The Haptic Rendering of Polygonal Models involving Deformable Tools

S. D. Laycock[1] and A. M. Day[1]

School of Information Systems, University of East Anglia, Norwich, NR4 7TJ, UK
S.Laycock@uea.ac.uk, AMD@sys.uea.ac.uk

Abstract. The sophistication of simulations for training and analysis is increasing with users demanding higher quality virtual environments. Complex polygonal models are often required to achieve the desired realism but put a strain on the necessary frame rates. Haptic Feedback is now used in many simulations for applications ranging from medical training to virtual prototyping. With the high haptic update rate and the desire to include complex polygonal models a framework is required which can represent a complex scene for haptic applications. Rigid tools are often used in haptic simulations such as scalpels in surgical applications. However, very often the rigid property of the simulated tool does not correspond to the flexibility of the tool in the real world. This paper describes a method for the haptic rendering of deformable tools to improve the realism of interactive simulations.

1 Introduction

The ability to touch virtual objects on the screen by way of haptic feedback devices has brought a higher level of realism to virtual environments. It has broadened the area of applications to include a wide range of interactive training simulations. As these applications become more established the users will begin to expect a higher degree of realism to suit the needs of their applications. The work described in this paper aims to increase the realism of those simulations requiring tool use in a virtual environment. A variety of haptic devices have been produced for research purposes and a small subset of these are commercially available. In this work the Phantom

Desktop produced by SensAble Technologies has been utilised [18]. For a recent survey on the developments of haptic devices and applications see [13].

For most haptic feedback applications the user is limited to interacting with the objects in the scene by probing with fixed points, using either a single point or a collection of points strategically located on the surface of an object. For the complete inclusion of tools in a virtual environment these current approaches are unsatisfactory. The first phase of our work aims to devise a haptic rendering algorithm for three dimensional tools. The second phase of our work is to include deformable tools in the virtual environment with haptic feedback. Initially we investigate the realistic interactions between deformable tools and rigid objects in a virtual environment. It is the restriction to rigid tools in haptic environments that motivates our work with the aim of achieving an increased realism for use in training simulations such as those used in dentistry. Certain dental operations, such as root canal surgery, could not be simulated with rigid tools as it is the flexibility of the tool that is fundamental to a successful operation. In addition to the flexibility, the sensations of touch are very important, as the dentist does not have a clear view of the treatment area.

2 Previous Work

Haptic Rendering is the process of calculating a reaction force for a specified position of the haptic input device. For the Phantom device the position is represented as a single point in the virtual environment. This point, which corresponds to the endpoint of the haptic device, is known as the Haptic Interface Point. A typical haptic rendering algorithm models the virtual environment as intersecting planes, and models the penetration of the point inside the plane using a spring-damper model. The first stage of a typical rendering algorithm involves collision detection. There are now an abundance of collision detection methods for determining intersections between a variety of geometric primitives. Min et. al. [15] published a survey on collision detection methods, including public domain software packages such as V-collide and SOLID.

To optimise the computation when determining collisions for haptic rendering it is possible to reduce the tool to a single point or to a collection of test points. This approach can be too approximate in certain cases such as those where a tool is required to slide over another object. More test points may be required to prevent intersections being missed which leads to a greater number of computations. A test point approach has been used in [14] and works reasonably well for simple polygonal models but the level of computation time is too high for complex scenes.

The second stage of a typical haptic rendering algorithm requires the intersected areas of the polygonal object to be determined and the normal of the face to be retrieved. A common approach is to use a penalty method where a force is returned based upon the penetration of the point inside the object. Zilles and Salisbury [20] and Ruspini et. al. [17] both illustrate the problems that can occur when using the penalty method. A constraint based method was developed by Zilles and Salisbury [20] to address this problem. In 1997 Ruspini et. al. described a similar method for calculating appropriate forces. Basdogan et. al. developed a method for allowing ray-based interactions to take place [2]. One of the aims of our work is to allow three dimensional objects to be used as tools which will interact with polygonal objects in a virtual environment.

Currently haptic feedback has been used in conjunction with virtual rigid tools in a number of applications ranging from surgery [9] to assembly tasks [19]. The interactions tend to be based on a single point, line segment or collection of points using the methods cited previously. Developing simulations with rigid tools induces a large overhead when determining accurate reaction forces, particularly when there are multiple points of contact. A magnetic levitation haptic interface device [7] has been developed and applied to simulating rigid tools. One test example simulates a virtual key and lock.

For certain applications the flexibility of tools should also be considered, otherwise the realism of the simulation will be unsatisfactory. To achieve systems that correspond to the real world combinations of hardware representing tools and software have been developed. One application aims to train medical staff in performing laparoscopy and endoscopy procedures. It incorporates haptic feedback conveyed from holding the handles of the real tools as they interact with a plastic replica of a human body part. The ENT Surgical Simulator [9] was developed for endoscopic sinus surgery simulation where the operator controls a replica of an endoscope.

Koutek and Post [12] have developed a method modelling flexible objects, using spring-damper models. Their approach allows flexible forks and probes to aid in object manipulation in a virtual environment. Reznik and Laugier [16] have produced a more complex deformable object. Their work involves simulating the control and deformations of a virtual fingertip as it probes a rigid surface. Both of the above procedures exhibit flexible behaviour but are not linked with force feedback.

Modeling deformable objects using spring models, as in the two methods discussed above, are useful as they are relatively easy to include in applications and their computation requirement is small. However, they are not based on the mechanical properties of the simulated material and therefore are sometimes too approximate. Finite Element Analysis, based on continuum mechanics is a much better approximation although the computation time is greatly increased. Berkeley proposed a new method called the Banded Matrix Method for greatly reducing the finite element methods computation time to allow it to be used in real-time applications [5]. The techniques were originally developed for real time skin surgery. Later Berkeley et. al. used this method to allow deformable objects to be probed incorporating haptic feedback [6].

Deformable tools have recently been incorporated with haptic feedback to a small extent in two applications. One example illustrates virtual painting using a three dimensional paintbrush to paint on the canvas [3]. The bristles of the brush deform based on a spring-damper model. Another example of simulating bristles with haptic feedback has been developed by Gockel et. al [11]. The bristles simulated are incorporated into a virtual model of a tooth brush.

The use of deformable tools in haptic environments is still in its infancy. An approach is required that will be able to model the physical behaviour of the deformations, coupled with a haptic rendering algorithm for incorporating tools. This algorithm should be capable of determining the reaction forces at the contact points anywhere over the surface of a three dimensional object.

3 The Haptic Rendering of a Simple Tool

The algorithm for the haptic rendering of tools introduced here is a constraint based method which allows contacts to be determined between any point on the surface of the tool and any point on the surface of a virtual object in the scene. The strategy for the algorithm is to ensure that a tool, the Visual Tool, always stays on the surface of the objects whilst a second tool, the Haptic Interface Tool, related to the haptic device is allowed to penetrate the objects. The Haptic Interface Tool will be located at the position of the Haptic Interface Point in the virtual environment initially aligned with the handle of the Phantom device. While the Haptic Interface Tool will be permitted to penetrate the virtual objects the Visual Tool will not. Instead, this tool will be constrained to the surface and displayed to the user as the virtual tool they are manipulating. A spring damper model can then be used to model a force between the two representations of the tools. This constraint method allows stable forces to be determined based on the penetration of the tool without the visual artifacts of the tool penetrating the surface of the objects in the scene.

The algorithm developed can be divided into stages. The first stage is to determine when a collision occurs between convex polygonal models representing the tool and one of the objects in the scene. Once a collision has been recorded the points of contact on each of the colliding objects need to be stored. The contact points can then be used to determine how to track the Visual Tool on the surface of the polygonal model. The tracking phase is required to determine the correct contacted face efficiently. The tracking stops when the user moves the tool away from the object. The contact points may need to be updated during the tracking phase. This is due to the fact that as the orientation of the tool changes during a simulation different parts of the tool will come into contact with the object. The rest of this section provides details of the tracking part of the algorithm. The next section discusses the free movement of the contact points that is required during the tracking phase.

Initially the Visual Tool and the Haptic Interface Tool will be located at the same position. Once a collision is determined between the Haptic Interface Tool and an object in the scene the Visual Tool will be constrained to the contact position on the surface of the object. This stage has been determined by finding the pair of closest points on the surfaces of the two intersecting objects, using SOLID, a 3D collision detection library [44]. At the point of contact the Visual Tool and the Haptic Interface Tool are still close together.

Figure 1 illustrates the process of determining the closest pair of points. The solid object represents the Haptic Interface Tool and the dotted outline represents the Visual Tool. The diagram on the left of figure 1 illustrates that before a collision both tools are located at the same position. The diagram on the right illustrates a later stage when the point of collision has been determined. The points of collision are identified by the black circles. The tool has moved in from the top left and has slightly penetrated the box as SOLID determines that a collision has taken place. The top black circle marks the position of the contact point located on the top surface of the box and the lower black circle represents the contact point located on the Haptic Interface Tool. It is now necessary to transform the Visual Tool to the contact point on the intersected object. This will ensure that the Visual Tool will not penetrate the object. The orientation of the Visual Tool is taken to be equal to the orientation of the Haptic Interface Tool.

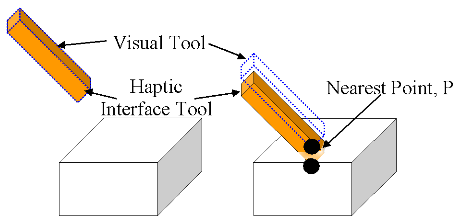

Fig. 1. Determining the Nearest Points, Left: Visual and Haptic Interface Tools are collocated, Right: Contact Point identified and the Visual Tool positioned.

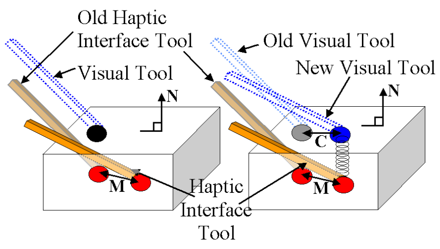

Fig. 2. Tracking the Haptic Interface Point, Left: Tools shown at previous time step along with the current Haptic Interface Tool, Right: Visual Tool tracked along with the current face.

The next stage of the algorithm needs to deal with the process of tracking the Haptic Interface Tool and rendering the Visual Tool appropriately constrained to the surface of the intersected object. This stage is performed as long as the Haptic Interface Tool remains in collision with the object.

To constrain the Visual Tool to the closest point compared to the Haptic Interface Tool the first stage will be to move the Visual Tool along the surface of the object. This can be achieved by determining the component, C, of the last stage of the Haptic Interface Tools movement, M, that runs perpendicular to the current normal. Equations (1) and (2) can be used to obtain this component.

$$Movement, M = NewPos - OldPos \tag{1}$$

$$PerpendicularComponent, C = M - (M.N)N \tag{2}$$

Where	$NewPos$	= The New position of the Haptic Interface Tool.
	$OldPos$	= The Old position of the Haptic Interface Tool.
	N	= Normal of the current contacted component.

The normal in the above equation will be determined from the normal of the geometric component that the Visual Tool is closest to. To reduce the time complexity of the algorithm at this stage the normals for each of the geometric components (Edge, Vertex, Face) will be precomputed in an initialization phase and stored in a data structure. This data structure is discussed later in this section.

Figure 2 illustrates the tracking procedure as the tool is moved in the direction of the vector, M. The left hand diagram of figure 2 shows the old positions of the Visual Tool and the Haptic Interface Tool. It shows the direction of movement of the Haptic Interface tool and it's new orientation. The right hand diagram illustrates the new position of the Visual Tool when the old Visual Tool is translated along the perpendicular vector, C. The orientation of the Visual tool is once again assigned the same orientation as the Haptic Interface Tool. At this point the contact force can be determined based on a spring between the contact points on the Visual and Haptic Interface Tools.

The next section of the algorithm is required to deal with the tracking of the Visual Tool over all the geometric components in the model. The important aspect is to be able to track from one component to an adjacent component efficiently. To this end a data structure has been employed to store uniquely the edges, faces and vertices of any convex model represented as lists of vertices and faces contained in a file. For each component stored pointers are specified indicating the adjacent components along with the current components normal vector. Obtaining the possible component that the Visual Tool could come into contact with becomes an efficient operation. For instance if the current contacted component is a face then to determine the next component requires at most three distance to line segment calculations. A threshold is provided to simulate the edges as thin flat rectangles with their orientation defined by the direction of the edges normal.

Figure 3 illustrates the transition of the Visual Tool from an edge to a face. Initially the tracking procedure is performed followed by a signed distance test to ensure the contact point is sufficiently near to the current component. In this case the end point has moved a sufficient distance and the current component is set to the adjacent face. The distances between the Visual Tools in the figure have been exaggerated for illustrative purposes. The contact point is now required to be constrained to the surface point. This procedure can be solved simply by determining the distance between the current face and the contact point on the Visual Tool.

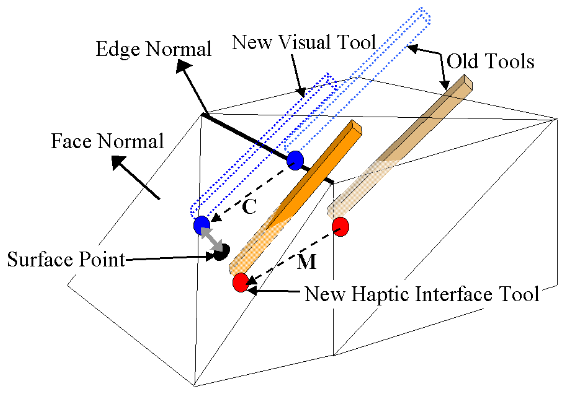

Fig. 3. The transition of the Haptic Interface Point from an edge to an adjacent face.

The current algorithm described will track the points over the surface of a convex polygonal model. However, there are some notable problems that arise due to the fact that the positions of the contact points on the surface of the tool will not remain fixed during the tracking phase. It can be envisaged that if the tools orientation is altered, which is highly probable, the tool will intersect the polygonal object at a different point. The next section illustrates the procedure used to update the contact point to permit freedom of movement over the surface of the tool.

4 Updating the Nearest Contact Points

The current contact points must be modified if the situations illustrated in figures 4 and 5 arise.

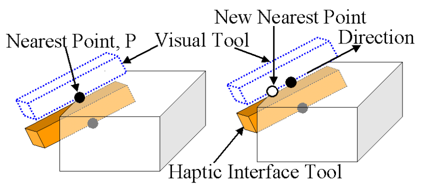

Fig. 4. Updating the contact points during sliding.

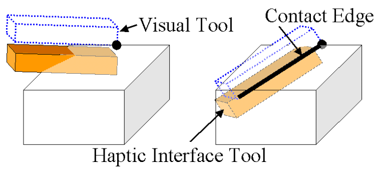

Fig. 5. Updating the contact points during rotation.

A new contact point is required if the current contact point located on the Visual Tool leaves the surface of the contact object while the Haptic Interface tool remains in collision. The two diagrams in figure 4 illustrate how this could occur. The left diagram shows the tool in contact with the edge of the white box. The nearest point, P, is calculated using the approach as described previously. The right diagram shows the positions of the tools at the next time step. The user has moved the tool, in the direction indicated by the arrow, expecting it to slide over one of the edges of the box. At this point it can be observed that the current contact point on the Visual Tool is no longer in contact with the object and a new contact point is required in order to achieve the desired sliding affect. To determine the new point of contact a ray is cast from the old contact point towards the Visual Tool in the direction of the current component's normal. The point of intersection obtained can then be transformed to the surface of the current object and set as the new contact point. This new point is indicated by the unfilled black circle in figure 4.

For the haptic rendering of points or spheres their orientation is not important. However, for the inclusion of Haptic Interface Tools the orientation is crucial in preventing inappropriate intersections with the polygonal objects. Figure 5 illustrates the problem that can occur. The left diagram shows the tool in contact with the top of the box and that the Visual Tool is constrained to this surface. The problem will occur if the tool is rotated as highlighted in the diagram on the right, depicting a later time step. As the tool is rotated it can be seen that the Visual Tool will penetrate the top of the box. At this point the algorithm should choose points along the edge in contact with the box, indicated with a thick black line in figure 5.

The approach developed to avoid this requires the Visual Tool to be transformed away from the contact object in the direction of the current normal. This ensures that the old contact point is no longer in contact and then the collision routine will be able to determine a new point of contact if one exists. The Visual Tool can now be transformed to the new contact point on the surface of the object to achieve the desired result.

With these extra constraints in place the Visual Tool and contact points will be appropriately positioned to ensure forces can be determined. This will allow the rigid tool to be used in the virtual environment. To further increase the realism of tool use in virtual environments deformable tools can be incorporated.

5 The Deformable Tool

By incorporating the physical properties of the tools in the virtual environment one can achieve an enhanced realism of simulation. The flexibility of the interaction tool is paramount in a number of applications and this inclusion forms the second phase of our work. The finite element method has been applied to this application to model the deformations of the tool using beam theory.

5.1 The Finite Element Method

The finite element method works by decomposing an object into smaller parts, known as elements. Physical formulae can then be applied to the elements individually before being assembled to describe the physical behaviour of the entire tool [10]. The physical formulae are able to represent the properties of the simulated material. The theory of a beam has been incorporated to determine the displacements of the elements of the tool when under specified loads. The formulae for the beam elements can be obtained from standard deflection formulae describing the nodal displacements [8]. Three dimensional beam elements have been incorporated to approximate the tool. Each element comprises of two end points, called nodes, each with six degrees of freedom. The degrees of freedom correspond to the translations and rotations about each of the axes. In this simulation the rotation about the x axis has not been considered and so each element has ten degrees of freedom. The removal of the twisting moment of the tool has reduced the number of computations and has not hindered in the results since the main deflections will exist about the z and y axes. The equations for the nodal displacements can be written in matrix form. This matrix is the stiffness matrix for a single element [1]. The stiffness matrix constitutes the equations that describe the displacements of each degree of freedom.

The next stage is to use the single element stiffness matrix for each element to describe the equations for the entire tool consequently the global stiffness matrix is formed. To show how this has been achieved an example of a tool partitioned into two elements will be described. Figure 6 illustrates a tool with the two elements labelled A and B and three nodes, denoted by black circles. The degrees of freedom of each node are labelled on the diagram.

The single element stiffness matrices for A and B need to be concatenated together. This procedure works by merging the equations which effect the nodes for both elements. Figure 7 illustrates the concatenation procedure. It shows a rectangle, representing the global stiffness matrix divided into smaller boxes. Each box representing the locations of the formulae linked to each node. It is clear from Figure 6 that elements A and B share node 2. This is represented by the overlapping region, highlighted with a dark grey box in figure 7. Two 10 x 10 3D beam element matrices can be inserted into the top-left and bottom-right rectangular areas of the global matrix. In the overlapping section the terms concerned with node 2 in the matrix for element A are added to the terms concerned with node 2 in the matrix for element B. The white areas in the figure represent blocks of zeros. The concatenation procedure can easily be extended for more elements. The global stiffness matrix will become large relatively quickly, as more elements are required, with the significant section running diagonally from top-left to bottom-right. To be able to determine the displacements for each element of the tool the following equation will be used.

$$KD = R \tag{3}$$

W where K = global stiffness matrix, D = deflections, R = load vector

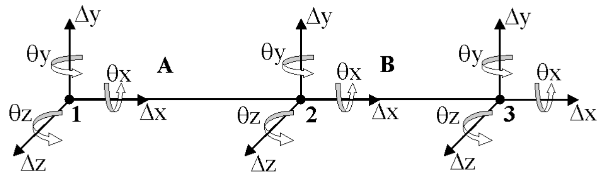

Fig. 6. Beam partitioned into two elements labeled A and B.

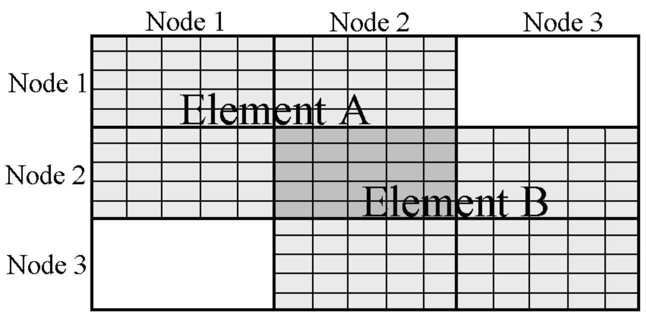

Fig. 7. The concatendation procedure illustrated for two elements.

The load vector, *R*, is a column vector comprising of the force components acting on the tool. How these forces have been determined will be discussed in section 6. The result that is required is the column vector of deflections stored in D. These deflections describe the displacement for every degree of freedom of each element.

The global stiffness matrix can be solved when the boundary conditions are taken into account. These conditions are defined by any known displacements. In this example it has been assumed that the first node of the tool is going to be located close to the handle and will not displace. This allows the first five rows and columns to be removed from the current global matrix to produce the actual global matrix, *K* used in equation (3).

The theory described so far assumes the tool to be aligned with the axes. This is rarely the situation as the user is able to orientate the tool to any orientation they desire as they move the arm of the haptic device. To allow the beam to be arbitrarily orientated local axes can be assigned to each element. The procedure will then be to transform the set of equations from the local coordinates of the beam elements to the global coordinates of the scene in which the forces are represented. The outline for the procedure used is described by Agarwal in [1].

The following equation can be used to obtain the global coordinate stiffness matrix. The element transformation matrix, *T*, is constructed based on the transformation matrix required to transform the elements from their local coordinates into global coordinates.

$$Kg = T^t kmT \tag{4}$$

where *Kg* = global coordinate stiffness matrix, *T* = Element Transformation Matrix, *km* = local coordinate stiffness matrix

Once Kg has been determined this can be concatenated and the boundary conditions can be applied. The concatenated matrix can then be applied to equation (3) to obtain the appropriate deformations for a given load vector, *R*.

To solve equation (3) an LU decomposition method has been employed. The LU decomposition method works reasonably well but the efficiency of the algorithm can be improved if a special property of the matrix, K, is considered. As the values in K fall close to the diagonal an algorithm designed for band diagonal matrices can be incorporated. For a ten element beam 2025 values are required for the standard method. Using the band diagonal matrix method the space requirement is reduced to 855 values and the time is reduced to approximately 10% compared to the standard LU decomposition.

The finite element theory can now be applied to the three dimensional object representing the virtual tool. The remaining stage is to determine the forces that will be exerted on the tool and use these to obtain the deflections of the elements.

6 Applying the Forces to the Elements of the Tool

Sections 3 and 4 detailed the new approach to the haptic rendering of three dimensional tools. The finite element theory described in the previous section can be applied to this tool to model its deformations under specified loads. The first stage is to divide the tool into ten equal sections or elements.

Each element of the tool is considered as a separate object when passed to the SOLID collision detection library. This will allow a separate force to be determined for each of the elements involved. A pair of contact points is determined for each element that intersects an object in the virtual environment. In most cases regarding collisions with a single convex object the contacted elements are limited to one or two. In our approach the first element that is intersected is set as the Contact Element and its contact point is used in the haptic rendering algorithm discussed previously. The Contact Element will be updated if another element in the Visual Tool intersects the object. This follows a strategy similar to the procedure discussed in section 4. All contacted elements can then be used to determine spring forces based on the distances between the contact points located on the elements of both tools. These forces are applied to the load vector, R and used in equation (3). The resultant deflections are then used to deform the tools.

Figures 8 and 9 illustrate two polygonal models during program execution. In both figures the Haptic Interface Tool is represented by the wire frame cuboids whereas the Visual Tool is represented by the filled polygonal object. The force determined is based on a spring force between the contact point on the intersected object, shown as a dark sphere, and the contact point transformed to the position of the Contact Element, shown as the lighter sphere. The highlighted triangle denotes the current contacted component. Figure 8 shows the Visual Tool sliding on the edge of the convex object. Figure 9 displays a convex object consisting of 12672 faces. The program is able to execute at approximately 1000Hz. For larger meshes the collision detection is performed every other iteration.

This work has been implemented on a 2.0GHz dual processor computer and developed using C++ with OpenGL for the graphics rendering. The Ghost SDK supplied with the Phantom device has been utilised for the haptic feedback.

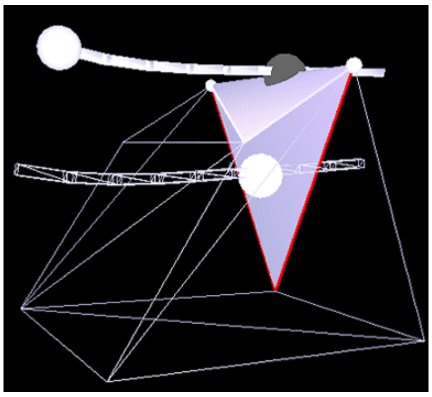

Fig. 7. Tool sliding over an edge. The Haptic Interface Tool is rendered in wireframe. The contact point is indicated by a dark sphere.

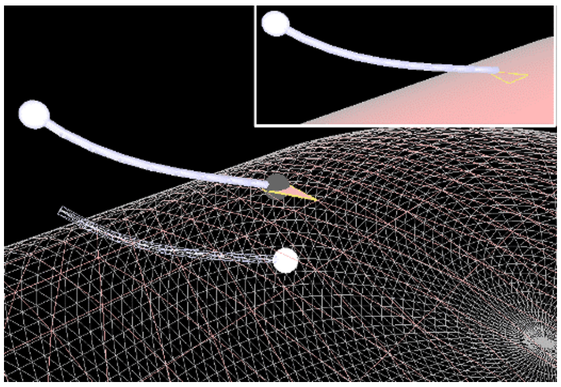

Fig. 7. Tool in collision with a mesh consisting of 12672 faces. Top Right: Object rendered in filled polygon mode and the Haptic Interface Tool is removed.

7 Conclusion

This paper has detailed a new approach to the haptic rendering of a three dimensional object incorporated as the tool in a virtual environment. It has shown that a Visual Tool representation can be included that will remain on the surface of the objects to provide stable haptic feedback without the visual artifacts of the tools penetrating the objects. The collision detection library titled SOLID has been used to determine the collisions between the tool and the objects in the scene. This library is able to determine the contact points when intersection occurs. These points can then be used in the proposed tracking method to constrain the Visual Tool to the surface of the convex polygonal model.

The second stage has shown how the approach can be incorporated with deformable tools to enhance the realism of simulations with virtual tools. The system is able to incorporate a three dimensional tool modelled using a Finite Element Method. The tool is able to deform and collide with polygonal models at haptic frame rates. The overall goal of this research is to develop a framework which will allow a variety of deformable tools to be simulated in the virtual environment.

References

1. R.B. Agarwal, ME273 Lecture Notes, Beam Element, http://www.engr.sjsu.edu/ragarwal/ME165/ME165_Lecture_NOtes_files/FEA_Lectures/Chapter_4_Beam_Element.pdf
2. C. Basdogan, C. Ho and M.A. Srinivasan, Haptic Rendering: Point and Ray Based Interactions, http://citeseer.nj.nec.com/9896.html.
3. B. Baxter, V. Scheib, M.C. Lin and D. Manocha. DAB: Interactive Painting with 3D Virtual Brushes, *Proc. Of ACM SIGGRAPH,* pp. 461-468, 2001.
4. G. van den Bergen. Proximity Queries and Penetration Depth Computation on 3D Game Objects. *Game Developers Conference,* 2001.
5. J. Berkeley, M. Ganter, S. Weghorst, H. Gladstone, G. Raugi and D. Berg, Banded Matrix Approach to Finite Element Modeling for Soft Tissue Simulation, *Virtual Reality: Research, Development and Application* 1999, 4:203-212.
6. J. Berkeley, M. Ganter, S. Weghorst, H. Gladstone, G. Raugi and D. Berg, Real-time Finite Element Modeling with Haptic Support, *Proceedings of the 1999 ASME Design Engineering Technical Conferences,* Sept. 12-15, 1999, Las Vegas.
7. P. Berkelman, R. Hollis and D. Baraff. Interaction with a Realtime Dynamic Environment Simulation using a Magnetic Levitation Haptic Interface Device, *Proc. IEEE International Conference on Robotics and Automation,* pp. 3261-3266, May, 1999, Michigan, United States.
8. R.D. Cook. Finite Element Modeling For Stress Analysis. John Wiley and Sons, Inc, 1995.
9. Ent Surgical Simulator Mid-Term Report, http://www.lockheedmartin.com/akron/busdev/sim&trng/medsim/report.midterm.htm
10. R.T. Fenner, Mechanics of Solids, Blackwell Scientific Publications, 1989
11. T. Gockel, U. Laupp, T. Salb, O. Burgert and R. Dillman. Interactive Simulation of the Teeth Cleaning Process using Volumetric Prototypes, *Medicine Meets Virtual Reality (MMVR)*, January 2002, Newport Beach, USA.
12. M. Koutek and F.H. Post. Spring-Based Manipulation Tools for Virtual Environments, *Proc. Of Immersive Projection Technology and Virtual Environments, 2001,* Springer, Stuttgart, Germany, p. 61-70.
13. S.D. Laycock and A.M. Day, Recent Developments and Applications of Haptic Devices, *to appear in Computer Graphics Forum*
14. S.D. Laycock and A.M. Day. Simulating Deformable Tools with Haptic Feedback in a Virtual Environment, *WCSG Short Papers proceedings,* 2003, pp. 75-81.

15. M.C. Lin and S. Gottschalk, Collision Detection between geometric models: a survey, *In the Proceedings of IMA Conference on Mathematics of Surfaces,* 1998.
16. D. Reznik and C. Laugier. Dynamic Simulation and Virtual Control of a Deformable FIngertip, *Proc. IEEE Int. Conf. on Rob. & Autom. (ICRA),* Minneapolils, MN, April 1996.
17. D.C. Ruspini, K. Kolarov and O. Khatib, The Haptic Display of Complex Graphical Environments, *Computer Graphics Proceedings, Siggraph 1997,* Los Angeles.
18. SensAble Technologies, http://www.sensable.com
19. R. Steffan, T. Kuhlen, and A. Loock. A Virtual Workplace including a Multimodal Human Computer Interface for Interactive Assemble Planning, *IEEE International Conference on Intelligent Engineering Systems (INES '98)*, Vienna.
20. C.B. Zilles, J.K. Salisbury. A constraint based God-Object method for haptic display. *Proc. Of the IEEE Conference on intelligent robots and systems,* 1995, pp146-151.