SpinalTap: An Architecture For Real-Time Vertebrae Drilling Simulation

Ryan Schmidt

12th April 2002

Abstract

This report describes the implementation of the SpinalTap virtual spinal surgery simulation system. The goal of the system is to simulate drilling into vertebrae. Two methods for reconstructing a vertebrae from CT scan data are presented. Several models of an L3 vertebrae are reconstructed, with varying results. An architecture for real-time high speed decimation of point volumes is explained and implemented. Simulations performed with test data show collision testing rates of over 400Hz on current PC hardware.

1 Introduction

Traditionally, training surgeons has been a difficult task. Before a surgeon can be trusted to perform procedures on a live human, he or she must show proficiency in training situations. Currently several methods of surgical training exist. Observation of an actual surgery will teach a new surgeon the steps involved in the procedure. This method does not provide the 'hands-on' experience critical to successful operations - practice is required. Working on human cadavers is effective, but is rather expensive. Artificial body parts are often used for training are not very realistic. Finally, training on sedated live animals raises animal rights concerns.

The goal of this project is to simulate spinal surgery procedures that involve drilling into the L3 vertebrae. During the operation a surgeon drills several holes into the vertebrae. Pedicle screws are inserted into these holes and used to stabilize the spine. Many novice surgeons are unfamiliar with the use of tools as powerful as a motorized spinal drill. The effects of errors in this procedure are immediate and irreparable, making the operation very dangerous. In addition, osteoperotic bone can be very brittle in some places and it is very easy to drill too far. A surgeon must be prepared for this, unfortunately experience is difficult to obtain without risk to the patient. This makes spinal drilling a good candidate for training in a virtual environment.

A virtual model of the spinal vertebrae will be constructed from CT data. Surface points are extracted from the CT images. A Radial Basis Function is fit to the these points, providing a smooth polygonizable surface with an accurate inside-outside test. Cylindrical point volumes are generated along pre-planned drill paths, then decimated during simulation to provide density feedback.

A critical piece of the simulation environment is the haptic-feedback drill. This drill gives the surgeon physical feedback similar to what he or she would feel in a real procedure. The design of this haptic drill is being undertaken by Marilyn Powers and is beyond the scope of this project.

A key problem is decimation of the vertebrae model at interactive rates. The physical feedback system requires an update frequency of 300-1000Hz to be convey realistic information about the rigid vertebrae surface. The visualization system must update at least 10 times a second for the user to consider the simulation interactive. These requirements limit the accuracy of the collision detection system.

The problems focused on have been reconstructing a virtual vertebrae model from CT data and real-time decimation of the point volume. An interactive surgery simulator has not been developed, instead simulations of drill movement have been created to validate the performance of the collision detection algorithm.

2 Previous Work

Real-time Surgery simulation systems have been presented by [26] and [14]. However, these systems are not amenable to being modified for bone surgery simulation. [26] simulated eye surgery as part of a larger tele-operated microsurgery robotic system. The system relied on non-linear finite element methods (FEM) to simulate deformation and cutting of corneal tissue. A force-feedback physical interface was used for manipulating surgical tools. This system included motion damping, allowing the surgeon to operate with increased precision. Visualization was done using implicit primitives under free-form deformation, where the deformation was based on a reduced representation of the finite element mesh. Results were rendered on an HMD or large back-projected display at an update frequency of 10Hz using high-end Silicon Graphics workstations.

[14] implemented a system for arthroscopic knee surgery simulation. However, the system lacked collision response mechanisms and did not allow for any modification of the bone structure. Visualization was accomplished using real-time volume rendering techniques on high-end graphics workstations. A SensAble Technologies PHANToM was used to provide user input and haptic feedback. A system that simulated the effects of cranial surgery on facial appearance was presented in [20]. Unfortunately this was not a real-time system. The system relied on spring FEM to attach a skin mesh to the skull structure.

Several other researchers have explored methods of simulating human physiology. Muscle deformations were explored in [7] and [17]. These systems simulated deformation effects based on FEM. [9] achieved FEM-based deformation of soft tissue in real-time using a linear elastic model with quasi-nonlinear elasticity. A force-feedback system was included, and rendering was done based on a deformed polygon mesh. However, the real-time performance of the system relied heavily on precomputation, which ruled out the ability to dynamically modify the structure of the finite element mesh. [11] also presented a system for real-time deformation which relied on preprocessing and did not allow modification of the element structure.

The systems presented so far have mainly been concerned with simulating deformation of non-rigid bodies. To this end, they have all relied on some type of finite element analysis to determine deformation. To achieve real-time performance they have relied on heavy preprocessing or supercomputer-class hardware resources. The systems that rely on precomputation have been unable to simulate any of the fracture scenarios presented in [16]. These simulations are unworkable for bone surgery, which requires modification of the underlying structure. However, FEM-based non-rigid body deformation is not necessary as bone is a relatively rigid structure. In addition, the only system presented so far which included an immersive simulation environment is [26].

3 A Virtual Vertebrae Model

A critical component of SpinalTap is the virtual vertebrae model. The efficiency of the collision detection system relies heavily on the vertebrae data structures. The model must meet these 4 primary requirements:

1. Support an accurate inside-outside test.
2. Support an accurate polygonization method.
3. Allow for fast rendering of decimated areas.
4. Be based on CT scan data.

The model must support an accurate inside-outside test in order to generate points inside the surface. These points are critical to the collision detection algorithm described in section 7. Achieving real-time animation framerates implies that the vertebrae surface must be rendered as a polygonal mesh. To guarantee spatial consistency, it is essential that this mesh be generated from the same data used to determine density values and collision points. Hence, the model must support a fast and accurate polygonization algorithm. Techniques must also be devised to permit fast viewing of the model during drilling.

The final restriction is the most problematic. The virtual vertebrae surface and density field are to be based on Computed Tomography (CT) data. A stack of CT slices of the L3 vertebrae will be the basis for the vertebrae model. Building the virtual vertebrae from this standard medical data source allows the system to work with any particular vertebrae and be tailored to individual patients.

3.1 Computed Tomography Data

3.1.1 Computed Tomography

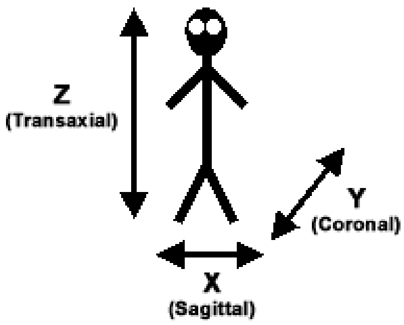


Figure 1: CT scan axis (Sagittal, Coronal, Transaxial).

Computed Tomography imaging, commonly known as CT or CAT (Computed Axial Tomography) scanning, creates greyscale images similar to x-rays. An x-ray beam and detector are swept 360 degrees around the object in question, creating a *slice* image that represents a thin planar section of the object. A set of these slices is called a *stack* or *volume*. This stack of images represents a volume in space and can be directly visualized with volume rendering tools [12] [10].

CT volumes are distributed as stacks of images along one of three primary axis. These axis are form a right-handed coordinate system and have a standard orientation in relation to patient position during scanning (see figure 1). The *Transaxial* axis runs from head to toe, with the positive direction being towards the head. The *Sagittal* axis is positive towards the patient's left hand. Finally, the *Coronal* axis is positive towards the patient's posterior. This project considers the sagittal, coronal, and transaxial axis to be the X, Y, and Z axis, respectively

A standard stack file format is defined as part of the Digital Imaging and Communications in Medicine (DICOM) specification. Many DICOM tools allow for re-slicing of a CT volume along any of the primary axis. This re-slicing generates a new stack of images.

3.1.2 L3 Vertebrae CT Stack

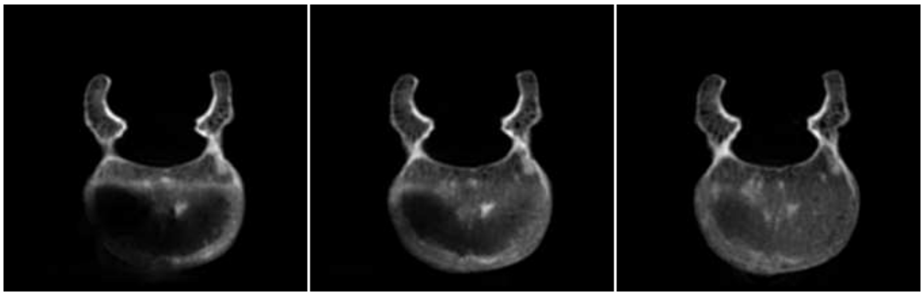


Figure 2: L3 vertebrae CT scan stack, slices 20-22.

A transaxial DICOM stack of 95 512x512 pixel images of the L3 vertebrae has been provided by the Laboratory of Human Anatomy and Embryology at the University of Brussels, Belgium. These images represent 180x180mm slices. The actual vertebrae volume covers at most 224x212 pixels on any slice, corresponding to approximately 79x75mm segments of the vertebrae. The excess image area has been removed and a few slices can be seen in figure 2. This implies that the maximum volume of the vertebrae is approximately 560,000mm^3. The maximum diagonal distance across a slice is about 109mm, implying that a drill bit will never intersect the vertebrae over more than 109mm of it's surface.

Actual vertebrae volume takes up a relatively small portion of the stack volume. Even after the excess image area has been cut down from 512x512 pixels to 224x212 pixels, over 80% of the volume is empty.

Several attributes are needed to map CT slice pixels into actual world coordinates. These are the X and Y pixel spacings, and slice spacing. X and Y pixel spacings are measured between the centres of two adjacent pixels. Slice spacing is measured between the parallel centre lines of two adjacent slices and represents the distance the CT scanner is moved between slices (see figure 3). Any point in the CT volume can be mapped to and from world coordinates based on these spacings.

The L3 stack in use is a transaxial stack. The X and Y pixel spacings for the stack (corresponding to the sagittal and coronal axis) are both 0.352mm. The slice spacing in the transaxial axis is 0.5mm. Hence, pixel resolution in the Z direction is significantly lower than in the XY plane. This is apparent when texture-mapping 3D planes with sagittal and coronal slices generated from re-slicing of the transaxial volume. These images appear to be stretched in the Z axis due to the lower Z resolution.

One other attribute available for the CT volume is the slice thickness. The slice thickness for the L3 transaxial stack is 1.1mm. This measure implies that each CT

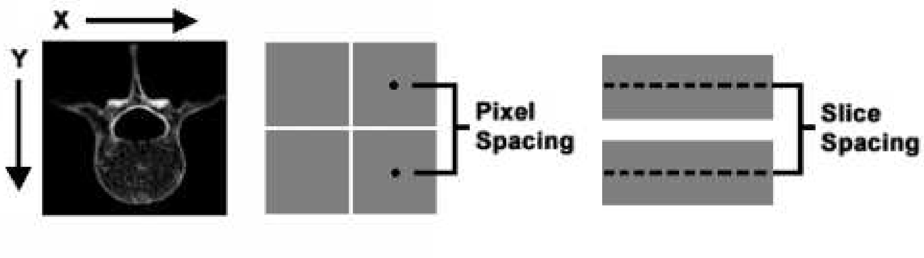


Figure 3: CT slice coordinate system, pixel spacing, and slice spacing.

slice represents a 1.1mm thick slice of matter. The slice thickness for the L3 volume is larger than twice the slice spacing. Hence, each slice slightly overlaps the slice above and below it. These noise due to these overlaps complicates the segmentation process described in section 3.2.3.

3.2 Surface Reconstruction

3.2.1 Previous Work

Many techniques have been applied to the surface reconstruction problem. Perhaps the simplest method is to apply a polygnization algorithm to the volumetric data [23]. The surfaces produced using this method are generally not smooth. Another option is to create a stack of 3D contours and stitch them together (REFS). This technique can produce relatively smooth surfaces. Unfortunately, stitching contours to produce a surface of arbitrary topology is very difficult. Several researchers have attempted to fit surfaces directly to a set of surface points. Early methods involved orienting planes to create signed distance functions, with reasonable results [18]. An enhancement to this technique invovled fitting Bernstein-Bézier patches to the signed distance function, achieving C^1 continuity [1].

While effective at constructing a polygonal surface, each of these techniques fails to meet one of the requirements of the vertebrae model. Namely, none provide a fast inside/outside point test. These methods all fit polygons or parametric surfaces to the CT data. However, recent research in radial basis functions [5] [6] [28] [3] [4] has provided a powerful method for reconstructing 3D models from a set of surface points. Fitting a radial basis function (RBF) to this set of surface points results in an implicit function - the simplest sort of inside/outside test. Any implicit surface polygonizer [29] can be applied tothe RBF to produce a polygonal mesh. The process of fitting and evaluating a radial basis function is described in detail in [5].

3.2.2 FastRBF

FastRBF is a commercial surface fitting tool that employs radial basis functions. The software is distributed by FarField Technology and a 30,000 point time-limited license is available for trial use. FastRBF includes an implicit polygonizer that produces opti-

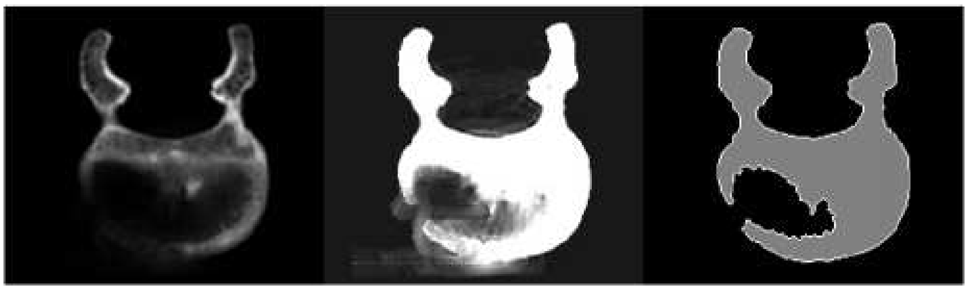


Figure 4: A problematic CT slice. On the left is the original CT image. Contrast and brightness have been increased to highlight ambiguous pixels. The resulting segmented slice is shown on the right.

mized triangle meshes. FarField Technology has implemented fast multipole methods, as well as the centre reduction and smoothing algorithms described in [5].

The final RBF defining the vertebrae surface has approximately 14,000 centres. However, the the largest initial set of centres had over 130,000 points, well over the 30,000 point limit allowed by the FastRBF trial license. This prompted a new method of fitting to be applied. The list of surface points was chopped into groups, based on z height. A separate RBF was fit to each of these groups with reduction applied. This reduced the total centre count to well under 30,000. A final RBF was then fit to these centres with smoothing applied.

There is some question as to how much accuracy is lost by fitting portions of the final surface separately. The real area of concern lies near the edges of the sub-surfaces. There is a possibility that the centre reduction algorithm would discard points that are unnecessary for the sub-surface, but would not be discarded if the entire RBF could be fit directly.

One option considered to decrease the likelihood of unwanted reduction was to fit an RBF to each slice, along with a range of adjacent slices on either side. The reduced set of centres for the slice would then be extracted. Reduced centres at the edge of any sub-surface are ignored, minimizing loss of important centres. However, in practice this technique was unnecessary. Reduction did not discard enough centres to seriously affect surface topology. However, RBF smoothing did affect surface topology, see section 3.3.5.

3.2.3 Segmenting CT Slices

Radial basis functions fit a smooth surface to a set of surface points and off-surface points. To fit an RBF to CT slices the surface points must be generated from greyscale image pixels. Determining surface points is an edge-finding problem. The set of surface pixels (and off surface points generated from surface pixels) defines the surface in 3D space.

Existing methods have been devised to automatically extract surface contours out of CT and MRI data [13] [19]. Surface points are then generated along these smooth contours. These techniques are generally used as an initial step and require verification by a trained professional. For the purposes of this project, only one CT stack needs to be segmented. The segmentation was done by hand, using the magic wand tool in the GNU Image Manipulation Program (GIMP). Manual segmentation is a time-consuming process.

The GIMP magic wand tool is a simple threshold-selection tool that can be used to find edges based on pixel intensity. This tool worked moderately well for many slices. Unfortunately, significant noise is present in most slices. These noisy pixels are visible in middle picture in figure 4. The dark U-shaped area in this slice is at the bottom of an indentation in the vertebrae surface, where high curvature exists in the plane of the slice.

Many pixels that are obviously outside of the surface have a higher intensity than pixels inside the visible boundary. The intensity-based magic wand tool selects these noisy pixels before selecting the inner pixels. To avoid selecting exterior pixels the user must manually manipulate the magic wand threshold. The danger here is that on adjacent slices slightly different thresholds may be necessary to avoid picking noisy pixels. This reduces the amount of coherence between the surface pixels of adjacent slices. The surface can 'wiggle' back and forth a pixel or two. To avoid this situation pixels deemed exterior to the surface needed to be manually deselected.

3.3 Pixel-Based Surface Reconstruction Method

3.3.1 Point Generation

Surface and off-surface points (OSPs) must be extracted from the CT slice images in order to fit an RBF surface. Surface points can be generated by isolating the surface pixels as described in section 3.2.3 and placing a point at the center of each pixel. The points can then be transformed into world space using the algorithm described in section 3.1.2. The resulting set of surface points is shown in figure 5.

Generation of OSPs is a more daunting task. As described in [5], the placement off OSPs is an important determinant of final surface smoothness. The first attempt at OSP generation was a pixel-based method. For each surface pixel, adjacent North, East, South, and West pixels were examined. If the surrounding pixel was inside the surface (in the solid grey areas of figure 4) an 'inside' OSP was produced. 'Outside' OSPs were generated for pixels determined to be outside the surface. An RBF was fit to the final set of points using the FastRBF software and methods described in section 3.2.2.

3.3.2 Initial Results

The initial manual segmentation of the CT slice images resulted in 44,327 surface points and 88,123 off-surface points. RBF's were iteratively fit to these 132,450 points. After reduction, the final point set numbered just over 28,000. The inital surface was fit with no smoothing (left image of figure 6). This surface is far too rough to be used in the simulation environment.

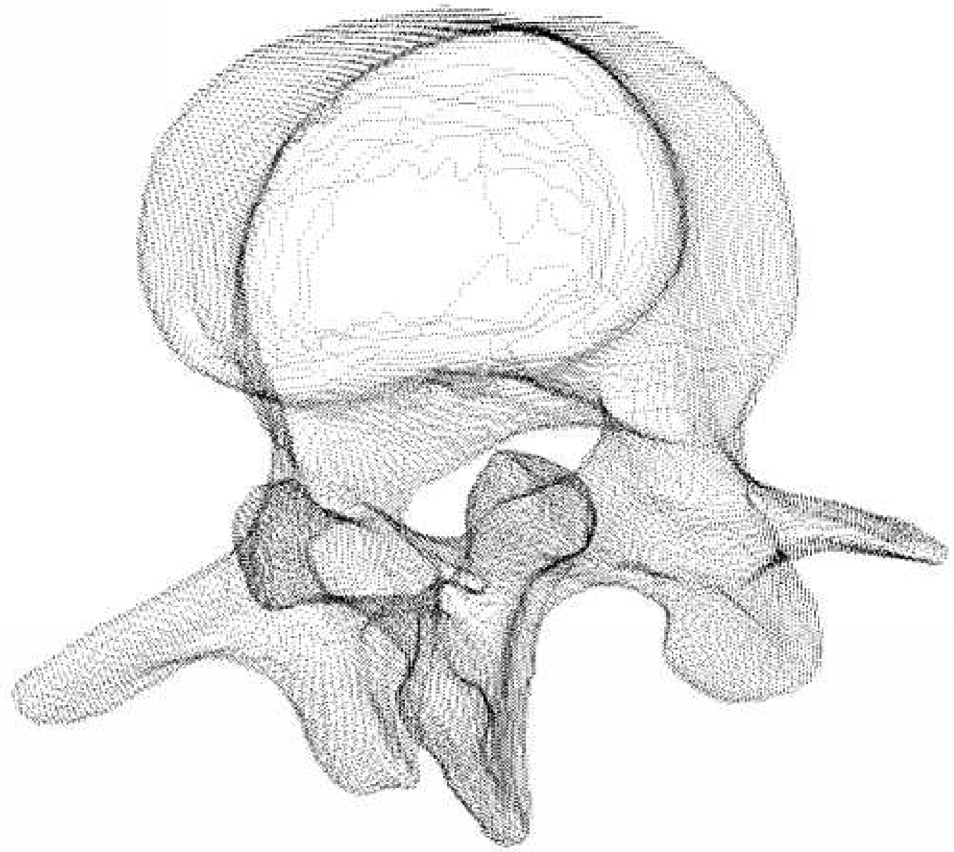


Figure 5: A Radial Basis Function is fit to this set of surface points to create a virtual vertebrae surface. The RBF automatically interpolates through the areas with low point density.

Figure 6: Results of pixel-based OSP generation. The left image has no smoothing. The middle image has smoothing factor 0.5, and the right image has smoothing factor 30.

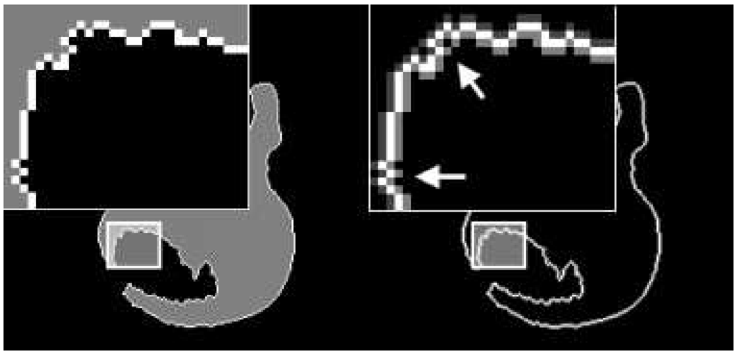


Figure 7: Irregular segmentation provided by the GIMP magic wand tool. On the left is the segmented surface. Off-surface pixels are shown on the right.

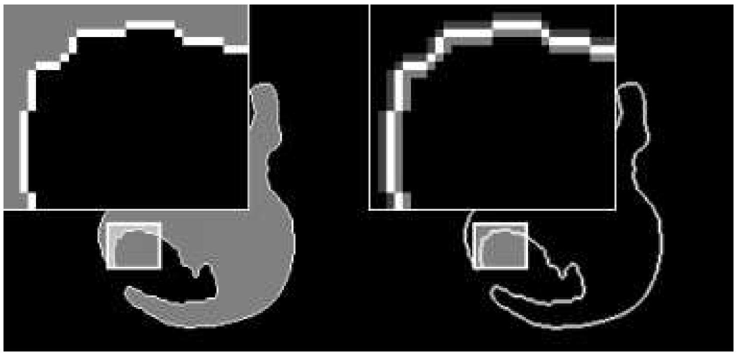


Figure 8: The same slice as shown in figure 7 after manual smoothing.

3.3.3 Segmentation Edge Smoothing

Analysis of the segmented CT slices provided a possible explanation as to why the initial surface was was so rough. The contours produced by the GIMP magic wand tool were very irregular, as can be seen in figure 7. The two arrows point out particularly noisy areas. Remember that an RBF is fit such that it will interpolate the values given for all surface and off-surface points. The line that passes through the center of each white pixel has very high curvature. The problem is multiplied when fitting a 3D surface due to registration errors between successive slices. This irregularity causes the 'bumpy' surface shown in figure 6.

The irregularity in surface pixels is mostly due to undersampling. The magic wand tool has very few pixels to work with and noisy edges are the result. To increase the smoothness of the RBF surface, the individual segmented edges needed to be smoothed. This smoothing was done by hand on each individual slice. The results of manual smoothing can be seen in figure 8.

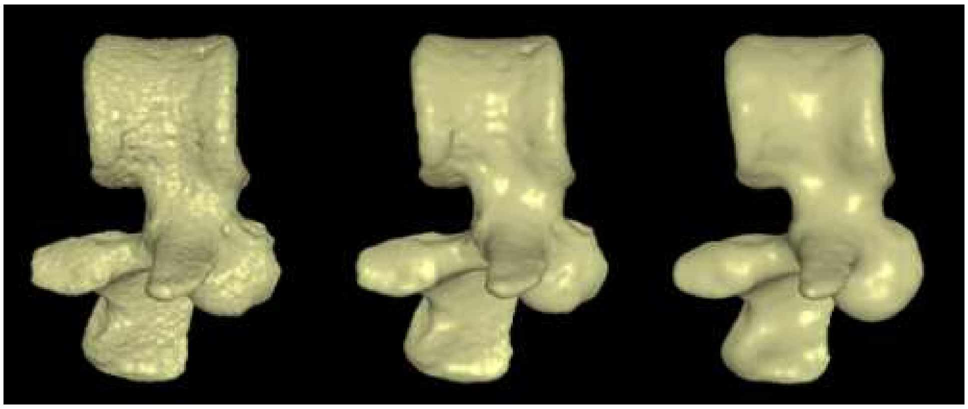


Figure 9: Results of RBF smoothing. Left image has no smoothing. Middle image has smoothing factor 0.5, and right image has smoothing factor 30.

3.3.4 Results After Smoothing

43,834 surface points remained after manual smoothing, from which 87,558 off-surface points were generated. The fitted RBF was marginally smoother than the initial surface, but still very rough. One obvious problem was that, while smoothing had been applied to each slice, no smoothing was done between consecutive slices. However, it was surprising that the pixel smoothing had so little effect. A likely culprit was the inherent smoothing applied by the resolution of the polygonization algorithm. Polygonization 'chops off' small areas of very high curvature. The high-curvature surface sections fit to the noisy pixels in figure 7 were likely removed by the polygonization algorithm. This implies that a smooth surface can not be fit directly to the point set generated from pixels.

3.3.5 RBF Smoothing

One last avenue for smoothing the RBF surface was available. A relaxation factor can be introduced which allows the RBF fitting process to generate a surface with less curvature. This process was applied with varying smoothing factors. As can be seen in figures 9 and 6, the smoothing factor is very effective. The right vertebrae in these images is arguably smooth enough to use in simulation.

RBF smoothing does come at a price. The smoothing factor reduces the accuracy of the final RBF. The fitted RBF is less constrained to pass through the surface and off-surface points given to the fitting algorithm. With a large smoothing factor, noticeable volume change can take place, shown in figure 10. The visible white pixels in the right image show a large deviation from the actual surface.

This deviation is important because density points will be generated in this area, however the CT slices contain no bone mass here. The density of points generated in these areas will be zero. This implies that the user will be able to drill several millimetres into the surface before experiencing any drill resistance. Obviously this

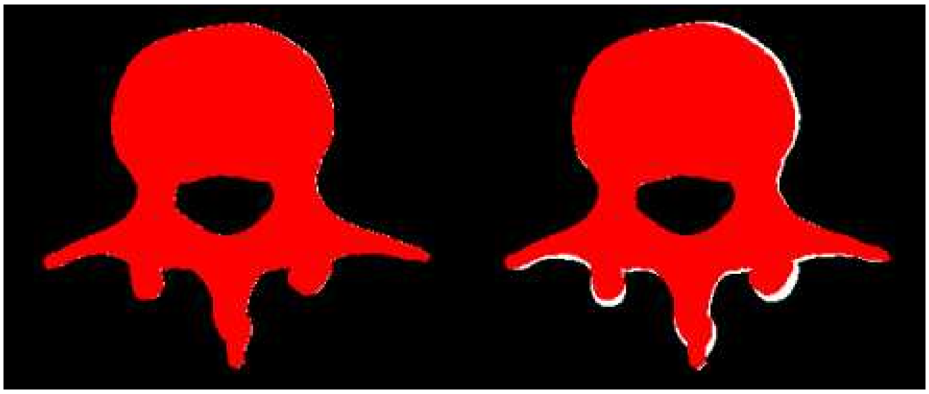


Figure 10: Comparison of fitting accuracy. The silhouette of two smoothed RBFs (white) are shown with the unsmoothed RBF (red) overlayed. The left image has smoothing factor 0.5, the right has a factor of 30.

surface cannot be used.

3.3.6 Conclusions

The pixel-based method of generating OSPs is unworkable. An accurate surface is too noisy, but a smooth surface exhibits too much volume change. A better method of generating off-surface points must be applied.

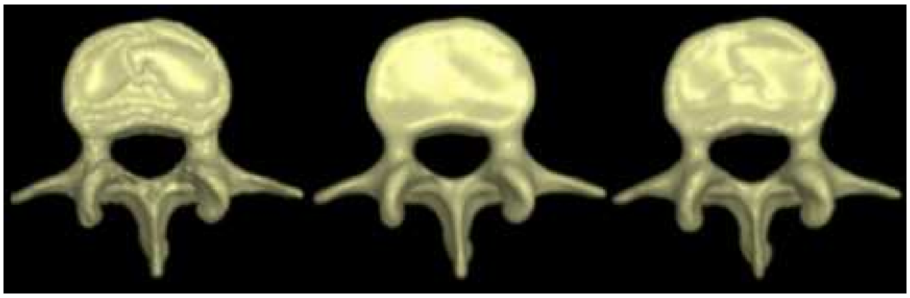
3.4 Normal Estimation Surface Reconstruction Method

3.4.1 FastRBF Normal Estimation

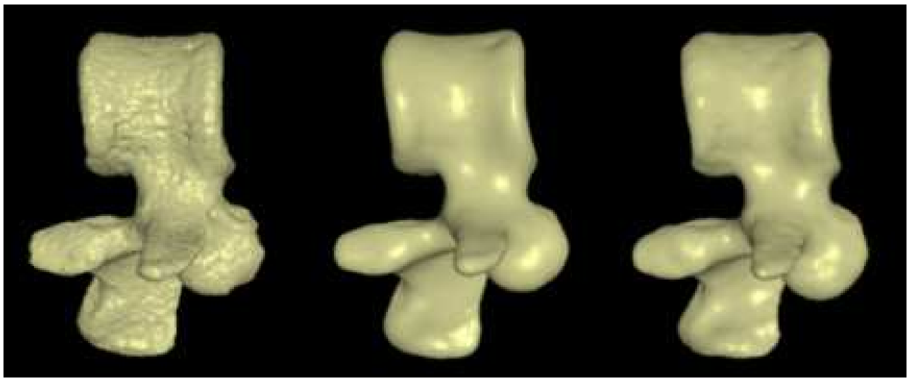
A strong determinant of final surface smoothness is the off-surface point (OSP) distance. In section 3.3, the distance to OSPs was rather low, relative to the size of the vertebrae. This produces a very constrained surface. Unfortunately more distant OSPs cannot be determined from slice images. Generating OSPs based on an estimated 2D normal is innefective for slices where even minor curvature exists in the slice (on the axis coming out of the slice image).

The typical method of generating OSPs is to project them out along the normal to the surface point [5]. Point normals are generally not available for an arbitrary point field. However, techniques have been devised to approximate a normal based on the local point field [5]. These methods rely on an outward-pointing vector to determine normal orientation. Given these outward-pointing vectors, FastRBF can generate OSPs at user-specified distances from the surface point. A new RBF is then fit to this set of surface and off-surface points.

These outward-pointing vectors can easily be generated from the CT image slices. The 4-neighborhood of each surface point is examined. Start with an initial vector of (0,0). Add one to the x coordinate if the left pixel is an 'outside' pixel, and subtract one



(a) *front*



(b) *side*

Figure 11: Comparison of normal and pixel-based OSP generation. Left and right images are pixel-based, with no smoothing and smoothing factor 30. Middle image is normal based with smoothing factor 0.5.

if the right pixel is outside. Similarly, add one to y for an outside top pixel, and subtract one for an outside bottom pixel. The resulting vector is a 2D outward-pointing vector, which is sufficient for the normal generation techniques. Note that this algorithm can produce ambiguous (0,0) vectors. Surface points with these ambiguities were simply discarded, as only 6 of the 43,834 were problematic.

3.4.2 Results

A surface point set was generated from an initial set of 43,828 surface pixels. FastRBF generated almost as many off-surface points, resulting in 87,485 final points. An RBF was fit to this set with smoothing and point reduction enabled. The set of output centers was reduced 84% to 14,344 centers. A smoother surface was fit to only 6775 centers,

Figure 12: Comparison of fitting accuracy between normal and pixel OSP generation methods. Left and right images are pixel-based with smoothing factors 0.5 and 30. Middle image is normal based with smoothin factor 0.5.

however the level of volume change was deemed to be too high.

The surface is visually much smoother, as can be seen in figure 11. In addition, the vertebrae volume is preserved quite well. The initial images in figure 12 were rendered at approximately double the resolution of the actual CT images. This implies that a one-pixel discrepancy between the smoothed red silhouette and underlying unsmoothed white silhouette is within the pixel accuracy limit allowed by the CT image.

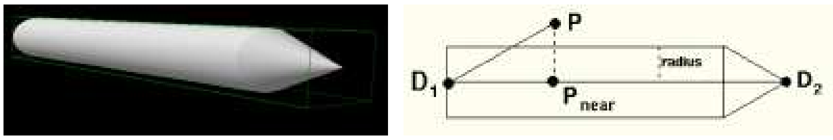
The normal-based RBF is as arguably as smooth a surface as is possible when fitting directly to pixel values. To increase surface smoothness a method such as fitting 2D piecewise splines to the surface pixels would be necessary. In addition, expert judgement is recommended to determine whether the set of surface pixels picked out from a slice is accurate.

4 The Virtual Drill

The virtual drill model is separated into two components - the drill handle and the drill bit. The drill handle would only be represented graphically and has been left out of the current implementation. It has no effect on the vertebrae drilling process. The drill bit is the only part of the system that can affect the vertebrae.

4.1 Drill Bit Geometry

The drill bit is composed of a cylinder with a conical end cap, as shown in figure 13(a). Unless the surgeon plans on drilling through the vertebrae into internal organs, the section of drill bit that needs to be tested for collisions will be no longer than approximately 50mm. The conical end-cap to the drill bit cylinder is expected to make up about 5mm of this length. The diameter of the drill bit is typically in the 2.5-3.5mm range.



(a) geometry (b) point test

Figure 13: virtual drill bit

4.2 Drill Bit Point Inside/Outside Test

Testing whether a point is inside our outside of the drill volume is a critical part of the collision detection process. Implicit definitions of a cylinder and cone exist and can be used to determine point inclusion. The canonical definitions of these primitives, oriented along an axis and based at the origin, provide a simple test. However, the drill bit may be arbitrarily oriented in space. This implies using the general equations of a cylinder or cone. These equations are rather complicated to produce and evaluate. Another option is to transform the test point back into the canonical primitive space. This requires an expensive matrix multiplication. Neither of these are very efficient.

The last option for a point inside/outside test is to use a geometrical method. This algorithm is based on a vector projection and magnitudes. The point in question is projected onto a line through the center of the drill bit, as shown in figure 13(b). The distance from point P to point P_{near} is used to determine whether or not the point is inside the drill bit volume. The algorithm is shown below.

Algorithm 1

$\vec{V}_{drill} = D_2 - D_1$	$d = \parallel \vec{V}_{dist} \parallel^2$
$\vec{V}_{point} = P - D_1$	$if\ (d > radius^2)\ then$
$\alpha = \frac{\vec{V}_{drill} \cdot \vec{V}_{point}}{\parallel \vec{V}_{drill} \parallel^2}$	$\quad outside$
$if\ (\alpha < 0 \parallel \alpha > 1)\ then$	$if\ (\alpha > \frac{length_{cylinder}}{length_{total}})\ then$
$\quad outside$	$\quad r = (1 - \alpha) * \frac{length_{cone}}{length_{total}}$
$if\ (\alpha \approx 0)\ then$	$\quad if\ (d > (r^2 * radius^2))\ then$
$\quad \vec{V}_{dist} = \vec{V}_{point}$	$\quad\quad outside$
$else$	$inside$
$\quad \vec{V}_{dist} = \alpha \vec{V}_{drill} - \vec{V}_{point}$	

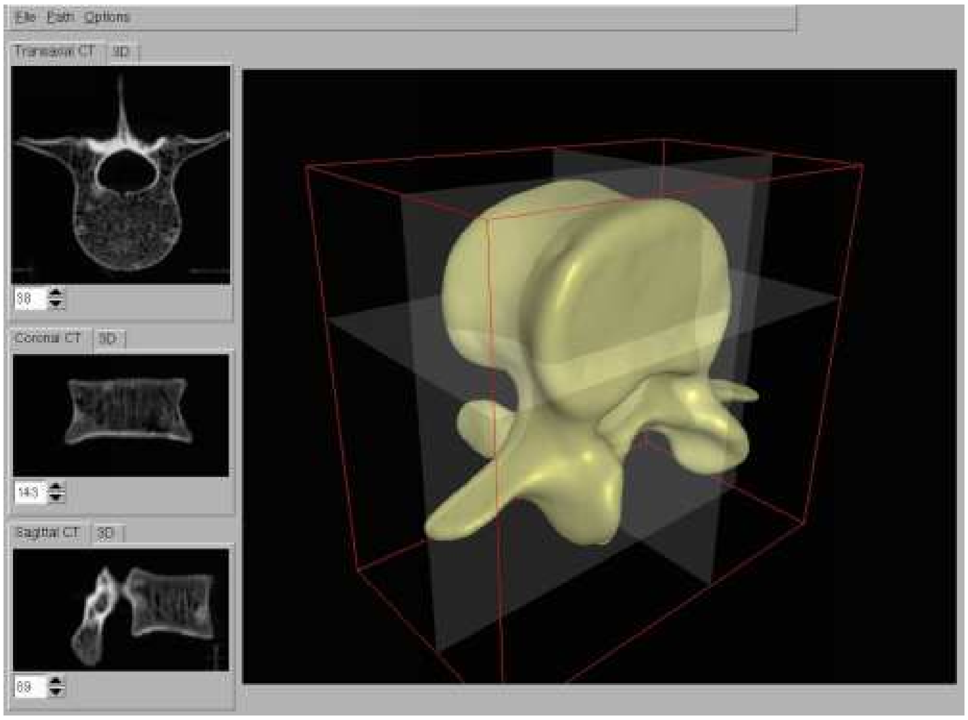


Figure 14: The Path Planner interface. The CT slice images displayed on the left correspond to the positions of the transparent guide planes intersecting the model on the right.

5 Path Planning Tools

5.1 Vertebrae Inspection

The interface used to plan drill paths is shown in figure 14. The main view window displays the vertebrae model and can be manipulated by the user. The three smaller views allow the user to step through the CT slice stack along the transaxial, coronal, and sagittal axes.

The positions of the transparent guide planes shown in the figure are dependent on the currently visible CT slice image (in each axes). These guide planes are meant to help the planner relate the CT image to the 3D model. They can be disabled for a clear view of the model surface.

5.2 Path Planning

Path planning is done by manipulating the end points of the path line. The drill surface is rendered along the path to aid the user in correct placement. Model transparency allows for manipulation of the end of the path inside the volume, shown in figure 15.

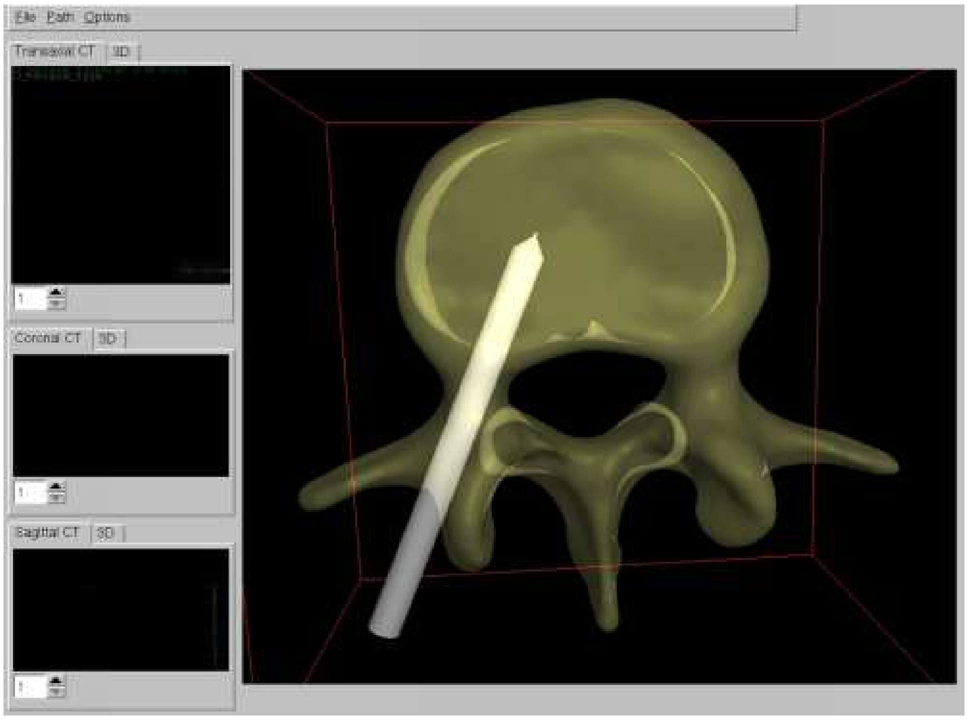


Figure 15: A sample path similar to those used for pedicle screw placement.

6 An Efficient Drilling Path Data Structure

6.1 Motivation

SpinalTap simulates the 'feel' of drilling into a rigid structure by providing collision feedback at rates of several hundred cycles per second. This feedback has to correspond with what the user sees on the screen and expects to feel. Two opposing problems exist - how to detect drill/volume collisions accurately and how to do so quickly. Generally one of those must be sacrificed to enhance the other.

Most collision detection systems are meant to be applied to large-scale interaction between volumes where few assumptions can be made about properties of the volume [24] [2][15]. SpinalTap, however, is a very specific application. The project goal is to decimate a section of a small rigid object with high accuracy. To achieve the collision rates required, a very simple, highly optimized collision detection algorithm is necessary. Arguably the simplest volume primitive is a single point. A set of points in space that fit inside the vertebrae surface can be generated, essentially creating a 'point volume'. Decimation is then accomplished by discarding points as they are found to be inside the drill volume.

Point volume density is a critical determinant of simulation realism. Too sparse a point volume and the feedback will be irregular or 'jumpy'. On the other hand, too

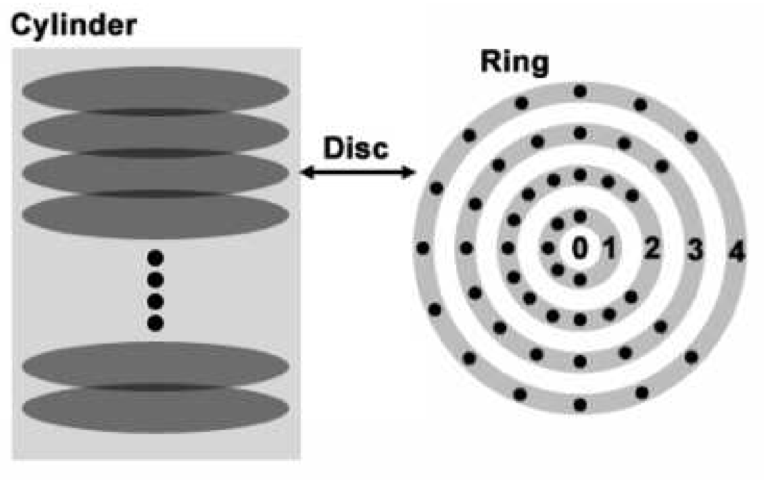


Figure 16: The cylindrical point volume data structure used for collision detection.

dense a volume results in unacceptably low collision cycle rates. SpinalTap currently uses a point density of 0.1mm. This provides adequate simulation speed. The haptic drill is not avaible for testing, so it is uncertain whether this resolution is sufficient. It has been suggested [25] that extrapolation could increase the frequency of feedback without requiring an increase in data resolution. These methods may be the only option if point resolution is too low, as it will be shown that lowering the resolution to even 0.5mm pushes the limits of current hardware.

6.2 A Cylindrical Point Volume

The obvious method of generating a point volume is to fill the entire vertebrae with points. This method results in ludicrous memory requirements and would be utterly impossible to process quickly. Here a basic fact about spinal surgery becomes very useful - spinal surgeons do extensive planning and know well before hand where they intend to drill. Using this pre-planned drill path, a smaller (but equally dense) point volume can be constructed by only generating points in the voxels along the drill path. Unfortunately initial analysis of this revised system showed that the time needed to process all voxels was still too high [27].

An alternate method involving non-static points that migrate with the drill bit surface was considered. This technique still provides the highest collision accuracy but it is unclear how to guarantee that migrating points cover the entire drill volume. This method also requires evaluation of the vertebrae RBF at run-time, which may be computationally infeasible even with fast multipole methods.

Again, the restricted problem domain comes to the rescue. SpinalTap is a system for simulation of spinal drilling surgery. Drills bits are cylindrical in shape, with a conical end cap. The surgeon can only drill along the pre-planned path in one direction - into the vertebrae. Finally, because of the conical end cap, volume decimation occurs 'outward', meaning that along any planar slice perpendicular to the drill path, the conical end cap will pass through the slice before the rest of the drill bit.

The obvious choice for the point volume is a cylinder of radius somewhat larger than the drill bit, oriented along the planned drill path. The cylinder can be split up into discs perpendicular to the drill path vector. Due to the 'drilling outward' property, these discs can be split into concentric rings of points. The data structure we are left with is a cylinder of discs of rings of points, shown in figure 16.

6.3 Filling The Point Volume

The point generation scheme begins with a path constructed using the path planning tool described in section 5. A canonical disc of point rings is generated and copied down the Z axis at 0.1mm intervals. This cylindrical point volume is then transformed to align with the path vector.

Points outside the vertebrae volume are culled using the same RBF used to create the vertebrae surface. The FastRBF software could be used for this, however in the interests of avoiding commercial tools the RBF was evaluated using the standard brute-force method. With an RBF of approximately 16,000 centers, this is an expensive operation. Culling the point volume is by far the most compute-intensive part of point volume generation. The process takes upwards of an hour for a large volume with 0.1mm point spacing. Indeed, this evaluation expense is one of the reasons why testing was limited to 0.1mm spacing. Generating a volume at 0.05mm spacing produced a point set file several hundred megabytes in size and took approximately 8 hours. Applying Fast Multipole Methods to this process would increase output speed, however the volume size is unaffected.

The final size of a point volume given the inital path length and radius is easily calculated using the following formulas:

$$points_{disc} = 3 * n * (n-1) \quad where \quad n = \frac{radius_{volume}}{spacing}$$

$$points_{volume} = points_{disc} * \frac{length_{path}}{spacing} + 1$$

To get a feel for these numbers, take an imaginary point volume of radius 4 and length 10. At 0.1mm spacing, each disc has 4,681 points and the entire cylinder contains 468,100 points. At 0.05mm spacing, each disc has 18,961 points and the cylinder contains 3,792,200 points. Halving the point spacing results in roughly an 8-fold increase in point count.

6.4 Disc Point Distribution

Since the point volume will ultimately be used to determine total feedback strength applied to the user, the distribution of points should be uniform. There is a possiblity that a completely uniform distribution would produce the physical analogue of aliasing. In this case, jittering could be applied to the uniform distribution - but an initial uniform distribution is still necessary.

Generating a uniform distribution on a regular grid is very simple, however doing so on circular discs is less intuitive. Obviously a uniform distribution along the axis of the cylinder can be achieved by simply spacing the discs at regular intervals. Inside the discs, however, each point should be equidistant from it's neighbours. One method of guaranteeing this involves circumscribing the disc inside an initial regular hexagon, then subdividing the equilateral triangles that make up the hexagon [21]). A geometric property of this subdivision is that all resulting triangle are equilateral, hence subdivision is repeated until the edge length of any triangle is less than the required point spacing. Unfortunately, another geometric property of this method is that when the subdivided points are connected with rings each ring contains only 6 points (the points of a hexagon). The high ring count provides increased collision resolution but incurs a large cost in the collision detection algorithm.

A somewhat less uniform method for populating the point disc has been devised that is optimal for collision detection. The innermost ring is created with 6 points spaced at 60 degree intervals (the vertices of a hexagon). The arc length between these points is computed and all successive rings are constructed such that their arc length is equivalent. It can be shown that the limit of the distance between the points is equal to the arc length as the radius goes to infinity. For 0.1mm spacing, this results in point spacing between 0.103mm and 0.104mm for any particular ring. The radius is determined simply by multiplying the ring number by the spacing value. Hence discs and rings are equally spaced, and points around a ring are near-equally spaced. One possible problem here is that the spacing between points on different rings is not uniform. Visually they appear to be, however further analysis has not been attempted.

7 Fast Collision Detection

7.1 Basic Algorithm

The basic collision detection algorithm is simply a point inside-outside test on the drill bit cylinder and conical end cap. First the line connecting the tip and base of the drill is projected on to the line down the center of the cylindrical point volume to determine how far along the path the drill is. Then each disc up to this point is tested against the drill volume. Testing a disc is relatively straightforward - points are processed from the inner to outer ring for inclusion in the drill volume using algorithm 4.2. Each disc maintains a pointer to the outermost completely destroyed ring, which reduces the number of points that need to be considered. Testing out to the last ring is also unnecessary. By determing where the drill line intersects the disc plane, finding the distance between this point and the center of the disc, then adding the drill bit radius, the outermost disc that can possibly have a point intersection is determined.

7.2 Correcting For On Path Bias

The collision testing algorithm performs best when the drill exactly follows the planned path. In this case the minimum possible number of rings must be tested, and these rings move from undestroyed to fully destroyed very quickly. However, as the drill diverges

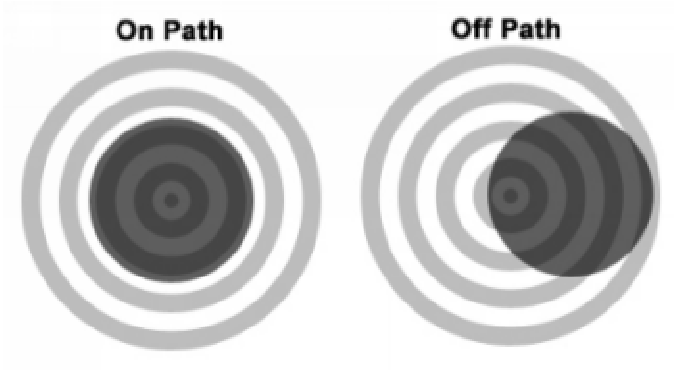


Figure 17: Degradation of the collision detection algorithm due to off-path drilling. All rings that overlap the drill (dark grey circle) must be tested.

from the planned path, more rings must be tested, and many points in these rings will not intersect the drill. This can be seen in figure 17. On the path only 3 rings need to be tested, while in the off path case all rings must be tested.

Testing all these extra points results in a significant slowdown. However, due to the cylindrical nature of the drill bit, these extra tests can be avoided. The elliptical projection of the drill bit onto the disc plane is convex. Only two intersections with any given ring are possible. Collision testing must start at a point on the ring that is inside the drill bit cylinder. Testing proceeds around the ring away from this point in each direction until a non-intersecting points are found. At this point it is guaranteed that no additional points intersect the drill bit. The vector from the disc center to the drill line / disc plane intersection point (calculated to determine the outermost ring) can be used to find a point that must be inside the volume if any collisions will occur. This algorithm is currently not implemented in the SpinalTap system.

8 Rendering

8.1 Vertebrae Surface

The vertebrae surface is rendered using the polygonal mesh generated from the vertebrae RBF. During the point volume generation stage, triangles entirely inside the volume cylinder are removed, creating a hole around the drilling area. This method occasionally leaves small holes around the edge of the drilling volume, as a triangle may intersect the volume cylinder but it's vertices may all be outside. This case could be avoided by casting rays along the edges of triangles near the volume and testing for cylinder intersection.

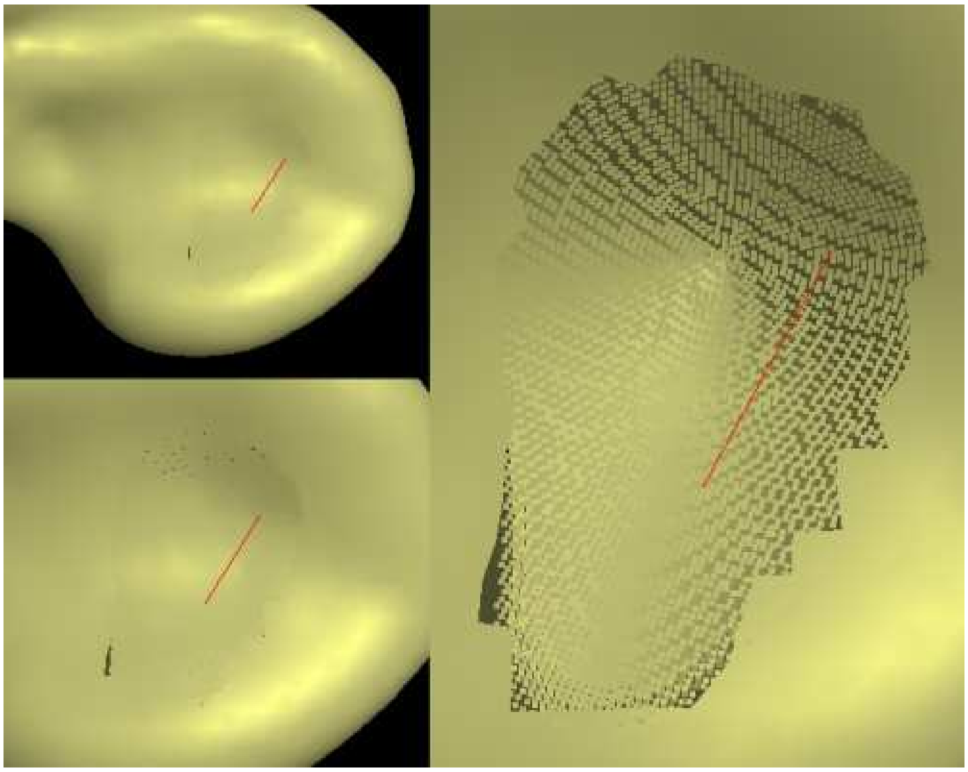


Figure 18: The point rendering method breaks down as the surface gets closer to the viewer. The upper left image is correct, however artifacts begin to appear in the lower left image and grow progressively worse, until the surface breaks apart.

8.2 Point Volume Rendering

Rendering of the point volume surfaces is done using an extremely basic splatting zwicker[22] [30] algorithm. Points in the volume are simply rendered using the GL_POINTS OpenGL primitive. As can be seen in figure 18, this method is of limited effectiveness. In an actual virtual surgery environment the vertebrae would be smaller than the picture on the left, so in this case rendering the points directly works well. However, as the vertebrae is enlarged the distance between points in relation to pixel size grows, and the surface breaks apart. Switching to a different splatting primitive at close range (such as spheres) and using adaptive splat sizing would result in a closed but rounded surface.

While the GL_POINTS primitive is very fast, the point volume used for testing contained approximately 1 million points. During drilling processor time is at a premium and the user is unable to see inside the drilling volume because the drill bit blocks the hole. Only surface points need to be rendered. Surface points are determined by examining the RBF value for the point. If the value is within a pre-defined threshold then

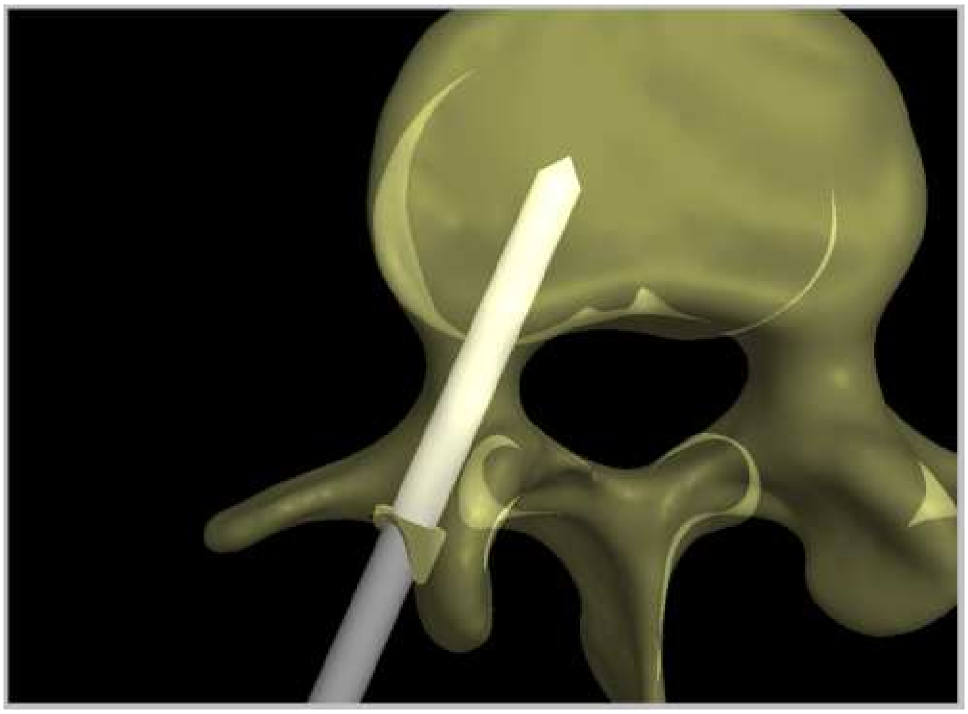


Figure 19: A transparent view of the model during a drilling simulation. The surface discs can be seen half way down the drill bit.

it is marked as a surface point. The current implementation marks and renders entire discs as surface discs instead of individual points, shown in figure 19. This costs some rendering time but avoids increasing the size of the point data structure. In practice, very few discs are surface discs and the rendering overhead is minimal.

A final issue with point rendering is normal calculation. Normals for surface points are taken from the vertebrae RBF. At inner points normals only need to be calculated once they are visible. When the user stops drilling, the final drill bit orientation is used to calculate normals for non-surface points near the drill bit surface.

Frame rates of 10 to 15 frames per second were achieved using this technique. The largest computational cost is the conditional test that must be performed for each point to determine whether or not it has been destroyed. If a color was stored for each point, the alpha component could be set to zero for destroyed points. This would allow for batch rendering of all points and provide a significat speedup. However, storing a 4-component color for each point increases the memory footprint of the point volume by one half.

9 Results and Analysis

9.1 A Realistic Sample Path

One of the main reasons surgeons drill in to vertebrae is to insert pedicle screws. These screws are attached to metal plates that stabilize the vertebrae when a ruptured disc is removed. Pedicle screws are inserted in roughly the same orientation as shown in figure 19. This path provides a good test for the performane of the collision system. Besides being relevant to intended use, it is nearly as long a path as is possible through a vertebrae. The culled point set contained 986,958 points in 378 discs.

9.2 Initial Simulation Attempts

A real haptic drill was not available to test the collision algorithms, so simulation of the positional input from an imaginary haptic drill was attempted in software. In hardware, the drill positional input would be received asynchronously by the operating system. The rendering and collision engines should also be running asynchronously, as the renderer cannot keep pace with the collision engine. This architecture was simulated using three threads on dual-processor Intel Pentium 3 and 4 machines. One thread moved the drill forward along the path. A second thread ran the collision algorithm continuously, and a third rendered the scene when necessary. Ideally, the collision algorithm would run one processor while the drill and visual updates ran on the other.

SpinalTap runs in the Linux operating system, which is not a real-time operating system (RTOS). An RTOS provides hard limits on things like process/thread priority and wait times. Linux does not. The first technique tried was to have the drill thread continuously move the drill a short distance (less that 0.001mm), sleeping for several hundred microseconds between moves. Initial results were very promising - the collision algorithm ran at several thousand cycles per second on a 1400 Mhz Pentium 4 dual-processor machine. However, the drill movement thread only woke up 100 times per second. Linux cannot provide an alarm signal at any higher frequency, so for most collision cycles the drill position had not changed. When the drill doesn't move, no new collisions occur, and the collision engine has to do very little work.

An attempt to enhance this system was made by removing the sleep call altogether. The drill movement thread then ran at 30,000 to 60,000 updates per second. At this rate, the movement distance had to be reduced from 0.001mm to 0.000001mm. Unfortunately, because of the limited accuracy of single precision floating point, large errors accumulated when moving such a small distance. The drill often diverged so badly from the path that it started moving backwards. In addition, because Linux cannot guarantee processor allocation to a particular thread, it was impossible to determine whether or not the collision engine had a processor to itself.

9.3 Collision Engine Timing

Attempts to simulate realistic drill input had to be abandoned due to deficiencies in Linux thread scheduling. However, some sort of measure of the collision engine efficiency was still needed. A simplified architecture was designed wherein the collision

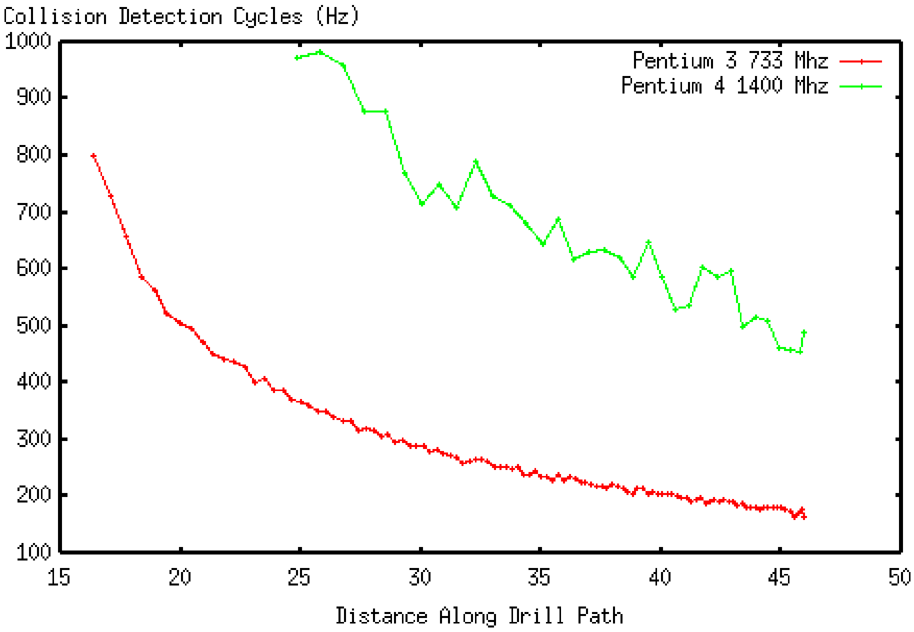


Figure 20: Comparison of collision detection rates between a 733 Mhz Pentium 3 dual processor machine and a 1400 Mhz Pentium 4 dual processor machine.

engine works in lock-step with the drill update thread. First the drill update thread moves the drill a small distance, then the collision engine runs, and the cycle repeats. The rendering thread was left unchanged. In this model it was safe to assume that the collision engine / drill movement cycle would be allocated one processor, and the drill rendering thread the other. The number of collision cycles per second was output and a graph of one of the data sets is shown in figure 20.

On a dual-processor 1400 Mhz Pentium 4 machine, collision cycle frequences of over 400Hz were achieved. Initial collision cycle frequencies are much higher because the number of point tests required increases as the drill penetrates the point volume. Point test count is highest near the end of the path, so the final numbers in the graph must be used when analyzing efficiency. The probability of increasing this rate beyond the 1000Hz barrier is discussed in section 9.5.

9.4 Problems

9.4.1 Off Path Slowdown

The graph shown in figure 20 was generated by a drill moving directly along the planned path - the optimal situation. An additional run of the simulation was made with the drill placed near the edge of the point volume. This results in less efficient collision point culling, described in section 7.2. The timing results of this off-path sim-

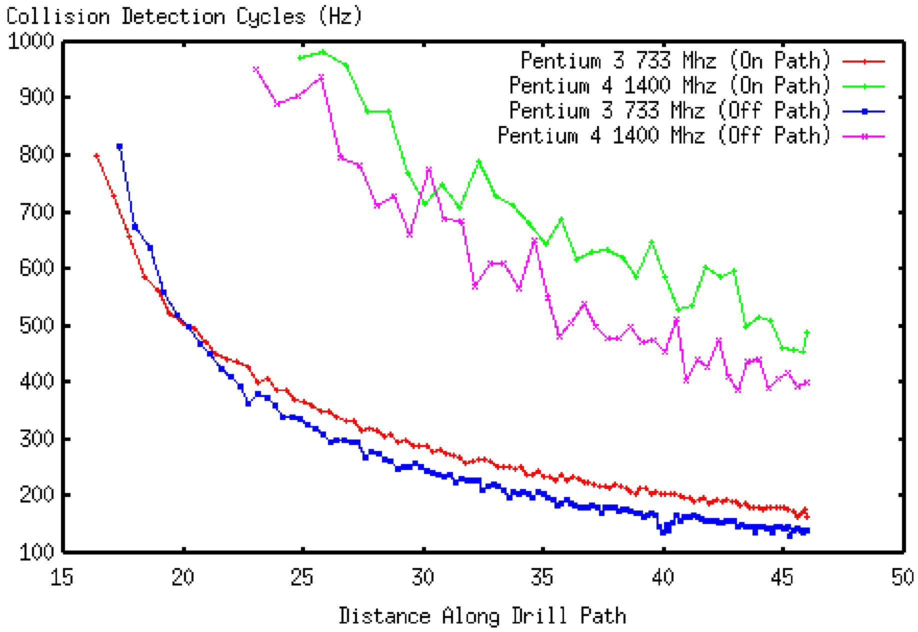


Figure 21: Comparison of on-path and off-path collision detection rates between a 733 Mhz Pentium 3 dual processor machine and a 1400 Mhz Pentium 4 dual processor machine.

ulation are plotted with the on-path values in figure 21. A slow down of approximately 20% is apparent on both the 1400Mhz and 733Mhz processors.

9.4.2 Hardware Bottlenecks

Correctly interpreting the results shown in figures 20 and 21 requires some knowledge of the underlying computer hardware. The first obvious discrepancy is the relatively smooth decrease in speed of the Pentium 3 chip versus the 'jumpy' irregularities in the Pentium 4 graphs. This can be traced back to peculiarites of the Pentium 4 architecture, particularly the dual-channel RDRAM memory bus. Dual-channel RDRAM has a peak throughput rate of 3.2 GB/s, however it has a rather high read latency. In addition, the Pentium 4 has a much deeper execution pipeline than the Pentium 3. Unless great care is taken to schedule instructions properly, read latency can cause pipeline stalls, which essentially force the processor to slow down until the memory can 'catch up'.

The next relevant statistic is that the 1400 Mhz Pentium 4 provides about a 2.8 times speedup in the on-path case, yet the clock frequency is only twice as high. The speedup rises to 2.9 times in the off-path case. Again, examining the memory bus architecture provides some answers. The Pentium 3 has the same dual-channel RDRAM as the Pentium 4, however it only has a 133Mhz memory bus. This means it can only support a peak memory throughput rate of 1.06 GB/s [8]. The Pentium 4 has a 400Mhz memory bus and supports the full 3.2 GB/s throughput rate. This is where the additional speed increase comes from. The speedup increases in the off-path case because more memory must be read (more of the volume must be processed).

The effects of memory latency and pipeline stalls on the Pentium 4 are especially apparent when comparing the on-path and off-path graphs for that architecture. In some cases the off-path drill actually beats the on-path drill, while in others it does much worse than the average 20% slowdown.

9.4.3 Limits on Current PC Hardware

Several fundamental limits can be extrapolated from the test data and information about available hardware. Each point uses 32 bytes of memory, so the 986,958 point data set requires 30.1 MB of RAM. In the on-path case the entire data set is never needed for a single collision cycle, some basic analysis shows that approximately 20% of the volume is needed in the worst case (at the end of the path). If the smart off-path collision detection algorithm described in section 7.2 is implemented, off-path drilling requires no more memory reads than on-path drilling, so 20% percent is a reasonable estimate for the actual number of points read in the most expensive collision cycles.

The question now arises of how much the point density can be increased before the simulation speed drops below acceptable levels, say 100 Hz. The assumption is made that a processor exists that can process data as fast as it is read (this assumption is not so unrealistic). At 32 bytes per point, this limits point reads to about 1 million per collision cycle, implying that the full data set is approximately 5 million points. In section 6.3 it was noted that halving the point spacing causes an 8-fold increase in point count. Reducing the resolution of the sample pedicle screw path to 0.05mm would increase the point count to over 8 million.

Note that this estimate is greatly oversimplified - in particular, requiring 20% of the point volume for a collision test seems rather high. However, the memory bus limits do exist, and will be problematic when trying to increase the realism of the simulation.

9.4.4 Numerical Accuracy

The memory bandwidth limits discussed in the previous section are only one difficulty with increasing simulation realism. Another is floating point numerical accuracy. The current system relies on single precision floating point values. Point coordinates are stored in world space, and the coordinates at the extents of the vertebrae volume require 3 digits on the left side of the decimal. It is commonly accepted that single precision floating point provides about 7 decimal digits of numerical accuracy. With 0.1mm spacing on points that require 3 digits for the integer world coordinate, only a few digits remain for spacing and calculation accuracy.

As the point spacing is reduced, more error will accumulate, until point coordinates become indistiguishable. The solution to this problem is to use double precision floating point numbers for point coordinates and collision testing. However, double precision math is much slower than single precision math. In addition, double precision point coordinates increase the size of the point data structure by 40%. This increases memory bus requirements by 40%, implying that the spacing will have to be reduced to meet memory requirements.

9.5 Parallel Collision Detection

Assuming that a 0.1mm point density is viable, it may be necessary to increase the collision cycle rate. Several optimizations can be made to the existing code using SIMD vector operations and optimized cache pre-fetching. These techniques will provide a reasonable speedup but are very processor-specific. The real question is whether the algorithm can be run in parallel on multiple processors

Since each disc is processed independently, the collision algorithm should scale directly to multiple processors. The range of discs to test can be initially determined, and a subset of discs assigned to each processor. No concurrency problems can arise as each set of discs is independent and no processor needs to know anything about the other processors. The number of points to process on each disc is relatively even, so work distribution between processors would be fairly uniform. It is likely that the scaling factor for parallel execution will be nearly linear.

10 Conclusions and Future Work

A model of an L3 vertebrae was successfully reconstructed from CT scan data. The accuracy of the model is restricted by the manual methods used to segment the CT slice images. While a moderately smooth surface was reconstructed, manual segmentation is clearly not the best way to determine surface points and smooth out edges. Fitting smooth contours to the individual slices would create a much smoother RBF. Development of a method for increasing consistency between the edges on consecutive slices is also critical. Expert judgement is needed to determine if the resulting RBF surfaces are acceptable.

The SpinalTap Path Planner user interface is very minimal. Many additions would be useful in a realistic surgery planning environment. One rather important omission was displaying the drill path on the CT slice images - without this it is difficult to tell exactly where the drill is going. If the Path Planner is to be useful in a real simulation system, Fast Multipole Methods must be implemented so that point generation can be done in a reasonable amount of time.

Volume decimation using a cylindrical point volume appears to be a very effective technique for drilling in to solid surfaces. However, the limitations on point density require investigation. It may be possible to apply more elegant sorting techniques. The method implemented in the SpinalTap system is essentially a brute force technique. A multiresolution algorithm in which a higher density point set is only tested near the drill tip (where collision count is high) may allow for increases in point density that avoid hardware limits. Extending the collision engine to a parallel implementation will increase the number of collision cycles that can be processed. If update frequencies of several hundred hertz are sufficient, a parallel algorithm will allow point density to rise and increase the realism of the simulation.

SpinalTap, in it's current state, provides a solid architecture for a realistic drilling simulation system. Lacking a haptic drill, it is difficult to determine whether or not the current point density and collision test rate are adequate. Even with a haptic drill in place, a real-time operating system seems necessary to provide the feedback rates required. These components lay the foundation for a powerful drilling simulation system that will be applicable to any sort of material for which density information can be obtained.

References

[1] Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 109–118, Los Angeles, California, August 1995. ACM SIGGRAPH / Addison Wesley.

[2] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):223–232, July 1989.

[3] R. Beatson and G. Newsam. Fast evaluation of radial basis functions, 1992.

[4] R. K. Beatson and L. Greengard. A short course on fast multipole methods. *Wavelets, Multilevel Methods and Elliptic PDEs*, pages 1–37, 1997.

[5] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *Proceedings of SIGGRAPH 2001*, pages 67–76, August 2001.

[6] Jonathan C. Carr, W. Richard Fright, and Richard K. Beatson. Surface interpolation with radial basis functions for medical imaging. *IEEE Transactions on Medical Imaging*, 16(1):96–107, February 1997.

[7] David T. Chen and David Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):89–98, July 1992.

[8] Intel Corporation. Intel chipsets comparison chart. http://developer.intel.com/design/chipsets/linecard.htm.

[9] Stéphane Cotin, Hervé Delingette, and Nicholas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, January - March 1999.

[10] Frank Dachille, Kevin Kreeger, Baoquan Chen, Ingmar Bitter, and Arie Kaufman. High-quality volume rendering using texture mapping hardware. *1998 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pages 69–76, August 1998.

[11] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. *Proceedings of SIGGRAPH 2001*, pages 31–36, August 2001.

[12] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):65–74, August 1988.

[13] Szekely et al. Segmentation of 2-d and 3-d objects from mri volume data using constrained elastic deformations of flexible fourier contour and surface models. *Medical Image Analysis*, 1(1):19–34, 1996.

[14] S. Gibson, J. Samosky, A. Mor, and C. Fyock. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback. *Lecture Notes in Computer Science*, 1205:369–??, 1997.

[15] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 171–180, New Orleans, Louisiana, August 1996. ACM SIGGRAPH / Addison Wesley.

[16] Matthew Holton and Simon Alexander. Soft cellular modelling: A technique for the simulation of non-rigid materials. *Computer Graphics International '95*, June 1995.

[17] Qing hong Zhu, Yan Chen, and Arie Kaufman. Real-time biomechanically-based muscle volume deformation using fem. *Computer Graphics Forum*, 17(3):275–284, 1998.

[18] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):71–78, July 1992.

[19] A. Kelemen, G. Sz, and e Gerig. Three-dimensional model-based segmentation of brain mri. In B. Vemuri, editor, *IEEE Workshop on Biomedical Image Analysis*, pages 4–13. IEEE Computer Society Press, 1998.

[20] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. Parish. Simulating facial surgery using finite element methods. *Proceedings of SIGGRAPH 96*, pages 421–428, August 1996.

[21] Brendan Lane. Personal communication, February 2002.

[22] Marc Levoy and Turner Whitted. The use of points as a display primitive. Technical Report 85-022, University of North Carolina at Chapel Hill, January 1985.

[23] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):163–169, July 1987.

[24] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):289–298, August 1988.

[25] G. Picinbono and J. Lombardo. Extrapolation: a solution for force feedback. In *International Scientic Workshop on Virtual Reality and Prototyping*, pages 117–125, Laval, France, June 1999.

[26] Mark A. Sagar, David Bullivant, Gordon D. Mallinson, Peter J. Hunter, and Ian W. Hunter. A virtual environment and model of the eye for surgical simulation. *Proceedings of SIGGRAPH 94*, pages 205–213, July 1994.

[27] Ryan Schmidt. Spinaltap interim report. 2002.

[28] G. Turk and J. O'Brien. Variational implicit surfaces. Technical Report GIT-GVU-99-15, Georgia Institute of Technology, 1998.

[29] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[30] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 371–378. ACM Press / ACM SIGGRAPH, August 2001.