GPU-Accelerated Interactive Visualization and Planning of Neurosurgical Interventions

Mario Rincón-Nigro, Nikhil V. Navkar, Nikolaos V. Tsekos, and Zhigang Deng, *Senior Member, IEEE*

Abstract—Advances in computational methods and hardware platforms provide efficient processing of medical imaging data sets for surgical planning. In the case of neurosurgical interventions that are performed via a straight access path, planning entails selecting a pathway, from the scalp surface to the targeted area, that is of minimal risk to the patient. We propose a GPU-accelerated approach to enable quantitative estimation of the risk associated with a particular access path at interactive rates. It heavily exploits spatially accelerated data structures and efficient implementation of algorithms on GPUs. We evaluate the computational efficiency and scalability of the proposed approach through extensive performance comparisons, and show that interactive rates can be achieved even for high-resolution meshes. Through a user study, and feedback obtained from domain experts, we identify some of the potential benefits that our high-speed approach offers for pre-operative planning and intra-operative replanning of straight access neurosurgical interventions.

Index Terms—GPU Acceleration, Neurosurgical Interventions, Risk Maps, and Straight Access

1 Introduction

Planning of interventional procedures from pre-operative imaging has led to the development of new effective interventional paradigms. Among the procedures that benefit from such pre-operative planning are neurosurgical interventions such as deep brain stimulation, biopsies, and shunt and insertion of external ventricular drains (EVD). Planning, often, entails processing of multi-contrast and multi-modal imaging (MRI/CT) to map anatomical and pathophysiologic features. A common practice is manual selection and assessment of the suitability of different entrance positions on the scalp surface of the patient in order to select an appropriate insertion path. To process, visualize and manipulate large volumes of 3D imaging datasets, automated surgical planning methods have been proposed in recent years [1], [2], [3], [4], [5], [6]. The current trend is to move as much of the planning tasks as possible into the operating room to avoid errors that can appear because of the discrepancies between pre-operative images and intra-operative images. In this context, techniques for integrating MRI, CT imaging, and surgical procedures are being actively investigated [7], [8].

Recently, Herghelegiu et al. [9] proposed a multi-level framework for planning biopsies of deep-seated tumors. The system is meant to assist planning at all stages from the selection of the target point within tumors, to the selection of a safe entry point that minimizes the risk of hit with vital structures. After selection of the target point, the neurosurgeon defines a set of regions on the surface of the skull, and the system displays a color-coded map that defines the risk associated with the paths within the region. The system also assists the neurosurgeon in the validation of the path by displaying information about regions in the path that require closer inspection, specifically, the regions in which the path is closer to any point in the brain vasculature. In terms of visualization, the system displays a set of 2D slices and volume rendered 3D views to assist the neurosurgeon in the planning and validation of the selected pathway.

All the aforementioned works describe approaches to find the optimal path for neurosurgical interventions either automatically [4] or by assisting the decision-making process of the neurosurgeon [5], [6], [9]. State-of-the-art approaches use criteria defined by the operator to generate color-coded projections of internal structures of the brain on the scalp surface. The input includes a geometrical representation of vital tissue (e.g., meshes [2] or segmented volumetric data [3], [10]) that should not be traversed by the path from a point on the scalp surface through which the operator may insert a surgical tool to the target point in the region of interest. With those approaches, the operator needs to select the target point carefully and precisely since the time required by the algorithms for computing optimized paths is generally long (e.g., [2]). Moreover, in most cases the target (i.e., the vital structure) is not a single point but rather a three dimensional (3D) structure or surface. Therefore, accessing all different possible targeted points within or on the region of interest requires orders of magnitude longer processing time than the single target point case [3], [2]. The possibility of performing this

- *M. Rincón, N. Navkar, and Z. Deng are with the Computer Graphics and Interactive Media Lab and the Department of Computer Science, University of Houston, Houston, TX 77204-3010.*
- *N. Tsekos is with the Medical Robotics Lab and the Department of Computer Science, University of Houston, Houston, TX 77204-3010.*
- *E-mails: mrinconnigro2@uh.edu, nvnavkar|nteskos|zdeng@cs.uh.edu.*

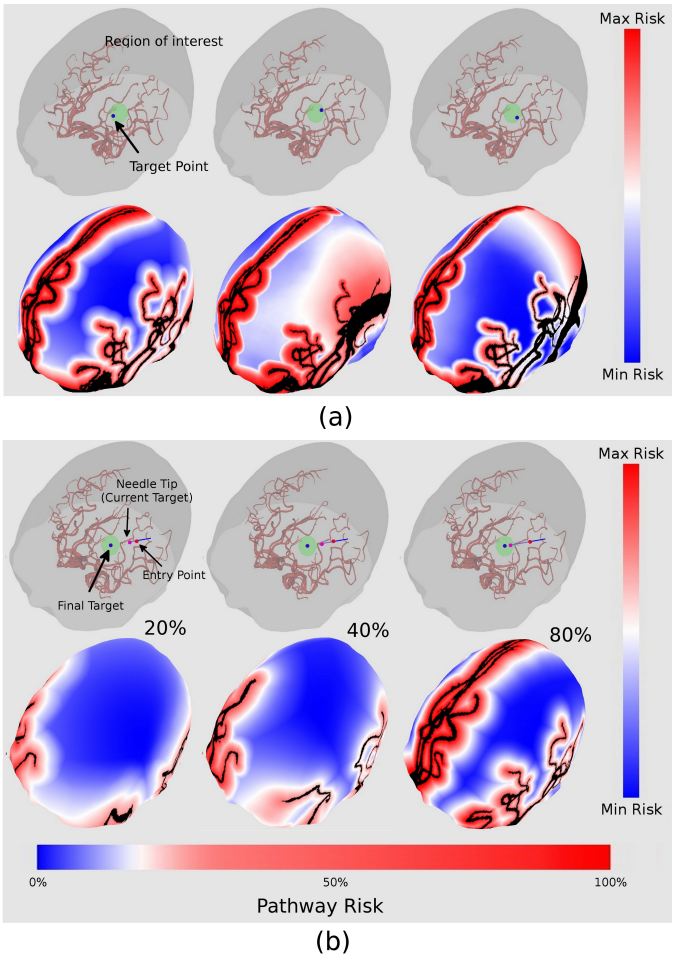

Fig. 1. Risk maps obtained using weights $k_1 = 0.1$, $k_2 = 0.9$. The computed risk values are normalized to the range of [0, 1]. Blue, white and red regions represent regions of low, medium and high risk, respectively. Black regions depict impermissible insertion areas. (a) Interactive planning by moving the target point within a region. Our high performance approach opens the possibility of interactive intra-operative visualization for computer assisted re-planning. (b) Moving the target point along a selected path. The shown risk maps are computed by placing the current target at the tip of the needle. The horizontal bar shows the normalized risk as a function of the percentage of covered pathway (i.e. 50% means that the needle is currently at half of the total length of the selected safe path). The insertion velocity can be planned according to the risk function along the selected path.

process in real-time or near real-time allows interactive visualization and planning, which would significantly reduce the planning time and thus allow effective intra-operative re-planning of neurosurgical interventions if needed.

In this article, we propose a GPU-accelerated method to process, visualize and plan neurosurgical interventions at the interactive or near real-time speed as it is required for intra-operative re-planning. It consists of two main components: (i) *Embedding the geometrical structures that represent critical tissue areas, pertinent to the procedure, into spatial data structures*. This speeds up computation of the geometric queries involved in the estimation of the risk associated with paths. (ii) *Implementing those algorithms onto graphics processing units (GPUs)*. This exploits the parallel nature of the problem, while effectively handling the involved irregular workload. Through various comparisons (e.g., between our approach and the CPU-based baseline implementation, and between our approach and an existing voxel based method), we demonstrate that besides its robustness and interactive speed (e.g., two orders of magnitude higher than the conventional method), our approach can generate safer straight access paths for neurosurgical interventions.

In practical terms, existing surgical planning approaches (e.g., the multilevel planning framework described in [9]) could benefit from a high-performance approach such as the one we propose, as it will open the possibility of intra-operative planning. In addition, the computational power might be beneficial in a pre-operative scenario by allowing to explore and visualize much larger planning spaces through fast processing, and further avoid possible complications in the procedure. For instance, our high performance approach would allow to include additional information and planning constraints (e.g. additional vital structures such as functional brain regions, and consider the angle of intersection of the tool with the targeted regions) with no large increases in computation time.

To the best of our knowledge, this work is the first to enable interactive estimation and visualization of risk in straight access neurosurgical interventions with mesh-based tissue representation. Our current clinical workflow for using the herein proposed approach entails interactively modifying (with a pointing device such as a mouse) the location of the target point and visualizing the effects on the risk maps. As a benefit of the high speed of our approach, the operator can interactively explore a much larger number of possible paths and target points, thereby enhancing the effectiveness and efficiency of the decision-making process involved in neurosurgical interventions.

1.1 Applicability

Fig. 1(a) illustrates an example of how the proposed approach can assist the operator to select both an optimal insertion and a target point. In this particular paradigm, the operator selects a target point within a pre-defined region (e.g., into the core or onto the surface of a tumor) and interactively visualizes the resulting risk map (color-coded) to guide the selection of an optimal insertion point. After a target point and an insertion point are selected, the operator can also view the risk map associated with the current position of the advancing needle (Fig. 1(b)). Such dynamic risk maps can significantly facilitate

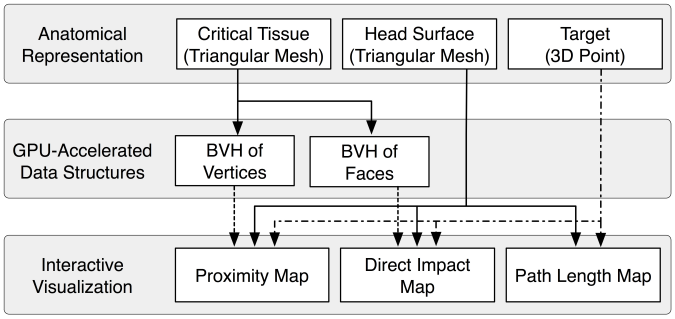

Fig. 2. Schematic view of the proposed approach. The operator can interactively visualize the risk associated to particular paths, and explore the effect of changing the position of the target point.

decision-making for the operator. When the operator selects a path, the path is usually further analyzed and inspected by looking at imaging planes (MR slices) orthogonal to the insertion path (also known as *bird's-eye view*). For stereotactic robot-assisted neurosurgical interventions, such as in the NeuroArm and NeuroMate systems, the velocity of the tooltip during insertion can be controlled based on the risk function along the path. The horizontal bar in Fig. 1(b) shows a visualization of the current risk at each point in the selected pathway. Each risk value along the pathway is computed by setting the needle tip as current target. The risk along the pathway increases non-linearly as the needle advances and reaches the maximum at the final target position. If the environment changes are known (when the tool is near the vital tissue, etc.), the operator can adjust the velocity of the insertion tool to advance more cautiously. Indeed, as shown in Fig. 1(b), the high (interactive) speed of this method allows the operator to see the exact risk from the surrounding tissue when the tool is being inserted, by simply placing the target point along the tip of the tool and recomputing the risk map at each step.

2 METHODOLOGY

2.1 Definition of Planning Risk Maps

In the formulation of our problem, selection of safe straight access paths entails minimizing a function that quantifies their associated risks. We define such a function based on the following criteria: (1) the paths cannot intersect any critical tissue, otherwise, this may result in unacceptable patient injury; (2) the paths should be as distant as possible from the critical tissue; and (3) the path length (i.e. the distance between the entry point and the target) should be as short as possible. In general, additional selection criteria can be seamlessly incorporated into this function, based on the particular needs of the procedure.

Inspired by [2], we also represent structures segmented from imaging as triangular meshes. Specifically, we represent the scalp surface as a triangular mesh $m_s = (V_S, T_S)$ and the critical tissue as another triangular mesh $m_c = (V_C, T_C)$. We consider the set of candidate paths as the line segments that extend from the vertices of m_s to a pre-defined target point p. To obtain the set of permissible paths, we discard all the unsafe paths that intersect with any triangle in m_c. Then, the risk associated with a particular path (belonging to the set of the permissible paths) starting at a vertex $v \in V_S$, is computed as:

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

$$d(\overline{vp}, v') = \begin{cases} ||v'-p|| & \text{if } (v-p)\cdot(v'-p) < 0 \\ ||v'-v|| & \text{if } (p-v)\cdot(v'-v) < 0 \\ \frac{||(v-p)\times(p-v')||}{||v-p||} & \text{otherwise} \end{cases} \quad (2)$$

Here $k_1 \geq 0, k_2 \geq 0$ are user-specified weighting parameters (see Fig. 3 for a visualization of the effect of varying these parameters). All the risk map visualizations in this writing were created using weights $k_1 = 0.1$ and $k_2 = 0.9$, unless otherwise noted. The first term in Eq. 1 accounts for the total length of the path, and the second term accounts for the proximity of the path to the critical tissue. As such, larger values of k_1 enforce the selection of short paths, while larger values of k_2 enforce the selection of paths that are farther away from critical structures. Brute-force computation of the set of permissible paths, as well as the risks associated with all the paths, turns out to be computationally-expensive even for low-resolution meshes. In order to efficiently perform the computation, we build acceleration spatial data structures from the geometric primitives of the critical tissue mesh. This significantly reduces the amount of computation required to obtain the risk map. An efficient GPU implementation of our approach further enables the computation of risk maps at an interactive rate even for high-resolution meshes. Fig. 2 shows a schematic view of our proposed approach. The operator can interactively visualize the risks associated with particular paths, and explore the effect of changing the position of the target point.

Calculation of the risk maps involves two computationally expensive steps. The first is the identification of all the paths that must be discarded, because they make direct impact on the critical tissue. In our specific formulation, it involves testing for intersection between a line segment and all the triangles in m_c. The second step is the computation of the second term in the Eq. 1 above. This computation involves determining, among all the vertices in m_c, which one is the closest to a specific line segment. Testing each path for intersection against each triangle, and computing its closest distance to each vertex are not affordable options for interactive visualization and planning.

2.2 Acceleration Spatial Data Structures

To avoid a large bulk of the computation required to obtain the risk map, without sacrificing precision, we

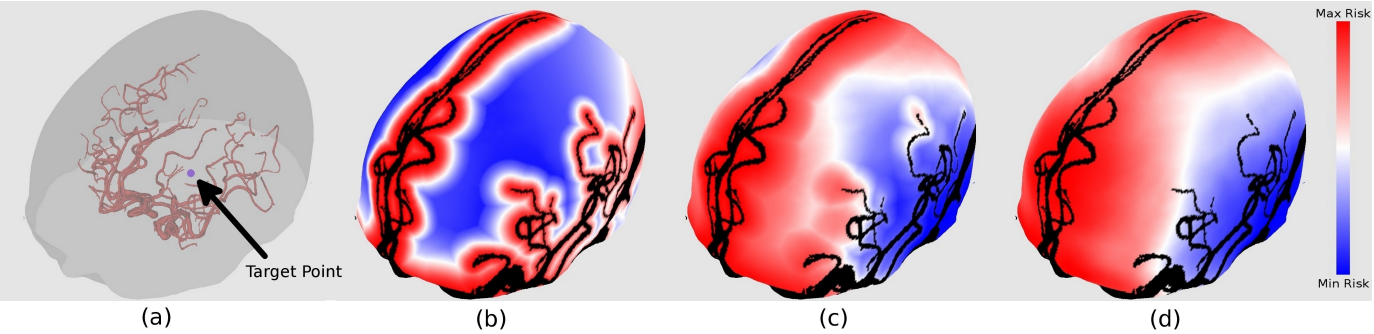

Fig. 3. Visualizations of risk maps for different pairs of weights. (a) Position of the target point. (b) Enforcing a large proximity: $k_1 = 0.1$ and $k_2 = 0.9$. (c) Enforcing the balance between length and proximity: $k_1 = 0.5$ and $k_2 = 0.5$. (d) Enforcing the selection of short paths: $k_1 = 0.9$ and $k_2 = 0.1$.

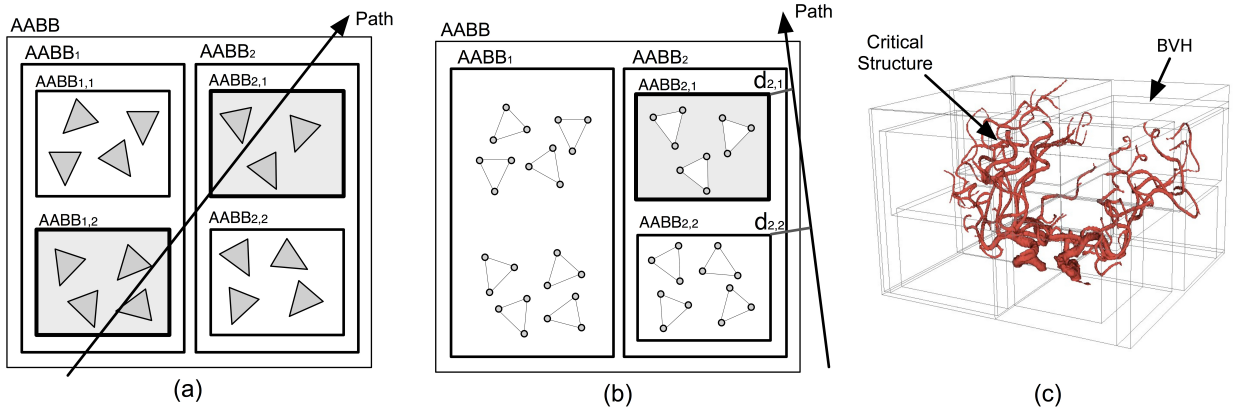

Fig. 4. Bounding Volume Hierarchy. (a) is an example of the BVH of triangles, (b) is an example of the BVH of vertices, and (c) shows bounding boxes for a 3D mesh (down to a depth of 3).

use acceleration spatial data structures. Specifically, we opt for the use of Bounding Volume Hierarchies (BVHs) [11], since they are inexpensive to compute, and they can be quickly restructured to support meshes that may go through dynamic deformations. It should be noted that in this work we only consider static meshes, not general dynamic meshes. A BVH is a partition of geometric objects into a hierarchy that can be used to accelerate geometric queries as the ones involved in our problem. Formally, BVHs are binary trees, where each node represents a group of objects and the region in space that they occupy, and children nodes are recursive partitions of the group of objects within their parent nodes. Fig. 4(a)-(b) show 2D examples of BVHs, and Fig. 4(c) shows bounding boxes for a 3D mesh (down to a depth of 3). We construct two BVHs from the critical tissue mesh m_c: one holding its triangles and the other holding its vertices. The BVH of triangles is used to compute the set of permissible paths, and the BVH of vertices is used for the computation of the proximity term (i.e., the second term in Eq. 1).

A geometric query over the BVHs involves a traversal of the data structure. To compute the set of permissible paths we need to figure out whether each path intersects a triangle in m_c. Starting from the root, at each step of the traversal, an intersection test against the children nodes is performed. If a node is intersected, then the traversal must be continued in a recursive manner over the substructure for which the node is root. Otherwise, the whole substructure can be safely ignored, because it is guaranteed that none of the triangles that it holds can be intersected by the line segment. Triangles are actually contained within the leaf nodes of the tree structure, so when a leaf node is reached, each of its triangles must be tested for intersection against the line segment. The traversal stops when a triangle hit by the segment is found or when all the triangles within the intersected nodes have been tested. Fig. 4(a) illustrates this idea for the BVH of triangles. The BVH traversal starts with a test of the box labeled AABB. As it is intersected, the traversal continues with tests of its two children (i.e., AABB_1 and AABB_2). AABB_1 is intersected, so both its children must be tested as well. $\text{AABB}_{1,1}$ is not intersected by the path (going further down that node can be safely discarded), but $\text{AABB}_{1,2}$ is intersected. Since $\text{AABB}_{1,2}$ is a leaf node, all the geometric primitives it contains must be tested for intersection. In the case of AABB_2, we can see that only its child labeled $\text{AABB}_{2,1}$ must be tested for intersection.

The computation of the proximity term (the second

term in Eq. 1) takes place in a similar fashion over the BVH of vertices. In this case, for each path we are interested in determining what is the closest vertex (in m_c) to the line segment. Fig. 4(b) illustrates a query over the BVH of vertices. For each path we determine which of the children of a node is closer, and which one is farther. In general the closest child is more likely to contain the closest vertex than the other nodes. If the closest vertex found within that node is closer to the line segment than any point in the bounding box of the farthest child node, then the farthest node can be safely discarded. In our example, node $AABB_2$ is the closest one, therefore the traversal is performed in a recursive manner over this node. Within $AABB_2$, child $AABB_{2,1}$ is the closest, and it is also a leaf node (containing vertex primitives), so the closest vertex within this node to our path is found. Since the closest vertex is closer to our path than the closest point in bounding box $AABB_{2,2}$, we can safely discard traversals through such node. In a similar way we can discard traversals through $AABB_1$.

2.3 GPU-based Parallel Implementation

In our proposed approach, the algorithm generates the risk maps by performing the calculations discussed in Section 2.2 for a large number of potential paths. Since the calculations for each path are independent from those of other paths, the computational task will be ideal for implementation on massively parallel processors such as GPUs. However, in spite of the intrinsic parallelizable nature of the problem, achieving a high-performance on GPUs is not straightforward. The origin of this is that the computations required for each path are highly variable, and this irregular workload leads to underutilization of GPU resources. To describe the presented GPU implementation, it is instructive to briefly review its operational structure. A GPU consists of a number of processing units, i.e. the streaming multiprocessors (SMs). In the streaming programming paradigm, grids of threads are subdivided into groups known as blocks. Different blocks are assigned for execution to different SM in a *static manner*. i.e. after a block has been assigned to an SM, it cannot be executed on another SM. After being assigned, the threads within each block are further subdivided into groups of 32 consecutive threads, the warps. Threads within a warp are executed in parallel within the SM in Single Instruction Multiple Thread (SIMT) mode. In SIMT mode, each thread in a warp executes the same instruction at the same time.

With respect to the problem of computing the risk map, a naive scheme of assigning work to processing units in CUDA or OpenCL would correspond to launch the execution of a grid with as many threads as the paths. Therefore, the computation for each thread would be performed in a single thread. As a result, the high variability of the required computations for different paths will lead to resource underutilization. Practically, the workload assigned to some blocks will need more time than other blocks, eventually leading to a situation where some of the SMs are idle for a portion of the overall computation time. The difference between the processing time required for different tasks can thus lead to a high workload imbalance that translates into performance degradation [12].

To obtain better performance, we balance the workload within the GPU using a centralized queue [12]. All the paths to be computed are kept on a single queue implemented as an array in the global memory. The head of the queue is also kept in the global memory, and it 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. In our work, long-running warps are those that only finish their execution when all the paths in the global queue have been processed. This can be appreciated when the warps process is considered: when they are first launched, the first thread takes the next task by updating the head of the queue through an atomic addition operation (it needs to be atomic in order to avoid race conditions). Then, each of the threads, within the warp, processes one of the paths in the task. When a warp completes this batch, it takes another task of paths, or finish execution in case the queue is empty. The processing of each path, first involves traversing the BVH of triangles for determining whether it hits the critical tissue, and then traversing the BVH of vertices to compute the proximity term of the risk function. With the centralized queue, none of the SMs is going to be idle, as long as there are remaining tasks to be processed.

3 Evaluation and Result Analysis

3.1 Computational Performance

To evaluate the efficiency of the proposed approach (i.e., to what degree it can be used in an interactive fashion), we measured the time required for performing the computation of the risk maps for meshes with different resolutions. Specifically, the mesh sizes were varied as follows: (i) for the scalp surface, the range of triangles was 2k to 276k and that of vertices was 4k to 553k; and (ii) for the critical tissue, the range of triangles was 2k to 276k and that of vertices was 46k to 300k. It is noteworthy that the number of vertices in the scalp surface mesh determines the number of paths to be processed.

We implemented our proposed approach on a GPU using CUDA, and timing was recorded using CUDA time events. With this, the recorded time includes both the execution time of the CUDA kernels, and the communication overhead due to data transfer between host and device. Specifically, it starts when the BVH data (as well as the scalp surface mesh data) are transferred to the GPU, and ends when the results from the computations of the path length and proximity terms are transferred to the main memory of the host. Our implementation was tested on a mainstream commodity desktop computer

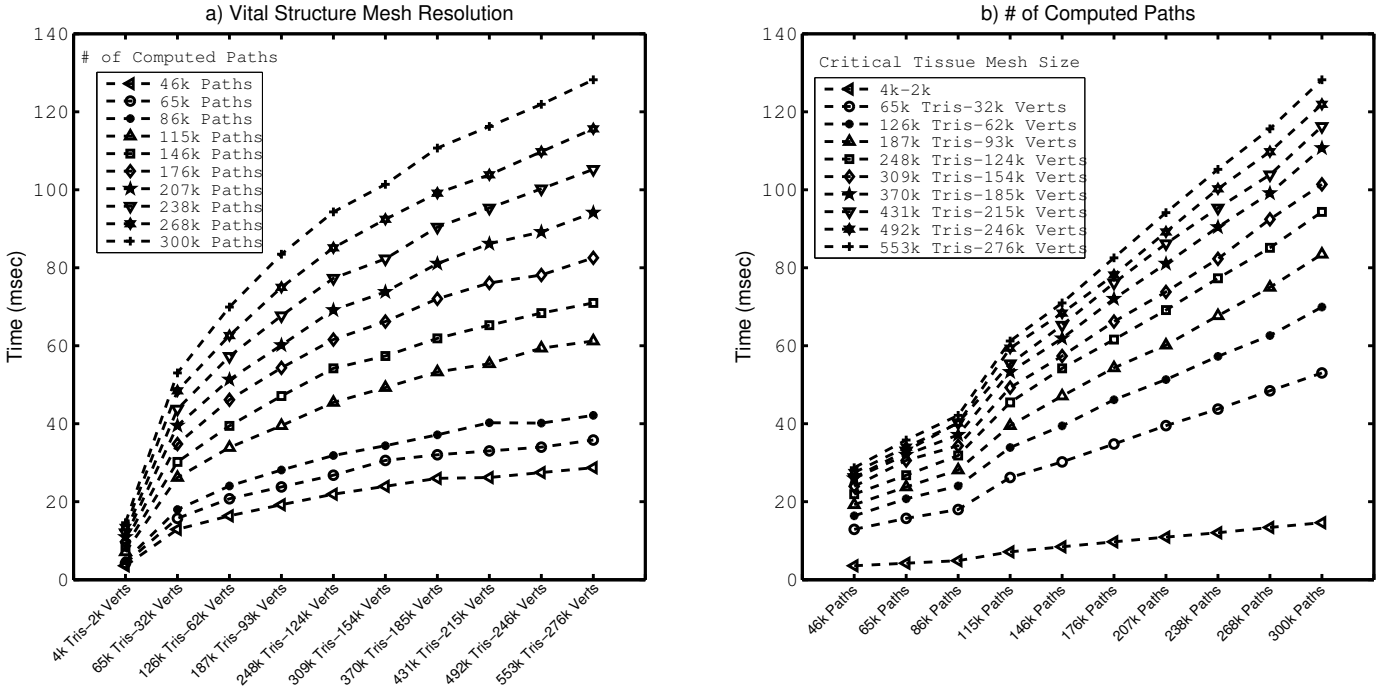

Fig. 5. Computational times for the generation of risk maps, with meshes given as: (number of triangles - number of vertices).

featuring an Intel Core i7 processor, a NVidia GeForce GTX 480 GPU, and 4 GB main memory. Fig. 5 reports the measured computational times for the above listed range of sizes of the two meshes (i.e. scalp surface and critical tissue). We observe that even for high-resolution meshes, our proposed approach can achieve interactive rates, and that it is scalable. It is noteworthy that the computational time is approximately linear to the size of the scalp surface mesh, since it depends on the number of paths to be processed. Indeed, as the mesh resolution increases, the number of paths increases linearly. By contrast, the required computational time demonstrates a logarithmic dependence on the size of the critical tissue mesh.

3.2 Comparison with CPU-based Baseline Implementation

We compared our proposed GPU-accelerated implementation with a CPU-based baseline implementation. The CPU-based baseline implementation computes the risk maps by following the same approach that we have previously described. No attempt was made to make use of the SIMD capability of the CPU in the baseline implementation. We measured the computing times of both the implementations while varying the resolution of the vital tissue mesh, and the number of paths that were considered. As shown in Fig. 8, a speed-up of two orders or magnitudes in our GPU-accelerated implementation over the baseline implementation was achieved. Interestingly, the GPU implementation also scales better to the number of computed paths and to the size of the vital tissue mesh than the CPU-based baseline implementation. That is, although both the implementations are algorithmically equivalent, the speed-up increases along with the size of the problem.

3.3 Performance Improvement due to the Task Queue

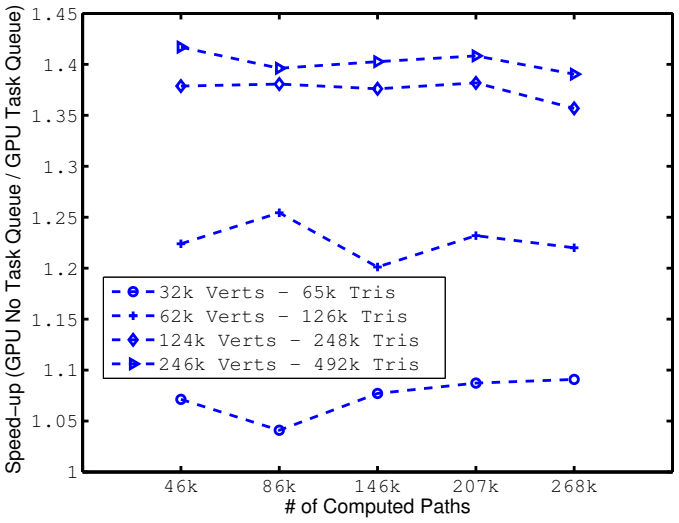

Fig. 6. Comparison between a GPU implementation with a task queue, and without a task queue in terms of the achieved speed-up (GPU no-queue time / GPU queue time).

A speed-up of up to 1.4x (i.e. a 29% reduction in computational time) was measured for a GPU implementation using the centralized task queue with respect to a baseline GPU implementation (see Fig. 6). In the

baseline GPU implementation we configure a grid that holds as many threads as there are paths to process. As shown in Fig. 6 the achieved speed-up increases with the size of the vital structure mesh, but is not significantly sensitive to the number of computed paths. The explanation for this is that when the size of the mesh is increased the application becomes memory bounded, that is, the computation becomes dominated by the time spent on memory operations.

3.4 Comparison with Voxel-based Approach

We also compared our proposed approach to a fast voxel-based method for computing the risk maps [3] running on the CPU. In both the approaches, we varied the size of the input structure as well as the number of computed paths. The input structure to the voxel-based approach [3] is volumetric data containing precomputed information about risks. The value stored in each cell of the 3D grid is an estimate of the risk involved in traversing the cell with the surgical instrument. The total risk associated with each path is then computed as the sum of the risks of all the voxels intersected by the surgical instrument. Since the input structure of the two methods are fundamentally different, we make the following assumption: the vital tissue mesh and the volumetric data are equivalent in terms of size, if the mesh is the iso-surface reconstructed from a segmented 3D image with the same resolution as the volumetric data by the classical marching cubes algorithm. Our mesh-based approach consistently outperforms the voxel-based approach for all of the input structure sizes and number of paths, as illustrated in Fig. 7. We also observe that our proposed mesh-based approach has a better scalability to problem size than the voxel-based approach. Although the computational time of both the mesh-based and voxel-based approaches is approximately linear to the number of computed paths, the voxel-based approach has the additional disadvantage of growing linearly with the size of the input structure. This leads to a quick loss of the interactive rate for the voxel-based approach, as shown in Fig. 7.

3.5 User Study

We conducted a user study in order to quantitatively assess the benefits that interactive computation of risk maps might have on the planning of straight access neurosurgical interventions (Fig. 9). Specifically, we explored the benefits that the introduced high performance approach might have on intra-operative target repositioning. We compared the safety, in terms of length and proximity, between paths that were planned using *visually-guided target position selection*, and paths selected through *risk-map guided target positioning*. The visually-guided approach stands for any method that requires planning outside of the operating room due to computational time limitations (e.g., [2]), and the risk map-guided approach stands for an approach which enables

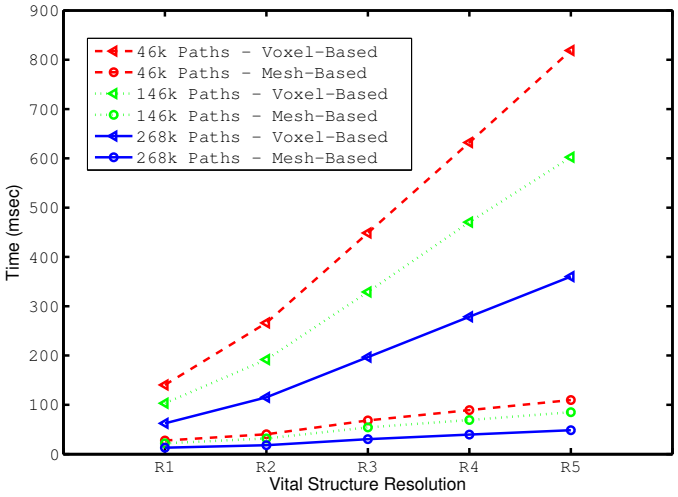

Fig. 7. Comparison between voxel-based approach [3] versus our mesh-based approach in terms of computational time. The equivalence of input structures is as follows: $R1$ is a 32k Verts-65k Tris mesh (for our mesh-based approach), or a 288x172x136 grid (for the voxel-based approach); $R2$ is a 62k Verts-126k Tris mesh, or a 576x172x136 grid; $R3$ is a 124k Verts-248k Tris mesh, or a 576x460x136 grid; $R4$ is a 185k Verts-370k Tris mesh, or a 576x768x136 grid; $R5$ is a 246k Verts-492k Tris mesh, or a 576x768x272 grid.

intra-operative re-planning, such as the one described in this article. For this purpose, seven subjects were asked to perform a set of tasks that emulate planning the insertion of a surgical tool for a straight access intervention. The subjects were graduate students of computer science with an average age of 26.4 years. Prior to the experiment, subjects were explained the concept of risk maps, and the tasks they had to perform. They had to select both the insertion point on the scalp surface and the target point within a given region of interest.

The experimental protocol consisted of the two mentioned treatments (i.e. visually-guided and risk map guided target positioning) distributed among twelve planning tasks (i.e. six tasks for each of the two treatments). Two regions of interest were defined (see their positions in Fig. 9), such that for each treatment the three tasks required planning the best insertion path for one of the regions using one of three different sets of risk map weights. The set of weights used were (i) $w_1 \equiv (k_1 = 0.1,\ k_2 = 0.9)$; (ii) $w_2 \equiv (k_1 = 0.5,\ k_2 = 0.5)$; (iii) $w_3 \equiv (k_1 = 0.9,\ k_2 = 0.1)$.

For the visually-guided treatment, the subjects were asked to find the best path they could find by selecting an insertion point and freely positioning the target without the aid of risk maps. That is, they relied on the visual inspecting and navigation of the rendered models for the surface of the head and the blood vessels. In addition to this, for each tried path, they were presented with information about the length and the proximity to blood vessels. After a satisfactory target point was selected, a

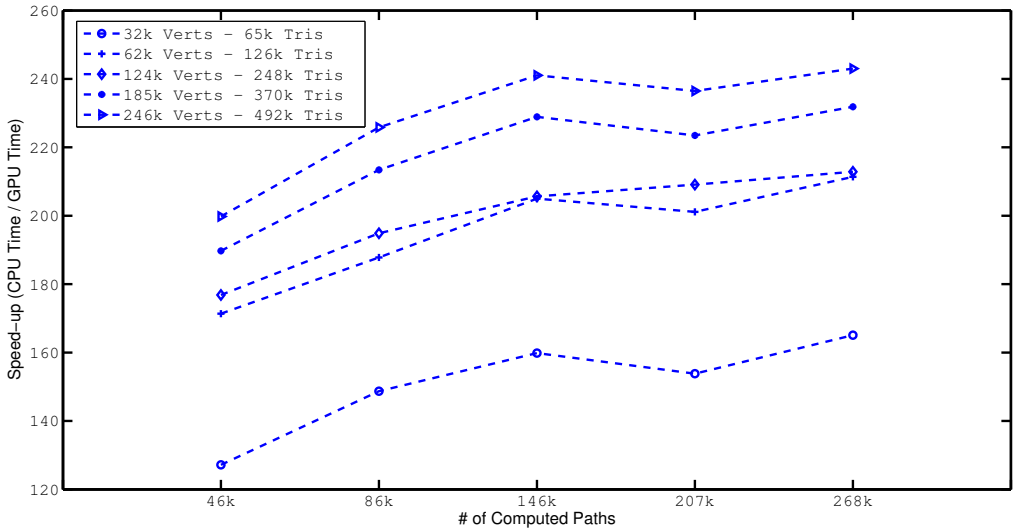

Fig. 8. Comparison between our GPU-accelerated and our CPU baseline implementations in terms of the obtained speed-up (CPU time / GPU time)

full color-coded risk map was displayed, and the subject was given the opportunity to reposition the entry point if desired.

For the risk-map guided target, the user was allowed to use guidance from the risk map in order to select a satisfactory target position. As the subject freely moved the target, full risk maps were interactively computed and displayed. Additionally, for each given target, the subject was given the opportunity to visualize a coarse grid computed within the region of interest that suggest safe regions to compute the target. This grid consist of 56 points within the region of interest, with each point holding the risk of the safest path that can be found, using the risk model presented in this article, if the point is selected as the target. A normalized color-coded visualization of the minimum risks is displayed on subject demand to guide target repositioning. It is noteworthy to mention that such visualization requires the computation of 56 full risk maps, and can be completed in less than two seconds. This computation has to be performed only once for a given region of interest.

We recorded the achieved lengths and proximity values for the planned paths, and results are shown in Fig. 10. A repeated-measures t-test show statistically significant reduction of length for every pair of weights (for w_1, $t = 2.898, p = 0.012$; for w_2, $t = 3.939, p = 0.001$; for w_3, $t = 3.738, p = 0.002$), as well as for proximity weight w_1 (for w_1, $t = -5.624, p = 0.00008$; for w_2, $t = -0.766, p = 0.457$; for w_3, $t = -1.317, p = 0.21$). For weights w_2 and w_3 we observed no evidence of improvement in terms of proximity. This can be explained by noting that weights w_2 and w_3 penalize the selection of long paths according to the risk model, in addition to the fact that, for most access pathways, their length in millimeters tends to be larger than the distance to the closest vessels.

3.6 Neurosurgeon Evaluation

The ultimate goal of our work is to be able to provide assistance to the neurosurgeon for intra-operative planning/re-planning. With the purpose of assessing how much the neurosurgeon can benefit from interactive visualization and planning within the operating room, two experienced neurosurgeons were consulted. The surgeons were explained the proposed approach, and the kind of computations that it can perform at interactive rates. They were asked to answer a questionnaire for rating how much they can benefit from some of the features enabled by the interactive approach, and asked to identify which surgical procedures can benefit, as well as to provide freeform feedback and suggestions.

The questionnaire asked to score from 1 to 6 (where 6 means "most useful" and 1 means "least useful") the usefulness of the following interactive features: (1) pre-operative risk-map guided target repositioning, (2) intra-operative risk-map guided target repositioning, (3) automatic suggestion of safest paths. Interactive pre-operative and intra-operative target repositioning were identified as highly beneficial features. Pre-operative planning is necessary for every surgical procedure, but it is considered to be a time-consuming process. As such, any tool that can speed up the process as well as provide with assistance information for exploring and validating the plan for the procedure is considered useful. Intra-operative target repositioning was identified as a highly desirable feature for surgical procedures in which tissue shifts occur (e.g., tumor resection in which brain shifts and changes form after opening the dura). Integration between the planning assistance software, the imaging acquisition mechanisms, and the surgical plan execution tool is the key in the intra-operative scenario. Automatic suggestion of safest paths was identified as an undesirable features as it takes away responsibility from the surgeon. It was stressed that the planning tool should

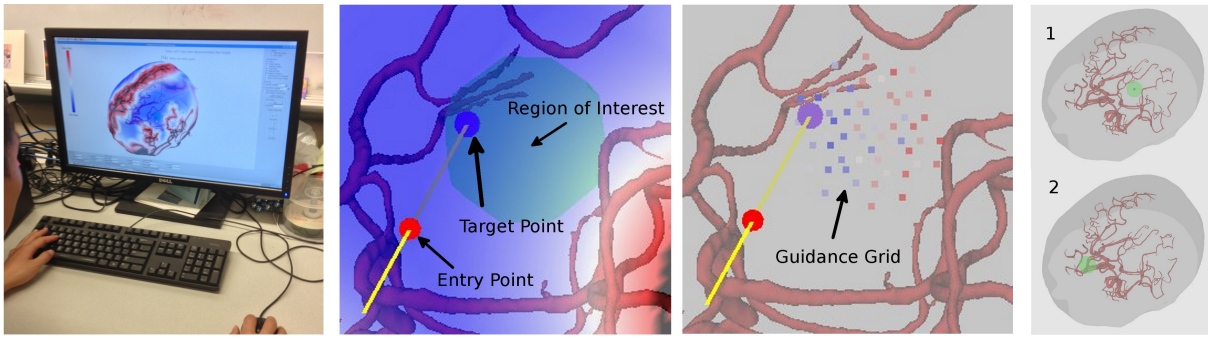

Fig. 9. (a) A snapshot of our user study, (b) zoomed view of the region of interest and selected path, (c) corresponding visualization of the grid for risk map-guided target positioning, (d) the two regions of interest specified for the user study.

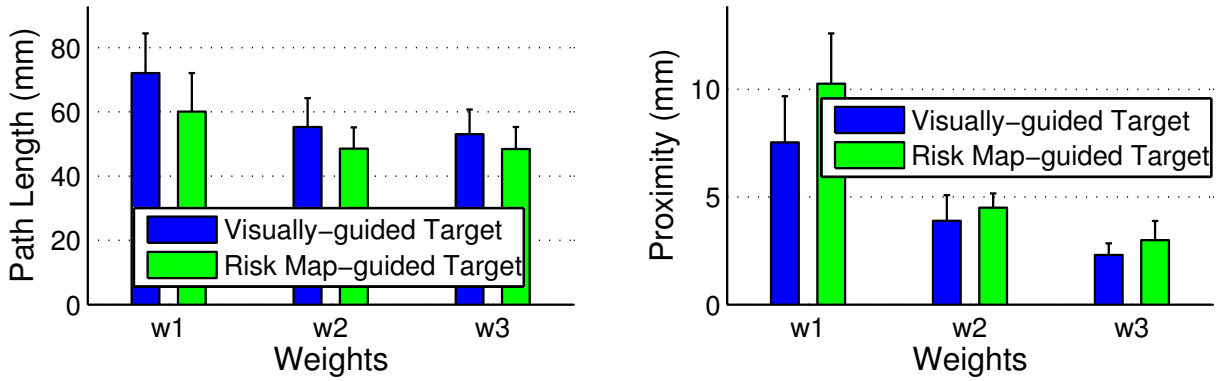

Fig. 10. Comparison of recorded lengths and proximity of the selected paths using conventional visually-guided target positioning (i.e., conventional target positioning through visual inspection), and risk-map guided target positioning (i.e. our approach), over three different sets of weights ($w_1 \equiv (k_1 = 0.1, k_2 = 0.9)$; $w_2 \equiv (k_1 = 0.5, k_2 = 0.5)$; $w_3 \equiv (k_1 = 0.9, k_2 = 0.1)$). For the visually guided target positioning subjects had to decide were to place the target by just looking and navigating through a 3D visualization of the head surface, vital structures and region of interest. For the risk-map guided target positioning, subjects were provided with an augmented view interactively showing the effect of target repositioning on the risk maps, as well as a risk grid that provided the subject with information about safe regions to position the target.

only provide assistance to the surgeon. Instead of automatic suggestion of a number of safest paths, it was recommended to show the surgeon a reduced safest area such as a cone with its tip at the target point, as this gives the neurosurgeon with the information that would both reduce the planning space and still leave space for the procedure to benefit from the surgeon experience and good criteria.

The mentioned procedures that could benefit from interactive planning include: deep brain stimulation for the placement of electrodes, shunt placement, small tumor resection, brain cyst resection, aneurysm treatment, and arteriovenous malformations (AVM). For all of these procedures it was also suggested to include information on the risk maps about functional regions of the brain, as these are actually more important in selecting the access pathways than the position of blood vessels (which can be slightly retracted during the surgical procedure). This functional regions could be included in the interactive approach as additional vital structures that should be either avoided, or whose intersection would carry a penalization that would affect the risk of the access pathways. Another related and important application that could be benefited by the proposed computational framework is the procedures using the gamma knife, which is used to apply radiation doses over tumors. Precise planning of radiation trajectories that minimize the exposure of neighboring tissue could be benefited by a high performance approach, as it requires heavy computations as well.

4 Conclusions and Future Work

This article introduces a novel efficient GPU-accelerated scheme for the generation of guidance-maps tailored for neurosurgical interventions with straight access. It allows ultrafast processing and visualization even for high-resolution meshes (e.g., 86k and 115k paths for a critical tissue mesh of 553k triangles and 276k vertices require 75 ms and 116 ms, respectively), which facilitates interactive planning and intra-operative visualization.

The current work is focused on neurosurgical interventions with straight access paths. In the future, we plan to further investigate efficient approaches for procedures on dynamically changing structures, secondary to breathing, cardiac beating, and/or to the interventional procedure. Another future research direction would be to extend the current work for efficient planning and visualization of surgical interventions with non-straight access paths.

ACKNOWLEDGEMENTS

This work is supported in part by NSF IIS-0914965, CNS-0932272, and research gifts from Google and Nokia. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the agencies.

REFERENCES

[1] E. J. L. Brunenberg, A. Vilanova, V. Visser-Vandewalle, Y. Temel, L. Ackermans, B. Platel, and B. M. ter Haar Romeny, "Automatic trajectory planning for deep brain stimulation: A feasibility study," in *Proc. of MICCAI '07*, 2007, pp. 584–592.

[2] N. V. Navkar, N. V. Tsekos, J. R. Stafford, J. S. Weinberg, and Z. Deng, "Visualization and planning of neurosurgical interventions with straight access," in *Proc. of International Conference on Information Processing in Computer-Assisted Interventions - IPCAI'10*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 1–11.

[3] R. R. Shamir, I. Tamir, E. Dabool, L. Joskowicz, and Y. Shoshan, "A method for planning safe trajectories in image-guided keyhole neurosurgery," in *Proc. of Medical Image Computing and Computer-Assisted Intervention - MICCAI 2010*. Springer, 2010, pp. 457–464.

[4] S. Bériault, F. A. Subaie, K. Mok, A. F. Sadikot, and G. B. Pike, "Automatic trajectory planning of DBS neurosurgery from multimodal MRI datasets," in *Proc. of 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," in *Proc. of Medical Imaging and Augmented Reality - 5th International Workshop, MIAR 2010*. Springer, 2010, pp. 149–158.

[6] C. Essert, C. Haegelen, F. Lalys, A. Abadie, and P. Jannin, "Automatic computation of electrode trajectories for deep brain stimulation: a hybrid symbolic and numerical approach," *Int'l J. of Comp. Assist. Rad. and Surg.*, vol. 7, no. 4, pp. 517–532, 2011.

[7] A. C. F. Colchester, J. Zhao, K. S. Holton-Tainter, C. J. Henri, N. Maitland, P. T. E. Roberts, C. G. Harris, and R. J. Evans, "Development and preliminary evaluation of VISLAN, a surgical planning and guidance system using intra-operative video imaging," *Medical Image Analysis*, vol. 1, no. 1, pp. 73–90, 1996.

[8] R. Viard, N. Betrouni, J. Rousseau, S. Mordon, O. Ernst, and S. Maouche, "Needle positioning in interventional MRI procedure: real time optical localisation and accordance with the roadmap." in *Proc. of IEEE Engineering in Medicine and Biology Society (EMBS)*, Nov 2007, pp. 2748–2751.

[9] P. C. Herghelegiu, V. Manta, R. Perin, S. Bruckner, and E. Gröller, "Biopsy planner - visual analysis for needle pathway planning in deep seated brain tumor biopsy," *Comput. Graph. Forum*, vol. 31, no. 3, pp. 1085–1094, 2012.

[10] R. Khlebnikov, B. Kainz, J. Muehl, and D. Schmalstieg, "Crepuscular rays for tumor accessibility planning," *IEEE Trans. Vis. Comput. Graph*, vol. 17, no. 12, pp. 2163–2172, 2011.

[11] J. Arvo and D. Kirk, *A survey of ray tracing acceleration techniques*. Academic Press, 1989, pp. 201–262.

[12] T. Aila and S. Laine, "Understanding the efficiency of ray traversal on GPUs," in *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conference on High Performance Graphics 2009*. ACM, 2009, pp. 145–149.

AUTHORS' BIOGRAPHIES

Mario Rincón-Nigro is a PhD student in the Department of Computer Science at University of Houston. His research interests include computer graphics, GPU computing, and high-performance graphics algorithms/systems. He had received his B.S. in Systems Engineering from University of Los Andes, Venezuela, in 2005, and his M.S in Computer Science from the University of Houston in 2012.

Nikhil V. Navkar is a PhD candidate at the Department of Computer Science at the University of Houston. His research interests include image-guided interventions and medical robotics. He earned his M.S. from University of Houston in 2009 and Bachelor of Engineering from Devi Ahilya University in 2005, respectively.

Nikolaos V. Tsekos is an Associate Professor of Computer Science at the University of Houston. His research interests include Magnetic Resonance Imaging and MR-Compatible robotic Manipulators. He earned his Ph.D. from University of Minnesota in 1995, M.S. from University of Illinois at Urbana-Champaign in 1992, and B.S. from University of Athens, Greece in 1989, respectively.

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. He earned his Ph.D. from University of Southern California in 2006, M.S. from Peking University in 2000, and B.S. from Xiamen University in 1997, respectively. He is a senior member of IEEE and a member of ACM.