandeh, "Toward Haptic Rendering for a Virtual Dissection", to be presented in ICMI 2006, Banff, Canada, November 2-4, 2006

[11] Soft Tissue Deformation and Cutting Simulation for the Multimodal Surgery Training, Yi-Je Lim; Hu, J.; Chu-Yin Chang; Tardella, N.; Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on 22-23 June 2006 Page(s):635 – 640

[12] Zhang, H., Payandeh, S. and Dill, J., (2004) "On Cutting and Dissection of Virtual Deformable Objects", Proceedings of IEEE International Conference on Robotics and Automation, pp. 3908-3913

[13] Lin, R., Payandeh, S. "Preliminarily Experimental Study of Isosurface Extraction for Surgical Training Environment", Proceedings of HAPTEX, VR Workshop on Haptic and Tactile Perception of Deformable Objects, 2005

[14] ElHelw M., Chung A., Darzi A. and Yang G. "Image-Based Modelling of Soft Tissue Deformation", Medical Image Computing and Computer-Assisted Intervention, 2003 Page(s): 83-90