Achieving Proper Exposure in Surgical Simulation

Christopher SEWELL[a], Dan MORRIS[a], Nikolas BLEVINS[b], Federico BARBAGLI[a], Kenneth SALISBURY[a]

Departments of [a] Computer Science and [b] Otolaryngology, Stanford University, Stanford, CA, USA

Abstract. One important technique common throughout surgery is achieving proper exposure of critical anatomic structures so that their shapes, which may vary somewhat among patients, can be confidently established and avoided. In this paper, we present an algorithm for determining which regions of selected structures are properly exposed in the context of a mastoidectomy simulation. Furthermore, our algorithm then finds and displays all other points along the surface of the structure that lie along a sufficiently short and straight path from an exposed portion such that their locations can be safely inferred. Finally, we present an algorithm for providing realistic visual cues about underlying structures with view-dependent shading of the bone.

Keywords. Surgical simulation, metrics, visibility, exposure, mastoidectomy

Introduction

In addition to providing the obvious advantages of low-cost, easily-accessible training opportunities, computerized surgical simulators are in position to take advantage of complete knowledge of the virtual environment and of the user's actions to analyze performance with regards to a wide range of elements of good surgical technique and to afford the trainee with valuable constructive criticism. Although some existing simulators report rudimentary metrics such as time to completion and collisions between instruments and objects that should not be touched, the development and generalization of metrics for a wide variety of surgical procedures remains an important open problem in the field.

In close collaboration with an otolaryngologist, we have developed a visuo-haptic simulator for mastoidectomy, a surgical procedure in which a portion of the temporal bone is drilled away in order to access the inner ear [1]. A data file is recorded when a user runs the simulator, and this file can be read and played back in a program that allows performance to be visualized and analyzed. We have devised, and are continuing to develop, a number of metrics for use with this simulator, including comparisons of the trainee's force profiles, velocity profiles, and bone removal characteristics to those of "model" mastoidectomies performed on the simulator by expert surgeons.

One important technique common throughout surgery is achieving proper exposure of critical anatomic structures so that their shapes, which may vary somewhat among patients, can be confidently established and avoided. In the context of this procedure, achieving proper exposure involves drilling until only a thin layer of bone remains over vulnerable structures (such as the facial nerve, sigmoid sinus, and dura). When the layer of bone is sufficiently thin, the structure can be seen, due to the partial transparency of the bone. Although it is not necessary to expose the entire structure, many, such as nerves and blood vessels, twist and turn in unpredictable ways, so it is imperative to expose enough of it such that the entire shape of the structure (within the surgical field) can be confidently inferred.

A few simulators, such as 5DT's Upper GI Endoscopy Simulator, track percentage of surface area visualized with the scope [2]. We have previously reported on a related metric that determines whether bone was within the user's field of view when drilled away, since it is essential to maintain proper visibility of the drilling region [3]. This enables the surgeon to avoid vulnerable structures just below the bone surface. If instead some bone is removed by "undercutting" (drilling beneath a shelf of bone that obscures visibility), there is increased risk of damaging these structures.

In order to accurately measure a trainee's ability to properly expose critical structures, the simulator must provide the user with realistic cues as to the structures' locations. In temporal bone surgery, the most important cues are visual, as a thin layer of bone is sufficiently translucent to make out underlying structures. This visual effect must therefore be provided in the simulator in order for our exposure metrics to be meaningful.

1. Direct Exposure

In our simulator, soft tissue structures are modeled using triangulated meshes, while the bone is represented using a hybrid data structure that allows computation of appropriate drill forces using rapid collision-detection in a spatially-discretized volumetric voxel representation while graphically rendering a smooth triangular mesh that is modified in real-time as the voxels are drilled away.

A given point on a structure is considered fully exposed when the ray from the given point to the viewpoint does not intersect any bone voxels nor any other soft tissue structures. Intersection between this ray and the other meshes is checked using an axis-aligned bounding box hierarchy [4], and collision detection between the ray and the bone is performed by sampling the ray at small increments (proportional to the voxel grid resolution) and directly indexing into the voxel array for each sample position to determine if there is a bone voxel at that location. This method does not guarantee that all ray-voxel intersections are found; the ray may cut through a corner of a voxel between samples. This could be remedied by sampling at increments equal to the voxel resolution and performing a box-ray intersection test for all existing voxels in the twenty-seven neighbor slots in the voxel grid for each sample point, but in general this is not necessary because the bone is partially transparent and we are primarily concerned with rays that intersect reasonably thick layers of bone. Points along the ray

beyond the maximum extent of bone (which, for computational simplicity, can be represented by a primitive bounding volume) need not be tested.

In reality, a structure should not be fully exposed; instead, there should remain a layer of bone thin enough such that the underlying structure can be seen (due to the partial transparency of the bone) but thick enough such that the drill does not touch the structure. Thus, samples closer to the given point than some user-specified threshold distance are not considered in the bone collision tests. See Figure 1A.

Each vertex on a selected structure is tested to determine whether it is directly exposed. Since this depends on the viewpoint, it must be recalculated whenever the viewpoint changes. In our simulator, the viewpoint can be repositioned by the user using the camera tool. It is not good practice to attempt to identify the subtle visual cues needed to determine a structure's location while in the midst of moving the camera, so the visibility checks are only needed at the beginning and end of a camera movement. (Since bone is only removed and not replaced, exposure cannot decrease for a constant viewpoint, so the need for a check at the end of one movement is obviated by the check at the beginning of the next movement.)

The run-time for the voxel collision detection is proportional to the product of the number of vertices, the voxel grid resolution, and the number of camera movements. Ray-mesh collision detection is also run once for each vertex on the structure of interest after each camera movement; each call could involve checks against up to m triangles (where m is the total number of triangles in all meshes in the simulation), but the bounding volume hierarchy usually reduces this closer to O(log m) operations.

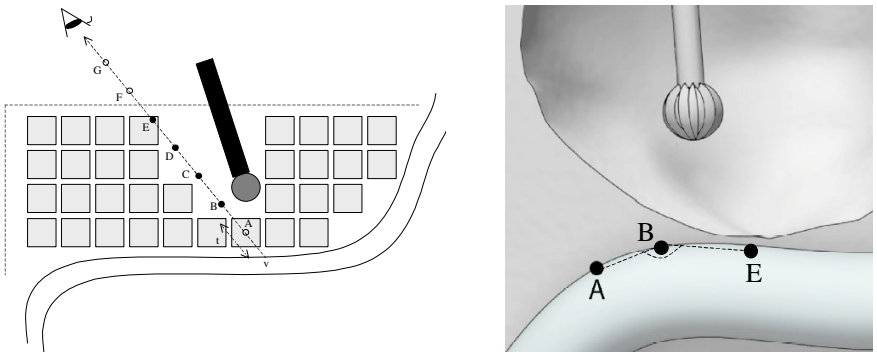


Figure 1A. At left, a given point (v) on a vulnerable anatomic structure (curved tube) is tested for direct exposure by tracing a ray from it to the viewpoint (eye). Points (A through G) at regular increments along the ray are tested for intersection with the bone voxels (light gray squares). Points (A) less than some threshold distance (t) from the given point are ignored, since proper exposure allows for a thin layer of bone to remain. (In fact, the trainee could be penalized for removing bone in this region, too close to the structure.) Points (F and G) beyond the bounding volume (large dashed rectangle) of bone need not be checked. Here points B, C, and D are checked and found not to intersect, but a collision is detected at Point E, so point v is not yet exposed. **Figure 1B.** At right, a point (E) is considered directly exposed if the overlying bone is sufficiently thin, as shown in Figure 1A. The position of another point (A) on the structure may then be safely inferred if the path distance along the surface (AB+BE) and "curvature" (the angle ABE along the surface through path midpoint B) are less than specified values.

2. Inferred Exposure

Having directly exposed some points along a structure, a surgeon may be able to confidently infer the location of other nearby points along the structure, provided that the structure is relatively straight in the vicinity. An expert may be able to establish a set of inferred points that includes all points of the structure within the surgical field by directly exposing only a small number of points in optimal locations; thus, the ratio of the sizes of the inferred and directly exposed sets, in addition to the absolute sizes of these sets, may be useful for determining a user's level of expertise.

Given a set of directly exposed points, as determined by the algorithm described in the preceding section, we make use of Dijkstra's algorithm [5] to determine the set of points whose locations can be safely inferred. The surface of the structure's mesh can be viewed as a graph (with the edges defined by the triangulation), and we attempt to find vertices for which there exists a path along the surface of sufficiently short length and sufficiently small curvature from a directly exposed point. Distance between two points along the surface is a more meaningful measure of their proximity than absolute distance because many structures (such as nerves and blood vessels) may twist and turn unpredictably. The mesh triangulation needs to be sufficiently fine and well-conditioned (i.e., no sliver triangles) so that the distance between any two points along edges of the triangulation closely approximates the shortest possible distance along the surface (although extremely high accuracy of distances is not necessary for this metric).

Dijkstra's algorithm is run once for each directly exposed vertex, which serves as the source. The algorithm repeatedly selects the vertex from the priority queue with the smallest shortest path estimate, and considers all outgoing edges from this vertex. For one such edge (v_1 , v_2), where v_1 is the vertex just returned by the priority queue, if v_1's shortest-path estimate plus the distance between v_1 and v_2 is less than v_2's current shortest-path estimate, the edge may be "relaxed". Relaxing an edge involves adding v_2 to the queue and to the set of inferred points and updating its shortest-path estimate. However, the edge is not relaxed (and thus v_2 is not added to the set of inferred points) if v_2's new shortest-path estimate is greater than some user-specified distance threshold, or if its curvature relative to the source exceeds another specified threshold. Curvature is calculated as the deviation from 180^0 of the angle formed by the source, the midpoint along the path from the source to v_2, and v_2. (The curvature is only calculated for paths longer than some minimum threshold, since the curvature of very short paths is heavily influenced by the triangulation topology.) Thus, only vertices for which there exists a sufficiently short and straight path along the surface from a directly-exposed vertex are added to the set of inferred points. See Figure 1B.

The asymptotic running time of Dijkstra's algorithm is O(E log V), where E is the number of edges and V the number of vertices in the graph. If, as is almost always the case with a mesh, E = O(V), (or, if the priority queue is implemented as a Fibonacci heap), this reduces to O(V log V). Since edges and vertices beyond the distance and curvature thresholds are never touched, V can actually be much less than the total number of mesh vertices if only a small region has been directly exposed and the distance and/or curvature thresholds are tight. This algorithm is called once for every directly exposed vertex after each camera movement.

3. View-Dependent Shading

In order to mimic the partial transparency of bone that enables structures to be seen without removing all overlying bone, we shade bone voxels near structures with appropriate colors or texture maps. A straightforward way of implementing this effect is to pre-compute the distances of all bone voxels from the underlying structures and then to interpolate between the shading color and natural bone color for each voxel in proportion to its distance. The distance between a voxel and a mesh is determined as the minimum distance between the voxel's (center) position and any triangle of the mesh. The distance between a point and a triangle is calculated by projecting the point onto the plane of the triangle, as described by Jones [6].

However, this has the effect of making the structure appear thicker all the way around. For example, in Figure 2A, if a viewer looks along the dotted line, he/she will think the dark structure (with gray shading) is directly along his/her line of sight, when in fact it is not. This effect is unrealistic and misleading.

Instead, a ray can be cast from the viewpoint through each voxel, as shown in Figure 2B. If and only if this ray intersects the structure is the voxel shaded, with intensity inversely proportional to the distance along the ray between the collision point and the voxel. As in Section 1, collision detection between this ray and the mesh is performed using an axis-aligned bounding box hierarchy. Rather than trace a ray through every voxel in the entire array, only those with a distance from the mesh (pre-computed as described above) less than a user-specified shading radius threshold need be considered. As in Section 2, these calculations depend on the viewpoint, but the rays are recast only at the end of every camera movement, since this visual effect is too subtle to be of great concern while actually moving the camera.

+

For each camera movement, ray-mesh collision detection is called once for each of m voxels, where m depends on the shading radius. Again, the bounding volume hierarchy usually reduces the potential checks against up to n triangles (where n is the number of triangles in the mesh for which the bone is being shaded) for each call to close to O(log n) operations.

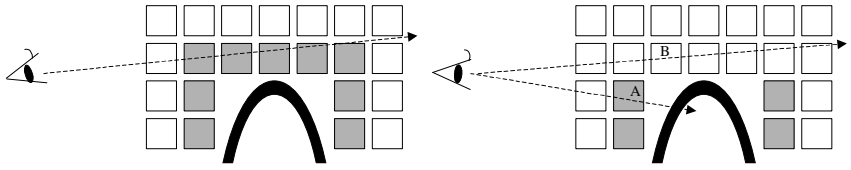
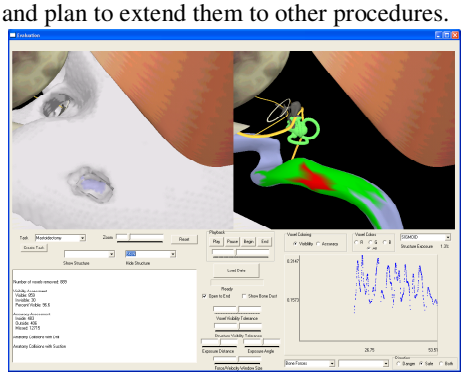


Figure 2A. At left, all voxels (squares) within a small radius of an underlying structure (black arc) are shaded. When the user looks along the ray shown, he/she sees shaded voxels, making it appear that the structure is along this line of sight, when in fact it is not. **Figure 2B.** At right, rays are cast from the viewpoint (eye) through each voxel within a small radius of the structure, and only those for which this ray intersects the structure are shaded. Thus, voxel A is shaded while voxel B is not.

4. Discussion

View-dependent shading has been incorporated into our interactive simulator, and an analysis of direct and inferred exposure has been added to our metrics console, which can replay runs of the simulator in real-time while providing visual and quantitative feedback, as shown in Figure 3A. While the simulation replays in the left panel, regions of selected structures shown in a cut-away view in the right panel are shaded based on whether proper exposure has been achieved. An example of shading the bone in the interactive simulator using a texture map is given in Figure 3B.

We are beginning a user study in which we will attempt to establish construct validity for several of our metrics, including those discussed in this paper. We intend to begin looking more closely at metrics related to force and velocity profiles, but some work remains for visibility-related metrics, including accounting for view obstruction by bone dust. Most of our work on metrics thus far has been related to temporal bone surgery, but we believe that many of these ideas are fundamental throughout surgery and plan to extend them to other procedures.



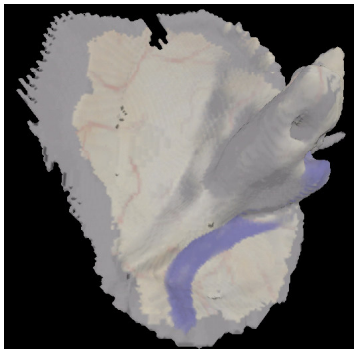


Figure 3A. At left, our metrics console, playing back a run of the simulator. In the left panel, shading of the bone is visible as the sigmoid sinus is approached. In the right panel, underlying structures are shown. The directly exposed portion of the sigmoid is shaded in one color, while the portion that can be safely inferred from the exposed region is shaded in another color. **Figure 3B.** At right, the back of the temporal bone, shaded in the proximity of the sigmoid sinus and of the dura, using a texture map applied to the voxels.

Acknowledgement and References

We would like to acknowledge NIH Grant R33 LM07295 for funding this work.

[1] Morris D, Sewell C, Blevins N, Barbagli F, Salisbury K: A collaborative virtual environment for the simulation of temporal bone surgery. *Proc of MICCAI* 2004, Vol. II, pp. 319-327.

[2] Bloom MB, Rawn CL, Salzberg AD, Krummel TM: Virtual reality applied to procedural testing: the next era. *Annals of Surgery* 2003, 237(3): 442-448.

[3] Sewell C, Morris D, Blevins N, Barbagli F, Salisbury K: Quantifying risky behavior in surgical simulation. *Proc of MMVR* 2005, pp. 451-457.

[4] Cohen JD, Lin MC, Manocha D, Ponamgi MK: I-COLLIDE: an interactive and exact collision detection system for large-scale environments. *Proc. ACM Interactive 3D Graphics Conf.*, 1995, pp. 189-196.

[5] Dijkstra EW: A note on two problems in connexion with graphs.*Numerische Mathematik*, 1959 1: 269-71

[6] Jones MW: 3D Distance from a point to a triangle. Technical Report, Department of Computer Science, University of Wales Swansea, 1995.