EG UK Theory and Practice of Computer Graphics (2009)
Wen Tang, John Collomosse (Editors)

A Haptic System for Drilling into Volume Data with Polygonal Tools

Y. Liu S. D. Laycock

School of Computing Sciences, University of East Anglia,
Norwich, NR4 7TJ, UK

Abstract

With the developments of volume visualization technology for complex data sets comes new challenges in terms of user interaction and information extraction. Volume haptics has proven itself to be an effective way of extracting valuable information by providing an extra sense from which to perceive three dimensional data. This paper presents a haptic system for using arbitrary polygonal tools for drilling into volume data. By using this system, users can select from a variety of virtual tools to gain continuous and smooth force feedback during the drilling of volumetric data. As the user manipulates the haptic device the tool typically only moves a small amount. By considering the locations of the data points, that are modified when drilling, a relatively small number of voxels are determined each frame which must be recomputed by a Marching Cubes algorithm.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Virtual reality

1. Introduction

Volume rendering has become widely utilized for applications such as Virtual Training [HBS97], Surgical Planning [QKS01] and Scientific Visualization [IN93]. The ability to visualize volume data directly is particularly important for medical applications where a correct anatomical view of the patient can prove vital for surgical planning. Recent developments in graphics accelerator cards have enabled systems to render large and complex volumetric data sets in a variety of different rendering styles aiding to the observer's perception of the data. Virtual Sculpting systems linked to haptic feedback devices have been available for many years, however, these often do not utilise the original volumetric data whilst sculpting.

In this paper, a novel approach to drilling into objects created from volumetric data is presented. Furthermore, previous systems often are limited to simplistic tools, such as spheres and cylinders. To overcome this limitation this paper presents a drilling approach which permits arbitrary polygonal tools. Figure 1 shows an image of a polygonal tool being used to drill into a three dimensional polygonal object constructed from CT data of a human pelvis. A haptic feedback device is utilised to manipulate the drill in three dimensions. Firstly, the algorithm used to update the volume data must be computationally efficient, especially considering the fact that the surface representation of the volume data may be constructed from millions of triangles. Secondly, the haptic feedback should be rendered such that when the probe point is moving across the voxel boundaries a continuous force is returned to the user. Lastly, since the haptics and visualization calculations will be performed in separate threads, mechanisms are required to ensure that each thread can be updated in a safe manner.

The following list details the key features that have been addressed in order to simulate the interactions that occur when drilling into objects represented by volumetric data.

1. A fast and stable technique.

A local Marching Cubes algorithm is proposed to speed up the process of updating the surface representation as the volume data is modified in real-time. The technique utilises the positions of the data points to reduce the number of voxels that must be updated to ensure the correct surface connec-

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG09/009-016

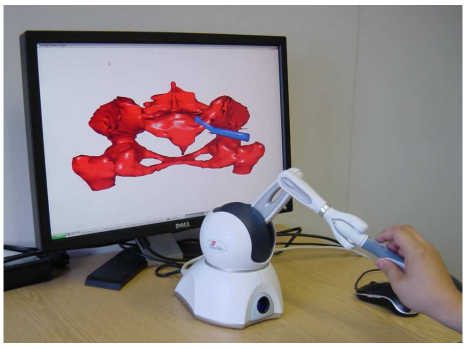


Figure 1: *The visual-haptic system illustrating drilling into a volumetric object constructed from CT data.*

tivity results. A novel Force-Map haptic rendering method is implemented to enable the user to obtain smooth and stable force feedback, this method also satisfies the requirement of achieving a haptic refresh rate of 1000Hz.

2. General purpose.

Integrated with volume visualization, haptic rendering and user interaction, volume haptics can be applied in a variety of domains, including art, industrial design, medical training simulations, and three-dimensional data navigation.

3. Surface characteristics are faithful to the original dataset during drilling.

Utilizing Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) as a basis for volume rendering enables a three dimensional view of patient specific data to be obtained. Enabling the user to interact with the patient data directly is useful in the medical field, particularly since surgeons commonly examine patients through their sense of touch. The algorithm presented in this paper enables the original data to be modified which keeps the surface characteristics faithful to the original data.

4. Fidelity of the simulator.

Many of the drilling simulators are limited to the use of spherical tools which is known to be an over-simplistic representation for real-time collision detection and computation of response force during simulation. This greatly limits the realism of the simulation for the actual drilling task. This paper presents a visual-haptic system to address the limitation of previous systems and introduces a new Force-Map haptic rendering method. This work is capable of incorporating arbitrary polygonal tools which allows different kinds of tools in various shapes and sizes. This is in contrast to most of the previous volume haptic systems which typically only enable a sphere or a cylinder to be used [WZWL03, KHPH05]. The Force-Map haptic rendering method uses a multiple test points algorithm which enhances the fidelity of the response force, and it also allows the user to perceive different forces as the haptic stylus detects different materials represented by volume data.

The remainder of this paper is organized in the following way: Section 2 introduces the related work, Section 3 presents the basic concept of the volumetric data construction, Section 4 presents arbitrary tool modification, Section 5 describes how a Force-Map algorithm for haptic rendering is utilised, Section 6 summarizes the results and the final section provides conclusions and future work.

2. Previous work

Most of the previous work utilises implicit equations to represent the shape of the drilling tool and calculates the force feedback based on the interaction between the object surface and the drilling tool [WZWL03, KHPH05, YH06, MSB*04]. Wang et al. [WZWL03] and Kim et al. [KHPH05] proposed haptic systems that allow probing and cutting of a virtual tooth, but the tool selection is limited to a spherical shape for implementation simplicity. Yau et al. [YH06] presents a dental training haptic simulator utilizing material stiffness and a spring force function. This simulator employs an adaptive octree data structure for the tooth modelling and an oriented bounding box for the cutting tool construction. Standard geometric shapes of the tools, such as a sphere, cylinder and cone are introduced for the cutting task. This work allows different tools to be selected for cutting which enhances the fidelity of the simulator, but still lacks the ability to incorporate irregularly-shaped tools.

Rhienmora et al. [RHD*08] presented a dental training system which employed a variety of virtual tools. The force computation algorithm is extended from a spherical tool to work with a cylindrical tool. They stated that their work has limitations with regard to cutting through different layers of the material which is an important task when developing a bone surgery simulator, for instance. The work presented in this paper allows the drilling tool to be accurately represented by a polygonal object. The drilling approach works in conjunction with the original volume data enabling the effects of drilling through an object's different layers to be realised.

Pflesser et al. [PPUH02] proposed a haptic system for virtual temporal bone surgery which uses a modified version of the Voxmap-Pointshell algorithm [MPT99]. Their approaches sample the surface of the drilling instrument and then generate appropriate forces at each sampled point. A number of samples are distributed around the drill and a ray-tracing approach is then employed to calculate the force vectors towards the tool centre, which can subsequently be combined to generate the overall force returned to the haptic feedback device. The ray tracing algorithm has the potential to miss voxel data located between two rays due to an insufficient sampling.

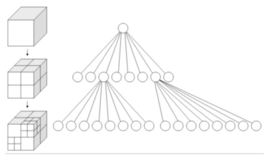
Eriksson et al. [EFW05] proposed a haptic milling surgery simulator using a localized Marching Cubes algorithm for the visualization. To improve the stability they employed a direct haptic rendering method with mechanisms to remove fall-through issues. The data inside the virtual drill is set to a vector pointing to the centre of the voxel. The output force is the sum of all those vectors. This approach works well when the drilling tool moves in a small area, but a "kicking" would result when the haptic test points move across the cubes' boundaries. To overcome the stability issues, a Force-Map haptic rendering algorithm is employed in this work to gain smooth and stable force feedback. Our initial ideas have been represented in [LL09]. Extending from this new local Marching Cubes techniques are presented here for arbitrary tools. Our approach handles the challenging problem of rendering dynamic volume data in real-time and fulfils the requirements for highly interactive haptic applications.

3. Volume data modification

The volume-based representation is a natural choice for rendering a collection of digital images produced by medical scanning technologies such as Magnetic Resonance Imaging (MRI) or Computed Tomography (CT). There are a variety of graphical rendering techniques for visualizing the three dimensional data, often with options to display the material properties such as density and viscosity within the voxels. This has the potential to greatly enhance a user's performance in medical and scientific three dimensional data exploration.

Extracting the global iso-surfaces from the volume data, based on Marching Cubes [LC87], can be time consuming especially when the volume data is derived from many high resolution digital images. However, in this work a local Marching Cubes algorithm and an Octree data structure are employed to enable the surface to be updated efficiently. The values of the volume data surrounding the haptic stylus can be adjusted to less than a surface threshold value depending on the application. By considering the material properties of the data contained within a voxel the rate at which the data is removed can be adjusted. Once the data has been updated the local Marching Cubes approach recomputes the surface surrounding the stylus. The volume that is updated depends on the resolution of the volume data and the shape of the tool used for the interaction. Section 4 discusses an approach to limiting the voxels that must be updated as the data is modified.

When using the Marching Cubes algorithm [LC87], a volume can be interpreted by generating polygons representing the surface, typically constrained to a specified value of the data. This algorithm proceeds firstly by either loading a series of image slices or by procedurally constructing a data set. A suitable threshold value which determines the surface location is chosen for the construction of the surface. After calculating the area occupied by the volume data, the system will use a collection of smaller cubes to cover the entire volume. Consequently, the system constructs a surface using the Marching Cubes algorithm by checking eight corners of each cube. The value of each corner is calculated based on the surrounding eight volume data values using tri-linear interpolation. The surface configuration of each cube is then described by 15 basic surface patterns, which are subsequently matched to the volume data to provide a set of triangles for each voxel. The location of the triangle's vertices on the edges of the cube will be determined by the chosen threshold value.


(a)

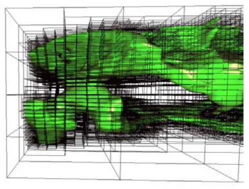

(b)

Figure 2: *(a) Octree data structure, (b) Pelvis data construction using Octree data structure.*

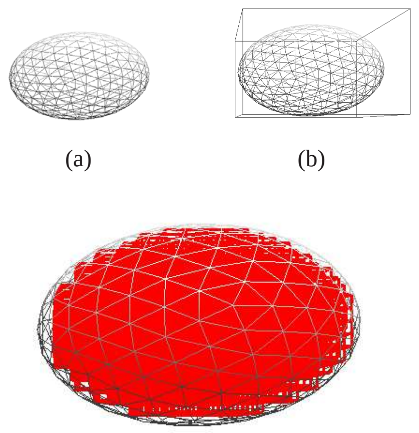
To handle large data sets, an Octree-based structure [FDFH96] is employed which enables the data to be changed dynamically in an efficient manner. The Octree-based structure uses a hierarchical representation of the data to efficiently detect and update localized changes to the data [EDW06]. Each node in the octree represents a cell which contains triangles. Initially paths in the octree from the root to a leaf (voxel) will only be created if triangles forming the surface reside in the voxel, Figure 2. If the haptic stylus reaches a region and edits the data where no surface triangles are present then a new surface is likely to result. At this point the octree is updated by traversing from the root to the leaf containing the modified data, creating any new cells for the octree that do not previously exist. If the data changes such that an octree cell no longer contains triangles on the surface, then the triangles and octree cells are removed from the structure.

Naturally the efficiency of the approach is affected by the chosen depth of the octree. There is a trade-off between the quality of the visualization and the efficiency of the approach. If a small octree depth is used fewer voxels containing large triangles will result, which can often exhibit undesirable edge aliasing. Conversely, too many voxels caused by higher octree depths will increase the computational load of updating the surface during tool-object intersection. The Octree depth selection also depends on the size of the volume data. If the grid is too small then the visualization is more complex when dealing with a huge number or a large area of volume data.

4. Tool Modelling and tool-object interaction

The system is able to function with an arbitrarily shaped drilling tool composed of polygons. This extension strives further than other work which only employs simplistic objects, such as single spheres or cylinders as the drilling tools represented by implicit functions.

Initially, a bounding box is constructed for each polygonal drilling tool which contains the whole shape of the tool. This is subdivided into eight equally sized children. Each child is recursively split into eight further children, as long as a pre-chosen depth, d, is not reached. The leaf nodes of the octree are then used to initialise a three dimensional grid of cells encompassing the entire object. The leaf nodes define the cells containing the surface polygons representing the tool. Finally, each remaining cell in the three dimensional grid must be classified as one of two types: empty nodes inside the tool and empty nodes outside the tool. A flood fill algorithm can be used to determine the cells that are inside the virtual tool. This method starts by choosing a cell known to be inside the tool object. Subsequently, it iteratively checks the 26 surrounding boxes until the boundary ones are reached. The approach results in all the interior boxes being labelled as interior. Figure 3 shows the three steps for voxelising the internal volume of an arbitrary polygonal tool. The scale of the tool may also be easily adjusted to satisfy the specific requirements of a given application. Through experiment we select the depth of the octree to obtain the desired voxel size.


(a) (b)

(c)

Figure 3: *Tool Preprocessing: (a) Original Polygonal tool, (b) Bounding box surrounding the tool, (c) Identification of internal boxes via flood fill.*

During the running of the program the polygonal tool interacts with the object derived from the volume data. To be able to effectively modify the data whilst drilling the volume, data points within the tool's bounding box are tested to determine if they are inside the tool's volume. Firstly, each data point must be checked with the three dimensional grid of cells to detect if the point is either in a boundary or interior cell. If a data point is located in a boundary cell, then it will be further checked against the tool's surface triangles located in the cell. After these steps, the values of all the data points inside the tool will be modified. After the data has been changed, the bounding box volume around the modified data points can be utilised to perform a local Marching Cubes algorithm to generate a new surface from the modified volume data.

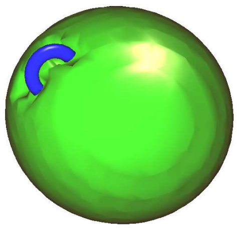


Figure 4: *Polygonal tool and object interaction.*

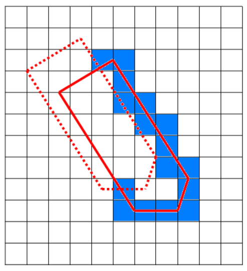


Figure 5: *Voxels need to be updated between two adjacent graphic frames when drilling. The dotted outline represents the drill tool at the previous frame, whilst the solid outline represents the drill at the current frame. The blue boxes represent areas that need to be calculated by the Marching Cubes algorithm.*

The efficiency of the method discussed above largely depends on the size of the tool. The larger the tool used to interact with the data, the more voxels that need to be updated and recalculated by the Marching Cubes approach. This limits the use of the complicated tool implementation. Since the haptic stylus moves slowly during drilling, especially when the tool interacts with rigid objects such as bones, due to the strong force feedback. The volume of data that must be changed between the adjacent graphic frames may differ by only a small amount, or indeed maybe exactly the same when the drilling tool does not move across a small voxel.

Thus, it is not necessary to update the whole bounding box in each graphic frame because of the largely overlapping area. Alternatively, the update step can only consider the new area compared to the data area in the previous frame, as shown in Figure 5, which avoids calculating the overlapping voxels twice in two frames. By using this method, the computation of the tool-object interaction is dramatically improved even when dealing with large polygonal tools. In detail, first of all, the changed data is detected for later use. Then the voxels containing that changed data are chosen to regenerate the new surface, as shown in Figure 5.

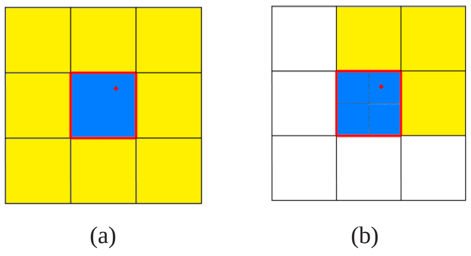


Figure 6: *To maintain the polygonal surface during drilling some of the neighbouring voxels must be updated. (a) Illustrates all the connected neighbours (yellow), whilst (b) represents only those boxes (yellow) that must be updated given the location of the data point that was modified.*

To ensure mesh connectivity and correct illumination of the surface, voxels surrounding the ones containing modified data points may also need to be recalculated. In terms of the illumination, vertex normals in neighbouring voxels may need to be recalculated since shared vertices may have changed. To solve this problem, each voxel contains pointers to its neighbours to enable an efficient recalculation of the vertex normals for illumination.

The mesh connectivity is more complex, since if data points change close to the boundary of a voxel they may require Marching Cubes to be run once again on their neighbours. This could result in a large number of voxels, with a potential of becoming worse if a higher resolution voxel grid is employed resulting in more voxels needing recalculation every frame. In order to acheive the requirement of the haptic system, this visual-haptic system utilises the positions of the data points to reduce the number of voxels that must be updated to ensure correct surface connectivity. Figure 6(a) illustrates one of the blue boxes, identified in Figure 5, which contains a single red data point which has been updated. Ignoring the location of this data point, the algorithm would need to recompute Marching Cubes on the neighbours, indicated in yellow. Instead of this approach, each voxel has been separated into eight data occupation areas to help in reducing the number of neighbouring voxels. If the data is changed when drilling, the data points affected will only cause surface changes in a local proximity and therefore the voxels near the data occupation area must be updated. As shown in Figure 6(b), if the modified data (red point) is in the top right corner of the voxel (blue), then it only calculates the voxels (yellow) near the top right corner.

5. Haptic rendering

In Section 2, the Direct Volume Haptic rendering approach was described. The aim of the approach is to generate force feedback directly from the volume data without extracting an intermediate representation. The Force-Map algorithm, used in this system, is a direct haptic rendering method. This work initially employs a haptic feedback approach similar to [PPUH02] which also represents the drill as a number of sample points, distributed approximately around the surface of the tool. However, in this work the sample points are tested for contact with the volume data. Given a sample point position, a vector calculated from the occupancy force-map can be output. By using this method, the force feedback is stable and smooth even though it has a similar force cube encoding system. The force vectors stored in the data are calculated based on the local surface, which also benefits from the advantages of the surface-based haptic rendering approach. The synchronisation of updating the graphic and haptic loops enhances the fidelity of the virtual visual-haptic system when applied to real applications. The three steps below outline the Force-Map haptic rendering method adopted for a surface representation of dynamically changing voxel data.

1. Initialisation and construction

Initially all the normals of the triangles contained in each octree leaf node (voxel) are averaged to result in a single force vector representing the data in the voxel. The larger the voxel is, the more volume data points lie within it. Additionally, only the data inside the voxel is assigned to a force vector while others are set to none. After this initialisation step, all the data near to the surface is set to a force vector which approximately equals the closest surface normal.

2. Upgrade and reconstruction

When the surface is updated in the haptics thread the data points that are found to lie inside the new voxel are set to a force value based on the triangle's face normal. If there is more than one triangle in the voxel, the averaged face normal will be used. Some force values in the old surface might also need to be updated since the triangles forming the surface in the voxel may have changed.

3. Calculating the force feedback

The force vectors stored in the data must be combined appropriately before being returned to the haptic device. When the virtual drilling tool moves into the volume data, a haptic test point checks the surrounding eight data values in the three dimensional space. These eight data values are referred to as the force cube in this work. The corners of the force cube contain the force vectors stored in the data. Tri-linear interpolation is employed here to enable an interpolated force vector to be calculated for any position inside the force cube. Another advantage of using the tri-linear interpolation method is that the haptic test point can be smoothly moved from one force cube to another without any force discontinuities occurring between them. Therefore, all these force cubes together form a volume in which continuous haptic feedback can be generated as the tool moves.

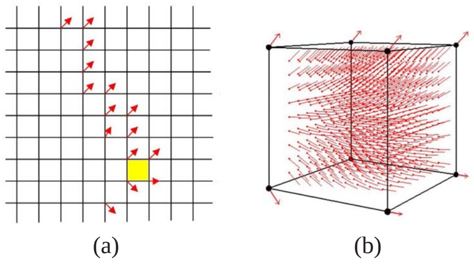


Figure 7: *Force-Map haptic rendering, the red arrows represent the force vector. (a) The yellow square indicates one force cube displayed in two dimensions. (b) The same single force cube in three dimensions.*

5.1. Multi-Point implementation

Multi-point haptic rendering algorithms are capable of enhancing the fidelity of the application. Section 3 stated the fact that the data can be changed based on the shape of the drilling tool. After the new surface has been generated the haptic rendering algorithm should return the corresponding force feedback. In order to achieve this target, a number of haptic points are distributed approximately around the surface of the drilling tool. At each time step, each haptic point is tested in the constructed Force-Map to calculate the contribution to the overall haptic force.

For any real application, drilling with a single point does not lead to a realistic result. The approach involving multiple test points approximating the drilling tool is usually preferred. Section 4 stated the fact that the data can be changed based on the shape of the drilling tool. After the new surface has been generated the haptic rendering algorithm should return the corresponding force feedback. In order to achieve this target, a number of haptic points are distributed approximately around the surface of the drilling tool. These are equal to the vertices of the haptic tool. At each time step, each haptic point is tested in the constructed Force-Map to calculate the contribution to the overall haptic force.

5.2. Multi-Layer implementation

In many applications, the properties of the simulated materials differ depending on the location being drilled. This is particularly the case in medical and dental applications where the material properties of each voxel must be considered. For example, drilling through soft tissue should be very different to drilling through rigid bone. The Force-Map method can easily incorporate this issue by simply setting a scaled force vector where the scaling factor is related to the neighbouring voxel data. Specifically, in the second step of the Force-Map method, the data values are set to an averaged face normal of all the triangles contained in a given cube. If a multi-layer implementation is required in the haptic system, the data can be set to a scale of that averaged face normal. The sum of the collision forces are applied to the dynamic object and numerically integrated to obtain a force vector. This vector is the force sent to the haptic device. In terms of the graphic rendering, a colour value is set to the corners of the cube based on the material properties of the data. These colour values can then be interpolated when the surface is recalculated to ensure the modified triangles are coloured appropriately.

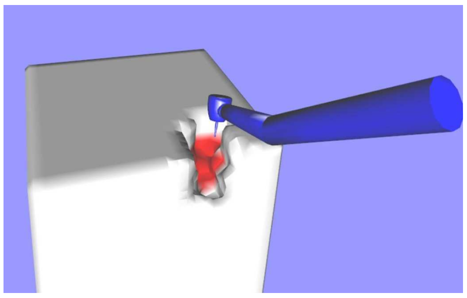


Figure 8: *Multi-Layer implementation*

Figure 8 shows the virtual haptic tool drilling across two layers. The neighbouring layers are assigned different colours to illustrate the change in force feedback that will be perceived by the user.

6. Results

Figure 9 illustrates a procedurally generated sphere along side a surface representation of a human pelvis. The surface was extracted from 87 CT slices obtained at the Norfolk and Norwich University Hospital, UK.

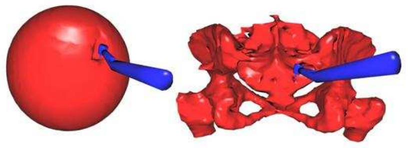


Figure 9: *The left sphere-like object is created procedurally whilst the right hand image was extracted from 87 CT image slices. Each slice contains 256 X 256 pixels.*

The work has been tested on a Quad Core 2.4GHz processor PC with a NVIDIA Geforce 8800GTX graphics card. To provide haptic feedback a PHANToM Omni device, produced by SensAble Technologies has been employed. By using the system, a user can drill into rigid objects using arbitrary types of tools constructed from polygons.

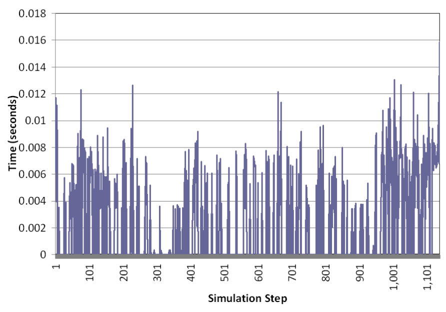


Figure 10: *A graph presenting the time taken to update the surface during drilling with a polygonal tool(octree depth: 4).*

The volume on Figure 4 has been calculated from a data set comprising of 100 x 100 x 100 data points. This data has been sampled to create a triangular surface mesh. Figure 10 shows the time required to perform the surface modification and Force-Map updates during rendering, which allows users to efficiently obtain visual cues. The Force-Map can be sampled at a higher rate in the haptic feedback loop to obtain stable force feedback. Figure 11shows that if the octree depth is five, the display has higher resolution but this increases the update time. The spikes occurring in Figure 10 and Figure 11 are due to the device moving faster and covering more voxels between updates.

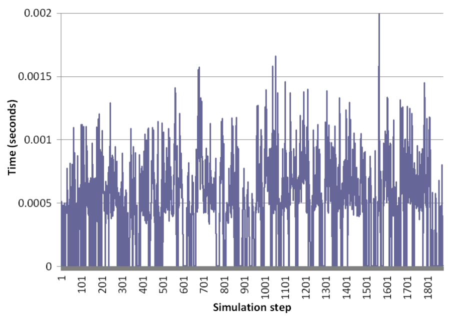


Figure 11: *A graph presenting the time taken to update the surface during drilling with a polygonal tool.(octree depth: 5).*

7. Conclusions

In this paper, a visual-haptic system which allows arbitrary polygonal tools to be utilised when drilling into objects, based on volume data, is presented. By utilising the locations of the data points being modified a small number of voxels can be computed to enable the connectivity of the surface to be maintained during drilling. The Force-Map approach samples this surface to compute a continuous force field which can, in turn, be sampled in the haptics thread to acheive smooth force feedback at voxel boundaries.

Since the higher resolution of volume data affects the speed of the surface update, future work will improve the efficiency in this respect. Further improvement of the haptic rendering will focus on generating tangential forces which is an important property of the drilling application. Morris et al. [EDT*06] employed a method to calculate the tangential forces in their visual-haptic system. They approximate the local surface normal by calculating a vector from the voxel data pointing to the drilling axis, but this approximation does not meet the requirement of obtaining a precise drilling force. The future work will implement the tangential force by using the Force-Map haptic rendering algorithm.

References

[EDT*06] E E. R., D D. M., T T. E., Barbagli F., Pai D. K., Superiore S., Anna S.: Standardized evaluation of haptic rendering systems. *Proceedings of the 14th IEEE Haptics Symposium* (2006).

[EDW06] Eriksson M., Dixon M., Wikander J.: A haptic vr milling surgery simulator using high resolution ct data. *Proceedings of Medicine Meets Virtual Reality 14* (2006), 138Ű144.

[EFW05] Eriksson M., Flemmer H., Wikander J.: A haptic and virtual reality skull bone surgery simulator. *Proceedings of World Haptics* (March 2005).

[FDFH96] Foley J. D., Dam A. V., Feiner S. K., Hughes J. F.: *Computer Graphics: Principle and practice*. Addison-Wesley, 1996.

[HBS97] Ho C., Basdogan C., Srinivasan M.: Haptic rendering: Point and ray based interactions. In *Proc. of the Second PHANToM Users Group Workshop* (1997).

[IN93] Iwata H., Noma H.: Volume haptization. In *Proc. of IEEE Symp. on Research Frontiers in Virtual Reality* (1993), pp. 16–23.

[KHPH05] Kim L., Hwang Y., Park S. H., Ha S.: Dental training system using multi-modal interface. *CAD & A 2*, 5 (2005), 591–598.

[LC87] Lorenson W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH* (1987), pp. 163–169.

[LL09] LIU Y., LAYCOCK S.: The force-map haptic rendering algorithm for drilling into volume data. In *WSCG 09* (2009).

[MPT99] MCNEELY W. A., PUTERBAUGH K. D., TROY J. J.: Six degree-of-freedom haptic rendering using voxel sampling. In *ACM SIGGRAPH* (1999), pp. 401–408.

[MSB*04] MORRIS D., SEWELL C., BLEVINS N., BARBAGLI F., SALISBURY K.: A collaborative virtual environment for the simulation of temporal bone surgery. In *In MICCAI* (2004), pp. 319–327.

[PPUH02] PETERSIK A., PFLESSER B., U.TIEDE, HÖHNE K. H.: Haptic rendering of volumetric anatomic models at sub-voxel resolution. In *10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (2002), p. 66.

[QKS01] QINGSONG Z., KEONG K. C., SING N. W.: Interactive surgical planning using context based volume visualization techniques. In *MIAR '01* (2001), p. 21.

[RHD*08] RHIENMORA P., HADDAWY P., DAILEY M. N., KHANAL P., SUEBNUKARN S.: Development of a dental skills training simulator using virtual reality and haptic device. *NECTEC* (2008), 140–147.

[WZWL03] WANG D., ZHANG Y., WANG Y., LU P.: Development of dental training system with haptic display. *The 12th IEEE International Workshop on Robot and Human Interactive Com.* (Nov. 2003), 159–164.

[YH06] YAU H. T., HSU C. Y.: Development of a dental training system based on point-based models. *Computer-Aided Design & Applications. 3*, 6 (2006), 591–598.