GPU-Accelerated Interactive Visualization and Planning of Neurosurgical Interventions

Mario Rincón-Nigro, Nikhil V. Navkar, Nikolaos V. Tsekos, and Zhigang Deng ■ *University of Houston*

A proposed GPU-accelerated method enables interactive quantitative estimation of the risk associated with neurosurgical access paths. It exploits spatially accelerated data structures and efficient implementation of algorithms on GPUs. In evaluations, the method achieved interactive rates, even for high-resolution meshes.

Using preoperative imaging to plan interventional procedures has led to new, effective interventional paradigms. Neurosurgical procedures that benefit from such planning include deep brain stimulation, biopsies, and shunting and insertion of external ventricular drains. Planning often entails processing multicontrast and multimodal imaging (magnetic resonance imaging and computed tomography) to map anatomical and pathophysiologic features. A common practice is manual selection of different entrance positions on the patient's scalp and assessment of their suitability, to determine an appropriate insertion path.

To avoid errors due to discrepancies between preoperative and intraoperative images, the trend is to move as much of the planning as possible into the operating room. Toward that end, we propose a semiautomated GPU-accelerated method to process, visualize, and plan interventions at interactive or nearly real-time speed. It has two main components:

- It embeds the geometrical structures representing critical-tissue areas pertinent to the procedure in spatial data structures. This speeds up computation of the geometric queries involved in estimating the risk for paths.
- It implements computation on GPUs, which exploits the problem's parallel nature while effectively handling the involved irregular workload.

Evaluations have demonstrated that our method is robust and interactive and can generate safer straight access paths. Figure 1 diagrams the method.

To the best of our knowledge, our method is the first to enable interactive estimation and visualization of risk in straight-access neurosurgical interventions with mesh-based tissue representation. Owing to its high speed, the operator can interactively explore a much larger number of possible paths and target points. This ability enhances the effectiveness and efficiency of decision making in interventions.

Applying Our Method

The operator first selects a target point in a predefined region—for example, in the tumor's core or on its surface. He or she then interactively visualizes the resulting risk map to guide selection of an optimal insertion point. Figure 2a illustrates this process.

After selecting a target point and an insertion point, the operator can view the risk map for the advancing needle's current position (see Figure 2b). Such dynamic risk maps can significantly facilitate decision making. When selecting a path, the operator usually further analyzes it by looking at imaging planes (magnetic-resonance-imaging slices) orthogonal to it (also called a bird's-eye view). For stereotactic robot-assisted neurosurgical interventions, such as in the NeuroArm and

 Published by the IEEE Computer Society

neuromate systems, users can control the needle's velocity during insertion on the basis of the risk function along the path. The horizontal bar in Figure 2b indicates the current risk at each point in the path.

Our method computes each risk value by setting the needle tip as the current target. The risk along the path increases nonlinearly as the needle advances and reaches the maximum at the final target position. If the operator knows the environment changes (when the needle is near critical tissue, and so on), he or she can adjust the needle's velocity to advance more cautiously. As Figure 2b shows, this method's interactive speed lets the operator see the exact risk from the surrounding tissue when inserting the needle by simply placing the target point along its tip and recomputing the risk map at each step.

Creating Risk Maps

Selecting safe straight access paths entails minimizing their risk functions. We define this function on the basis of three criteria:

- Paths can't intersect any critical tissue; otherwise, unacceptable patient injury might occur.
- Paths should be as distant as possible from critical tissue.
- Paths should be as short as possible.

Generally, additional selection criteria, based on the particular procedure's needs, can be seamlessly incorporated into this function.

Inspired by Nikhil Navkar and his colleagues' research,[1] we represent structures segmented from imaging as triangular meshes. Specifically, the scalp surface is a triangular mesh, $m_s = (V_S, T_S)$, and critical tissue is another triangular mesh, $m_c = (V_C, T_C)$, where V is the set of vertices and T is the set of triangles. The candidate paths are line segments extending from the vertices of m_s to a predefined target point p. To obtain the set of permissible paths, we discard the paths that intersect any triangle in m_c, which are unsafe.

Then, the risk for a particular path (in the set of the permissible paths) starting at vertex $v \in V_S$ is

$$Risk(v,p) = k_1 \|v-p\| - k_2 \min_{v' \in V_C} d\left(\overline{vp}, v'\right), \quad (1)$$

$$d\left(\overline{vp}, v'\right) = \begin{cases} \|v'-p\| & \text{if } (v-p)\cdot(v'-p) < 0 \\ \|v'-v\| & \text{if } (p-v)\cdot(v'-v) < 0 \\ \dfrac{\|(v-p)\times(p-v')\|}{\|v-p\|} & \text{otherwise} \end{cases},$$

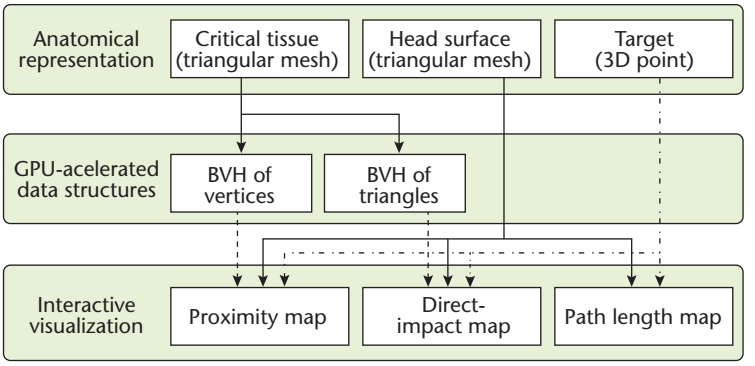


Figure 1. A schematic view of our method. The operator can interactively visualize the risk associated to particular paths for surgical interventions and explore the effect of changing the target point's position. BVH stands for bounding volume hierarchy.

where $k_1 \geq 0$ and $k_2 \geq 0$ are user-specified weighting parameters and d is a distance function. (Figure 3 visualizes the effect of varying k_1 and k_2). For the risk maps in this article, we used $k_1 = 0.1$ and $k_2 = 0.9$, unless otherwise noted. The first term in Equation 1 accounts for the path's total length; the second term accounts for the path's proximity to critical tissue. So, larger values of k_1 enforce the selection of short paths, whereas larger values of k_2 enforce the selection of paths farther from critical tissue.

Brute-force computation of the set of permissible paths or the risk for every path is computationally expensive, even for low-resolution meshes. To efficiently perform the computation without sacrificing precision, we build *acceleration spatial data structures* from the critical-tissue mesh's geometric primitives. An efficient GPU implementation of our method further enables the interactive computation of risk maps, even for high-resolution meshes.

Acceleration Spatial Data Structures

Specifically, we use *bounding volume hierarchies* (BVHs)[2] because they are inexpensive to compute and can be quickly restructured to support meshes that can undergo dynamic deformation. However, in this article, we consider only static meshes, not general dynamic meshes. A BVH partitions geometric objects into a hierarchy that can be used to accelerate geometric queries such as those in our problem. Formally, BVHs are binary trees in which each node represents a group of objects and the region they occupy. Children nodes are recursive partitions of the group of objects in their parent nodes.

Figures 4a and 4b show 2D examples of BVHs; Figure 4c shows bounding boxes for a 3D mesh (down to a depth of 3). For m_c, we construct two BVHs: one holds its triangles and the other holds its vertices.

Related Work in Surgical Planning

To process, visualize, and manipulate large volumes of 3D imaging datasets, researchers have proposed automated surgical-planning methods.[1–6] Regarding the incorporation of planning into the operating room (intraoperative planning), researchers are investigating techniques for integrating magnetic resonance imaging, computed tomography, and surgical procedures.[7,8]

Recently, Paul Herghelegiu and his colleagues proposed a multilevel framework for planning biopsies of deep-seated tumors.[9] Their system assists planning at all stages, from selecting the target point to selecting an entry point that minimizes the risk of hitting critical tissue. After selecting the target point, the neurosurgeon defines a set of regions on the skull's surface. The system then displays a color-coded map that defines the risk for the paths in those regions. The system also helps the neurosurgeon validate a path by displaying information about areas in it requiring closer inspection—specifically, where it's near the brain vasculature. To help the neurosurgeon, the system displays a set of 2D slices and volume-rendered 3D views.

All these approaches find the optimal path either automatically[4] or by assisting the neurosurgeon's decision making.[5,6,9] State-of-the-art approaches use operator-defined criteria to project color-coded images of the brain's internal structures on the scalp. The input includes a geometrical representation of critical tissue (for example, meshes[2] or segmented volumetric data[3,10]) that the path shouldn't traverse. The operator must select the target point carefully and precisely because the algorithms that compute optimized paths generally take a long time.[2]

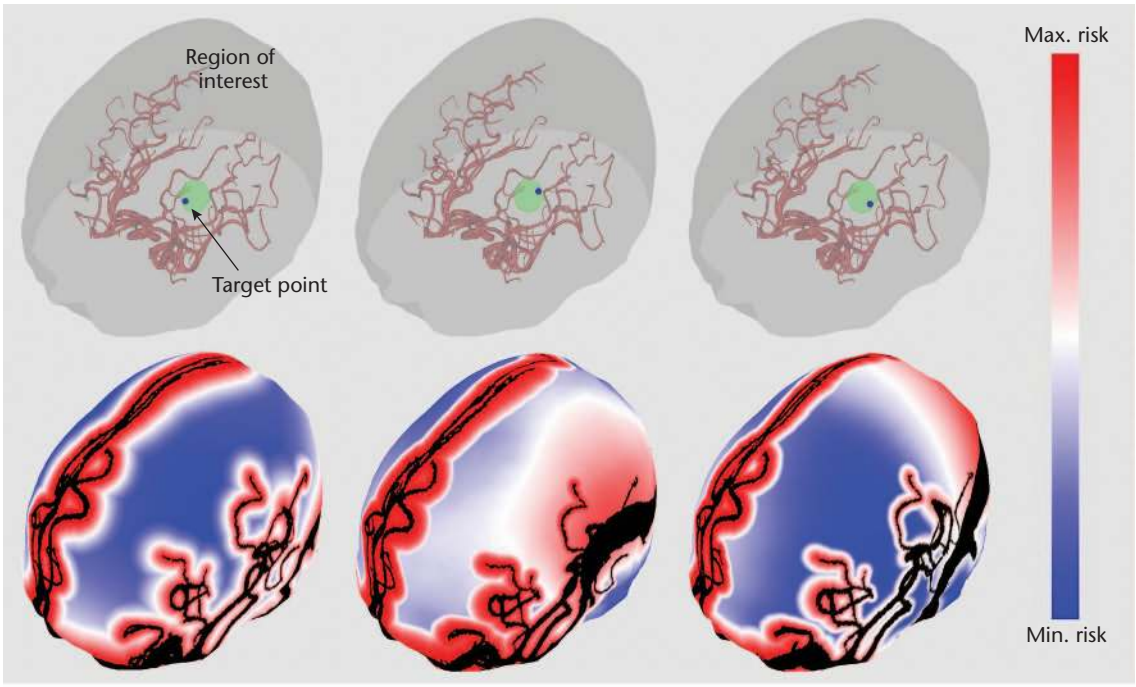
Moreover, in most cases the target isn't a single point but a 3D structure or surface. So, accessing all the possible target points in or on the region of interest requires orders of magnitude longer processing time than a single target point.[2,3] The ability to perform this process in or near real time allows interactive visualization and planning. This will significantly reduce planning time and thus allow effective intraoperative replanning of interventions. This is our research's goal (see the main article).

References

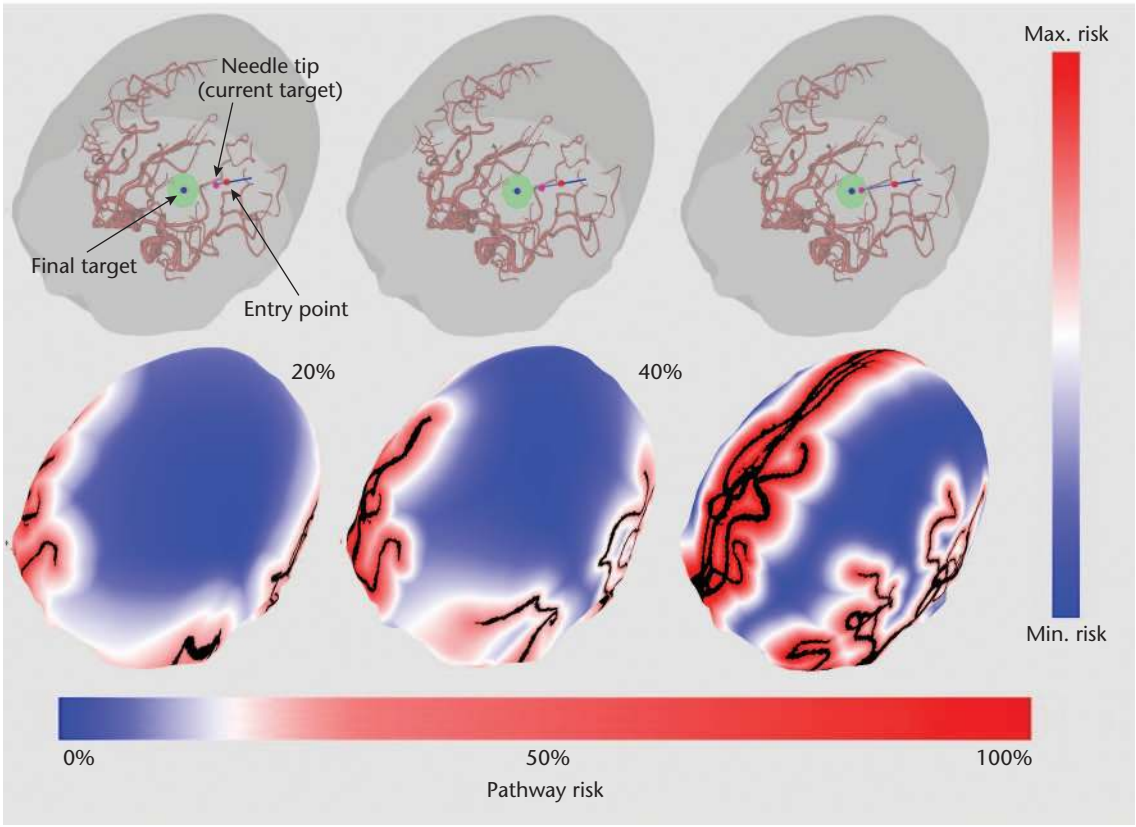
1. E.J.L. Brunenberg et al., "Automatic Trajectory Planning for Deep Brain Stimulation: A Feasibility Study," *Proc. 10th Int'l Conf. Medical Image Computing and Computer-Assisted Intervention* (MICCAI 07), vol. 1, Springer, 2007, pp. 584–592.
2. N.V. Navkar et al., "Visualization and Planning of Neurosurgical Interventions with Straight Access," *Proc. Int'l Conf. Information Processing in Computer-Assisted Interventions* (IPCAI 10), Springer, 2010, pp. 1–11.
3. R.R. Shamir et al., "A Method for Planning Safe Trajectories in Image-Guided Keyhole Neurosurgery," *Proc. Medical Image Computing and Computer-Assisted Intervention—MICCAI 2010*, Springer, 2010, pp. 457–464.
4. S. Bériault et al., "Automatic Trajectory Planning of DBS Neurosurgery from Multimodal MRI Datasets," *Proc. Medical Image Computing and Computer-Assisted Intervention—MICCAI 2011*, Springer, 2011, pp. 259–266.
5. C. Essert, C. Haegelen, and P. Jannin, "Automatic Computation of Electrodes Trajectory for Deep Brain Stimulation," *Medical Imaging and Augmented Reality*, Springer, 2010, pp. 149–158.
6. C. Essert et al., "Automatic Computation of Electrode Trajectories for Deep Brain Stimulation: A Hybrid Symbolic and Numerical Approach," *Int'l J. Computer Assisted Radiology and Surgery*, vol. 7, no. 4, 2011, pp. 517–532.
7. A.C.F. Colchester et al., "Development and Preliminary Evaluation of VISLAN, a Surgical Planning and Guidance System Using Intraoperative Video Imaging," *Medical Image Analysis*, vol. 1, no. 1, 1996, pp. 73–90.
8. R. Viard et al., "Needle Positioning in Interventional MRI Procedure: Real Time Optical Localisation and Accordance with the Roadmap," *Proc. 29th Ann. Int'l Conf. IEEE Eng. in Medicine and Biology Soc.*, IEEE, 2007, pp. 2748–2751.
9. P.C. Herghelegiu et al., "Biopsy Planner—Visual Analysis for Needle Pathway Planning in Deep Seated Brain Tumor Biopsy," *Computer Graphics Forum*, vol. 31, no. 3, 2012, pp. 1085–1094.
10. R. Khlebnikov et al., "Crepuscular Rays for Tumor Accessibility Planning," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 12, 2011, pp. 2163–2172.

A geometric query over a BVH traverses the data structure. For the triangle BVH, with which we compute the set of permissible paths, we must figure out whether each path intersects a triangle in m_c. Starting from the root, at each step of the traversal we perform an intersection test against the children nodes. If a node is intersected, the traversal must continue recursively over the substructure for which the node is the root. Otherwise, we can safely ignore the substructure because the path can't intersect any of its triangles. Because the leaf nodes contain triangles, when the traversal reaches a leaf node, it must test whether the triangles intersect the path. The traversal stops when it finds an intersected triangle or when it has tested all the triangles in the intersected nodes.

Figure 4a illustrates this process. The traversal first tests box AABB. Because the path intersects this box, the traversal tests its two children ($AABB_1$ and $AABB_2$). The path intersects $AABB_1$, so both its children must be tested. The path doesn't intersect $AABB_{1,1}$ (we can safely discard going farther down that node) but intersects $AABB_{1,2}$. Because $AABB_{1,2}$ is a leaf node, its geometric primitives must be tested for intersection. Regarding $AABB_2$, we need to test only its child $AABB_{2,1}$.



(a)



(b)

Figure 2. Risk maps, with the computed risk values normalized to the range [0, 1]. Blue, white, and red represent low, medium, and high risk; black depicts impermissible insertion areas. (a) Interactive planning by moving the target point within a region. (b) Moving the target point along a selected path. Our method computed the risk maps by placing the current target at the needle's tip. The horizontal bar shows the normalized risk as a function of the percentage of the covered path (that is, 50 percent means the needle is at half of the total length of the selected safe path). Operators can plan the insertion velocity according to the risk function along the path.

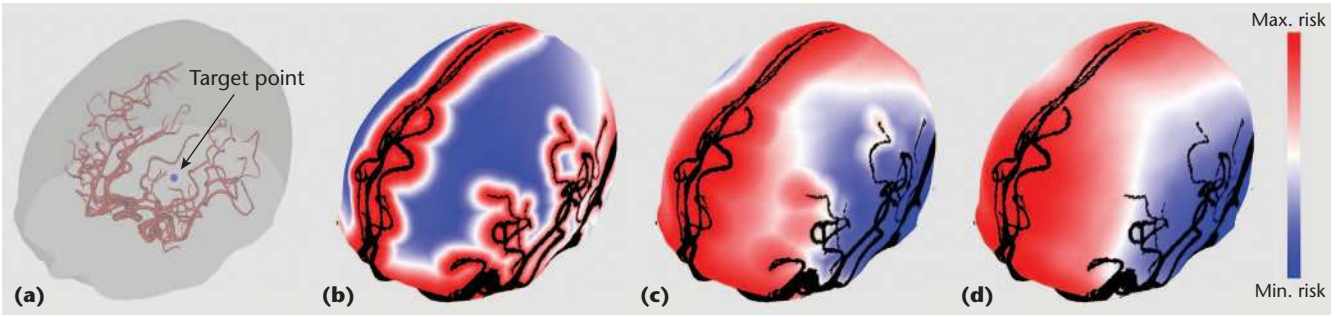


Figure 3. Risk maps for pairs of user-specified weights k_1 and k_2 with different values. (a) The target point's position. (b) Enforcing a large distance between the path and critical tissue: $k_1 = 0.1$ and $k_2 = 0.9$. (c) Enforcing a balance between the path length and the distance from critical tissue: $k_1 = 0.5$ and $k_2 = 0.5$. (d) Enforcing a short path: $k_1 = 0.9$ and $k_2 = 0.1$.

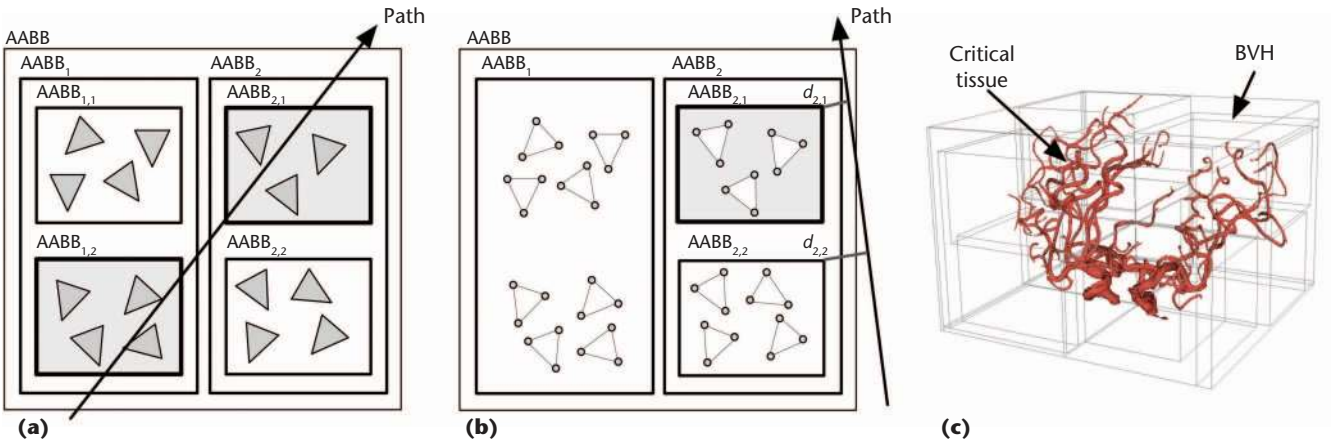


Figure 4. BVHs: (a) a BVH of triangles, (b) a BVH of vertices, and (c) bounding boxes for a 3D mesh (down to a depth of 3). A BVH partitions geometric objects into a hierarchy that can be used to accelerate geometric queries. In Figure 4b, d is the distance from a closest vertex of a geometric object to the path.

Computation of the proximity term (the second term in Equation 1) occurs similarly over the vertex BVH. Here, for each path we determine the vertex (in m_c) closest to the path. Figure 4b illustrates a query over this BVH. For each path, we determine which child of a node is closer and which one is farther. Generally, the closer child will more likely contain the closest vertex. If the closest vertex found in that node is closer to the path than any point in the farther child's bounding box, we can safely discard the farthest node. In our example, node $AABB_2$ is closer, so the traversal occurs recursively over it. In $AABB_2$, $AABB_{2,1}$ is closer and is a leaf node (containing vertex primitives), so we've found the vertex in this node that's closest to our path. Because this vertex is closer than the closest point in $AABB_{2,2}$, we can discard traversals through that node. We similarly discard traversals through $AABB_1$.

Our GPU-Based Parallel Implementation

Because the computation for each path is independent from that for other paths, the task is ideal for massively parallel processors such as GPUs. However, achieving high performance on GPUs isn't straightforward.

A GPU consists of streaming multiprocessors (SMs). Streaming programming subdivides grids of execution threads into blocks of threads. It statically assigns different blocks to execute on different SMs. That is, a block assigned to one SM can't execute on another SM. An assigned block's threads are further subdivided into warps—groups of 32 consecutive threads. A warp's threads execute in parallel in the SM in SIMT (single instruction, multiple threads) mode, with each thread executing the same instruction at the same time.

To compute the risk map, a naive scheme of assigning work to processing units in CUDA (Compute Unified Device Architecture) or OpenCL would correspond to executing a grid with as many threads as paths. So, each thread's computation would occur individually. The resulting high variability of the computations for the different paths would lead to resource underutilization. The workloads assigned to some blocks would need more time than those assigned to other blocks, leading to some SMs being idle for part of the computa-

tion time. The different tasks' different processing times would create a high workload imbalance that degrades performance.[3]

To improve performance, we balance the workload in the GPU using a centralized queue.[3] The paths to compute are on a single queue implemented as an array in the global memory. The queue's head is also in the global memory and is visible to every thread in the grid. Instead of assigning the computation of a path to a single thread, we make long-running warps to retrieve batches of paths (tasks) from the global queue. These warps finish executing only when all the paths in the global queue have been processed.

You can appreciate this approach by considering the warp process. When a warp launches, the first thread takes the next task by updating the queue's head through atomic addition (it must be atomic to avoid race conditions). Then, each thread in the warp processes a path in the task. When a warp completes this batch, it takes another task or finishes execution if the queue is empty. The processing of each path first traverses the triangle BVH to determine whether the path intersects critical tissue and then traverses the vertex BVH to compute the proximity term. With the centralized queue, no SM will be idle, as long as tasks still need processing.

Evaluating Our Method

Our evaluation consisted of quantitative experiments, a user study, and feedback from neurosurgeons.

Computational Performance

To evaluate our method's efficiency (to what degree it can be used interactively), we measured the time to compute risk maps for two types of meshes with varying resolutions:

- For the scalp surface, the number of triangles ranged from 2k to 276k, and the number of vertices ranged from 4k to 553k.
- For critical tissue, the number of triangles ranged from 2k to 276k, and the number of vertices ranged from 46k to 300k.

The number of vertices on the scalp surface mesh determines the number of paths to process.

We implemented our method on a GPU using CUDA and recorded the timing using CUDA time events. The recorded time included both the CUDA kernels' execution time and the communication overhead due to data transfer between the host and device. It started when the BVH data (and scalp surface mesh data) were transferred to the GPU. It ended when the results of the computation of the path length and proximity term were transferred to the host's main memory. We tested our implementation on a mainstream desktop computer featuring an Intel Core i7 processor, an Nvidia GeForce GTX 480 GPU, and 4 Gbytes of main memory.

Figure 5 reports the computation time. Even for high-resolution meshes, our method achieved interactive rates and was scalable. The computation time was approximately linear to the scalp surface mesh's size because it depended on the number of paths to be processed. Indeed, as the mesh resolution increased, the number of paths increased linearly. In contrast, the computation time demonstrated a logarithmic dependence on the critical-tissue mesh's size.

Comparison with a CPU-Based Implementation

We compared our GPU-accelerated implementation with a CPU-based baseline implementation. The baseline implementation computed risk maps by following the approach we described before; it didn't use the CPU's SIMD (single instruction, multiple data) capability. We measured both implementations' computation time while varying the critical-tissue mesh's size and the number of paths.

Our GPU-accelerated implementation achieved a speedup of two orders or magnitudes over the baseline implementation (see Figure 6). Interestingly, it also scaled better with the critical-tissue mesh's size and the number of paths. That is, although both implementations were algorithmically equivalent, our implementation's speedup increased with the problem's size.

Performance with the Centralized Queue

A GPU implementation using the centralized queue had a speedup of up to 1.4× (29 percent less computation time) compared to a baseline GPU implementation (see Figure 7). The baseline implementation employed a grid with as many threads as paths. As Figure 7 shows, the speedup increased with the critical-structure mesh's size but wasn't significantly sensitive to the number of paths. This is because when the mesh size increased, the application became memory bounded; that is, memory operations dominated computation.

Comparison with a Voxel-Based Method

We also compared our method to a fast voxel-based method for computing risk maps on the CPU.[4] For both methods, we varied the input structure size and the number of paths.

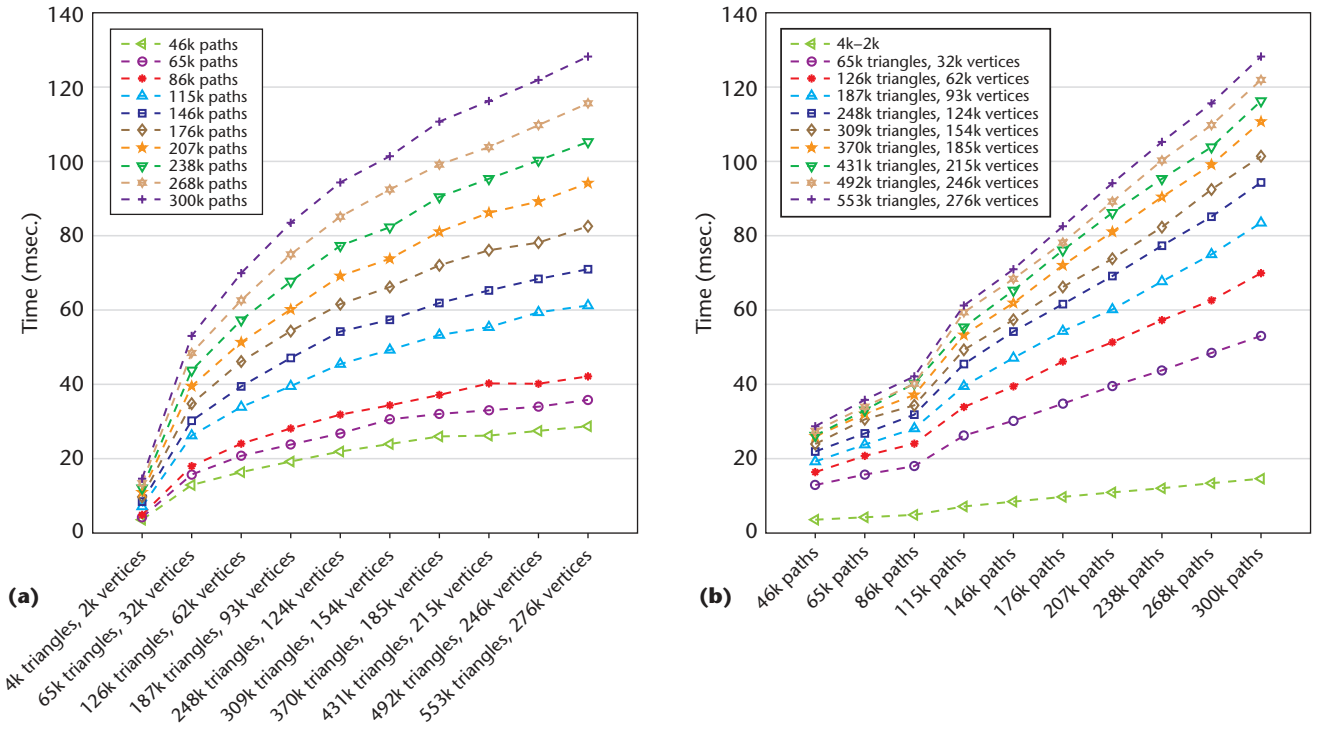


Figure 5. Computation times for generating risk maps, with meshes given as the number of triangles minus the number of vertices. (a) Critical-tissue mesh resolution. (b) The number of computed paths. Even for high-resolution meshes, our method achieved interactive rates and was scalable.

For the voxel-based method, the input structure was volumetric data containing precomputed risk information. Each cell of the 3D grid contained an estimate of the risk for traversing the cell. This method then computed the total risk for each path as the sum of the risks of all the voxels intersected by the needle.

Because the two methods' input structures differed fundamentally, we made the following assumption. The critical-tissue mesh and volumetric data are equivalent in size if the mesh is the isosurface reconstructed from a segmented 3D image with the same resolution as the volumetric data, using the classic marching-cubes algorithm.

Our mesh-based method consistently performed better for all the input structure sizes and numbers of paths (see Figure 8). It also scaled better with the problem size. Although both methods' computation times were approximately linear with the number of paths, the voxel-based approach grew linearly with the input structure size. This led to it quickly losing an interactive rate (see Figure 8).

User Study

Our user study quantitatively assessed how interactive computation of risk maps helps operators plan straight-access neurosurgical interventions (see Figure 9). Specifically, we explored how our method might help intraoperative target repositioning. We compared the safety, in terms of length and proximity, between paths planned using visually guided target position selection and paths selected through risk-map-guided target positioning. The visually guided method stands for any method requiring planning outside the operating room owing to computation time limitations.[1] The risk-map-guided method stands for any method that enables intraoperative replanning.

The seven subjects were computer science graduate students whose average age was 26.4 years. Before the experiment, we explained the concept of risk maps and the tasks the subjects would perform. These tasks were to select the insertion point on the scalp surface and the target point in a given region of interest.

The experiment comprised 12 planning tasks: six for each of the two methods. We defined two regions of interest (see Figure 9d). For each region, each of the three tasks required planning the best insertion path using one of three sets of risk-map weights. The sets were $w_1 \equiv (k_1 = 0.1, k_2 = 0.9)$, $w_2 \equiv (k_1 = 0.5, k_2 = 0.5)$, and $w_3 \equiv (k_1 = 0.9, k_2 = 0.1)$.

For the visually guided method, the subjects had to find the best path they could by selecting an insertion point and freely positioning the target without risk maps. That is, they visually inspected and navigated the rendered models of the head's surface and blood vessels. For each selected path, the system presented information about its length and proximity to blood vessels. After subjects se-

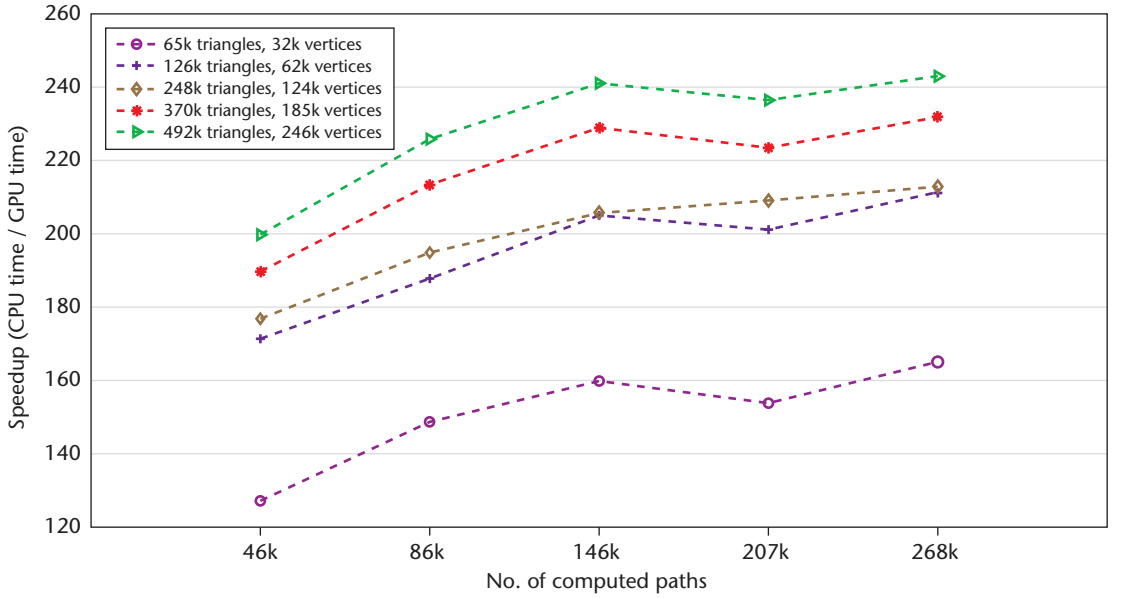


Figure 6. Comparing the GPU-accelerated and CPU baseline implementations in terms of speedup (CPU time / GPU time. The GPU-accelerated implementation achieved a speedup of two orders or magnitudes over the baseline implementation.

lected a satisfactory target point, the computer displayed a full color-coded risk map; they could reposition the entry point if they desired.

For the risk-map-guided method, as the subjects freely moved the target within the region of interest, the system interactively computed and displayed full risk maps. Additionally, for each target, the subjects could view a coarse 56-point grid covering the region of interest that suggested safe areas for placing the target. When subjects selected a target position, the system displayed the risk of the safest path it could find. To guide target repositioning, the subjects could request a normalized color-coded visualization of the minimum risks. Such visualization required computing 56 risk maps, which took less than two seconds. This computation only needed to be performed once for a given region of interest.

We recorded the planned paths' lengths and proximity values (see Figure 10). A repeated-measures t-test showed statistically significant reduced length for every pair of weights (for w_1, $t = 2.898$ and $p = 0.012$; for w_2, $t = 3.939$ and $p = 0.001$; for w_3, $t = 3.738$ and $p = 0.002$) and for w_1 (for w_1, $t = -5.624$ and $p = 0.00008$; for w_2, $t = -0.766$ and $p = 0.457$; for w_3, $t = -1.317$ and $p = 0.21$).

For w_2 and w_3, we observed no evidence of improvement in terms of proximity. This is because w_2 and w_3 penalized the selection of long paths and because most paths tended to be longer than the distance to the closest vessels.

Neurosurgeon Evaluation

To assess how neurosurgeons can benefit from interactive visualization and planning in the operating room, we consulted two experienced neurosurgeons. We explained our method and the computations it can perform interactively.

The neurosurgeons then completed a questionnaire. In it, they rated from 1 to 6 (1 = least useful and 6 = most useful) three interactive features:

- preoperative risk-map-guided target repositioning,

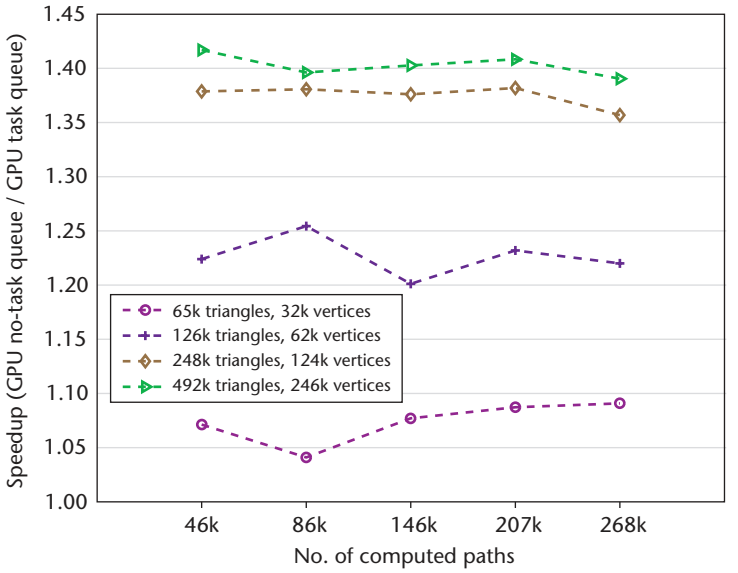


Figure 7. Comparing a GPU implementation with and without a task queue in terms of speedup (GPU no-queue time / GPU queue time). The implementation with the queue had a speedup of up to 1.4× (29 percent less computation time).

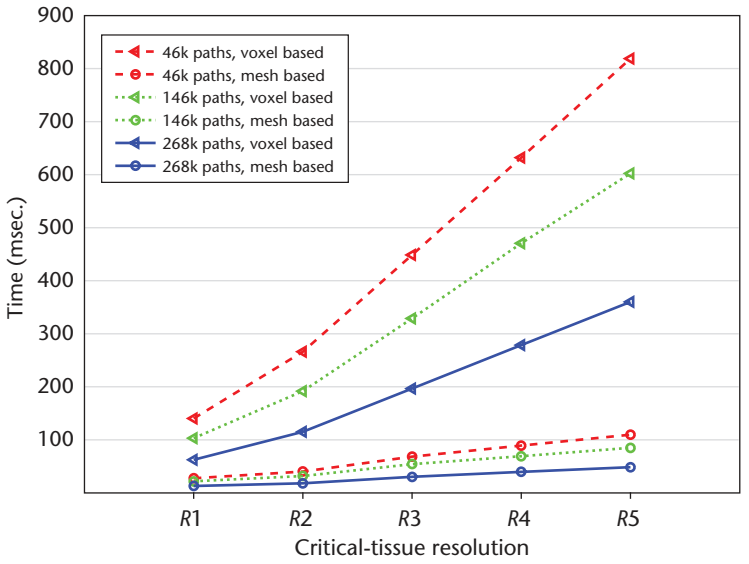


Figure 8. The computation time for our mesh-based approach and a voxel-based approach.[4] R_1 is a mesh with 32k vertices and 65k triangles (for our approach) or a 288 × 172 × 136 grid (for the voxel-based approach). R_2 is a mesh with 62k vertices and 126k triangles or a 576 × 172 × 136 grid. R_3 is a mesh with 124k vertices and 248k triangles or a 576 × 460 × 136 grid. R_4 is a mesh with 185k vertices and 370k triangles or a 576 × 768 × 136 grid. R_5 is a mesh with 246k vertices and 492k triangles or a 576 × 768 × 272 grid.

- intraoperative risk-map-guided target repositioning, and
- automatic suggestion of the safest paths.

They also identified which surgical procedures could benefit from our method. In addition, they could provide free-form feedback and suggestions.

The neurosurgeons identified interactive preoperative and intraoperative target repositioning as highly beneficial (average score: 5). Although preoperative planning is necessary, they considered it time-consuming. So, they would appreciate any tool that can speed up the process and provide information for exploring and validating a plan. They felt that intraoperative target repositioning was particularly desirable (average score: 6) for procedures involving tissue shifts (for example, tumor resection in which the brain shifts and changes form after the surgeon opens the dura). Integration between the planning-assistance software, image acquisition mechanisms, and plan-execution tool is the key in the intraoperative scenario.

However, the neurosurgeons disliked automatic suggestion of the safest path (average score: 2) because it takes away responsibility from the surgeon. They suggested that, instead of showing a global safest path, the system should display the risk maps over reduced areas of the scalp such as the area intersected by a cone with its axis along the global safest path and its tip at the target point. This would reduce the planning space while still allowing the procedure to benefit from the surgeon's experience and good judgment.

The procedures that could benefit from interactive planning included deep brain stimulation for electrode placement, shunt placement, small-tumor resection, brain cyst resection, aneurysm treatment, and treatment of arteriovenous malformations. For these procedures, the neurosurgeons suggested that the risk maps include information on functional regions of the brain, which are more important in selecting paths than the position of blood vessels (which can be slightly retracted during surgery). Our method could include these regions as additional critical tissue that should be avoided or whose intersection would increase the paths' risk.

Our method could also improve procedures using the gamma knife, which applies radiation over tumors. Precise planning of radiation trajectories that minimize the exposure of neighboring tissue requires heavy computation and could thus benefit from our method's performance.

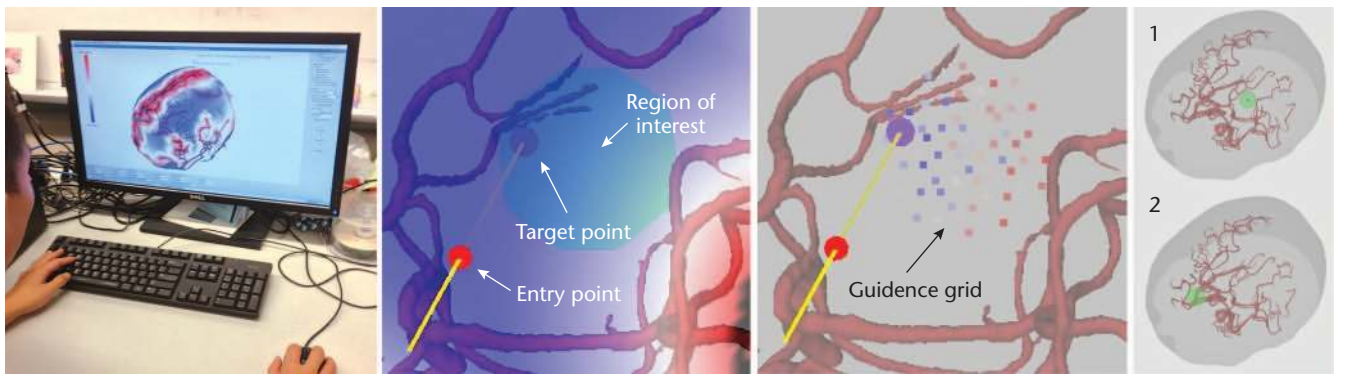


Figure 9. The user study. (a) A snapshot of the setup. (b) A zoomed-in view of a region of interest and selected path. (c) The corresponding visualization of the grid for risk-map-guided target positioning. (d) The two regions of interest.

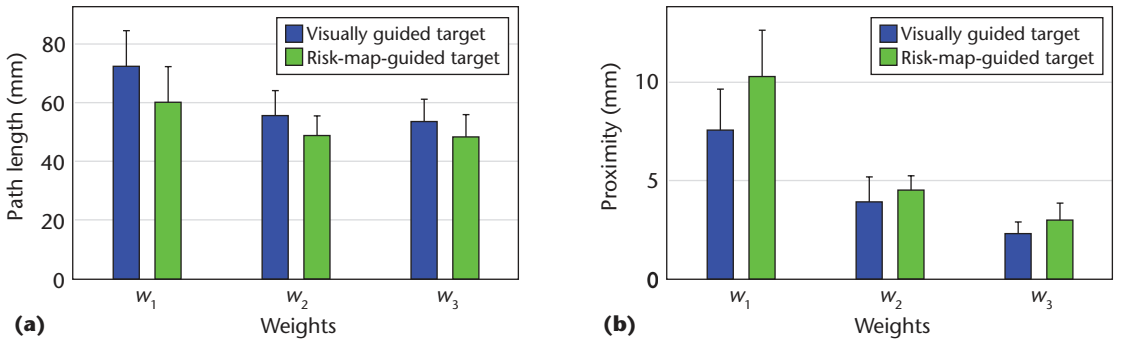


Figure 10. The (a) lengths and (b) proximity of the selected paths using conventional visually guided target positioning and risk-map-guided target positioning, over three sets of weights ($w_1 \equiv (k_1 = 0.1, k_2 = 0.9)$; $w_2 \equiv (k_1 = 0.5, k_2 = 0.5)$; $w_3 \equiv (k_1 = 0.9, k_2 = 0.1)$). For visually guided positioning, subjects placed the target by navigating through a 3D visualization of the head surface, critical tissue, and region of interest. For risk-map-guided positioning, subjects had an augmented view interactively showing how target repositioning affected the risk maps and a risk grid providing information about safe regions for positioning the target.

A high-performance method such as ours could benefit existing surgical-planning approaches (for example, Paul Herghelegiu and his colleagues' multilevel planning framework; see the sidebar) because it will enable intraoperative planning. In addition, the computational power might be beneficial in preoperative scenarios, letting surgeons explore and visualize much larger planning spaces while avoiding possible complications in the procedure. For example, planning could include additional information and planning constraints with no large increase in computation time. (For instance, surgeons could include additional critical tissue such as functional brain regions and consider the tool's angle of intersection with the targeted regions.)

We plan to further investigate efficient methods for procedures on structures that change dynamically owing to breathing, heartbeats, or the interventional procedure. We could also extend our current research to access paths that aren't straight.

Acknowledgments

US National Science Foundation grants NSF IIS-0914965 and CNS-0932272 and research gifts from Google and Nokia partly supported this research. Any opinions, findings, conclusions, or recommendations expressed in this article are the authors' and don't necessarily reflect the agencies' views.

References

1. N.V. Navkar et al., "Visualization and Planning of Neurosurgical Interventions with Straight Access," *Proc. Int'l Conf. Information Processing in Computer-Assisted Interventions* (IPCAI 10), Springer, 2010, pp. 1–11.
2. J. Arvo and D. Kirk, *A Survey of Ray Tracing Acceleration Techniques*, Academic Press, 1989, pp. 201–262.
3. T. Aila and S. Laine, "Understanding the Efficiency of Ray Traversal on GPUs," *Proc. 2009 ACM Siggraph / Eurographics Conf. High Performance Graphics*, ACM, 2009, pp. 145–149.
4. R.R. Shamir et al., "A Method for Planning Safe Trajectories in Image-Guided Keyhole Neurosurgery," *Proc. Medical Image Computing and Computer-Assisted Intervention—MICCAI 2010*, Springer, 2010, pp. 457–464.

Mario Rincón-Nigro *is a PhD student in the University of Houston's Department of Computer Science. His research interests include computer graphics, GPU computing, and high-performance graphics algorithms and systems. Rincón-Nigro received an MS in computer science from the University of Houston. Contact him at mrinconnigro2@uh.edu.*

Nikhil V. Navkar *is a PhD candidate in the University of Houston's Department of Computer Science. His research interests include image-guided interventions and medical robotics. Navkar received an MS in computer science from the University of Houston. Contact him at nvnavkar@cs.uh.edu.*

Nikolaos V. Tsekos *is an associate professor of computer science at the University of Houston. His research interests include MRI (magnetic resonance imaging) and MRI-compatible robotic manipulators. Tsekos received a PhD in biophysical sciences and medical physics from the University of Minnesota. Contact him at nteskos@cs.uh.edu.*

Zhigang Deng *is an associate professor of computer science at the University of Houston. His research interests include computer graphics, computer animation, and human-computer interaction. Deng received a PhD in computer science from the University of Southern California. He's a senior member of IEEE and a member of ACM. Contact him at zdeng@cs.uh.edu.*