Fast and Adaptive Fractal Tree Based Path Planning for Programmable Bevel Tip Steerable Needles

Fangde Liu, Arnau Garriga-Casanovas, Riccado Secoli and Ferdinando Rodriguez y Baena

***Abstract*—Steerable needles are a promising technology for minimally invasive surgery, as they can provide access to difficult to reach locations while avoiding delicate anatomical regions. However, due to the unpredictable tissue deformation associated with needle insertion and the complexity of many surgical scenarios, a real-time path planning algorithm with high update frequency would be advantageous. Real-time path planning for nonholonomic systems is commonly used in a broad variety of fields, ranging from aerospace to submarine navigation. In this paper, we propose to take advantage of the architecture of Graphics Processing Units (GPUs) to apply fractal theory and thus parallelize real-time path planning computation. This novel approach, termed Adaptive Fractal Trees (AFT), allows for the creation of a database of paths covering the entire domain, which are dense, invariant, procedurally produced, adaptable in size, and present a recursive structure. The generated cache of paths can in turn be analyzed in parallel to determine the most suitable path in a fraction of a second. The ability to cope with nonholonomic constraints, as well as constraints in the space of states of any complexity or number, is intrinsic to the AFT approach, rendering it highly versatile. Three-dimensional simulations applied to needle steering in neurosurgery show that our approach can successfully compute paths in real-time, enabling complex brain navigation.**

***Index Terms*—Reactive and Sensor-Based Planning, Surgical Robotics: Steerable Catheters/Needles**

I. Introduction

MINIMALLY invasive surgery (MIS) is becoming the standard of care for a range of medical procedures, including biopsies, targeted drug delivery, and brachytherapy cancer treatment. The advantages of MIS include less trauma for the patient, lower risk of complications, and a shorter full-recovery time. Current standard medical practice uses rigid tools, which enable good accuracy, but are not capable of accessing locations behind delicate regions. Steerable needles [1], [2] have the potential to overcome these limitations, and to improve reliability through automation, resulting in a significant advancement in keyhole surgery.

Existing steerable needle concepts can be classified in seven different groups, as outlined in [3]: base manipulation [4], bevel tip (with and without a "kinked tip") [5]–[7], pre-curved stylet [8], active cannula [9], [10], optically controlled needle [11], tendon actuated tip [12] and programmable bevel tip [13]–[15]. Our own design, code-named Soft Tissue Intervention and Neurosurgical Guide (STING) [16]) has a bio-inspired design that reproduces the multi-segment ovipositor of certain parasitic wasps, is made of flexible plastic and is fully Magnetic Resonance Imaging (MRI) compatible. It has the ability to steer along three-dimensional paths without duty cycle spinning along the insertion axis, as shown in Figure 2, and thus offers an ideal target system for the path planning technique described in this work.

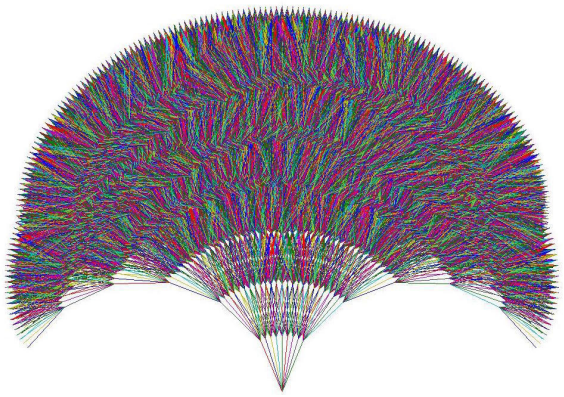

Fig. 1. Generic L-tree illustrating the Adaptive Fractal Trees concept. The density of paths corresponds to the space coverage that can be achieved in real-time with modern GPUs.

In most of these applications, the uncertainties arising from tissue deformation during insertion and consequent need of frequent path replanning to track the motion of one or several targets, warrants a real-time path planning algorithm, with a high update frequency [17]. The design of real-time path planning algorithms capable of online updates, however, is challenging, especially when differential constraints are present. The problem is NP-hard [18]. General methods from variational optimization [19] [20], or approaches from optimal control such as the Gauss pseudospectral method [21], are capable of accurately finding the optimal solution; however, they require significant computational time. Potential fields based methods [22] and most other probabilistic methods are unable to handle nonholonomic constraints. Linear path planners for chained-form systems [23] have been applied for some steerable needle designs, but cannot cope with control saturation associated with large tissue deformation. The limited robustness of probability maps [24] or inverse kinematics based approaches [25] prevents their use in safety-

Manuscript received: August, 31st, 2015; Revised December, 4th, 2015; Accepted January, 18th, 2016.

This paper was recommended for publication by Editor Ken Masamune upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the European Research Council under the European Union Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no [258642-STING].

The authors are with the Mechatronics In Medicine Laboratory, Mechanical Engineering Department, Imperial College London, UK. fangde.liu@imperial.ac.uk

Digital Object Identifier (DOI): see top of this page.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2528292, IEEE Robotics and Automation Letters

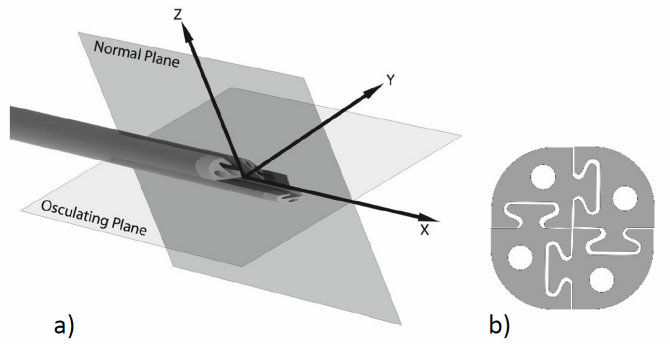

Fig. 2. **a)** Rendering of the STING distal end [28] **b)** STING cross-section with interlocking mechanism.

critical applications, such as in surgery. Path planners based on homotopy groups [26] or Lie groups [27] exploit symmetries in the path to accelerate the processing speed. However, these solutions must be computed iteratively due to their nonlinear nature, leading to an unbounded computational time.

Sampling-based methods are the dominant trend [29] in problems with differential constraints. Standard approaches such as the Dijkstra method, or the improved, heuristics based version, A* [29], are able to effectively find paths with obstacle avoidance, but the search is excessively time consuming. Even algorithms that improve on A* by reusing previous search information [30] require significant computational time, and can only be scaled to multiple CPUs. Rapidly-Exploring Random Trees (RRTs) [31] [32], and specifically Reachability-Guided RRTs (RG-RRTs) [33], are becoming increasingly popular due to their ability to quickly explore the entire domain and cope with curvature constraints for needle steering. RRTs perform well in environments with relatively simple obstacles, presenting short computational times that allow online path replanning during insertion [34]. However, in congested environments, with complex obstacles, even purpose-developed heuristically accelerated RRTs present computation times that are relatively long and unbounded [35] [36].

A common issue in the majority of existing approaches is that they perform the search sequentially, relying on serial CPU computing, the speedup potential of which is limited. Instead, by exploiting the power of the Graphics Processing Unit (GPU) for general purpose processing, the computation time can be reduced by over one order of magnitude. Some early approaches to path planning on the GPU are reported in [37] [38], highlighting their potential advantages over CPU-based algorithms. However, the performance improvement of these algorithms is limited to ten times that of CPU based implementations, a result which can be improved. Parallelization of RRTs is also reported in the literature [39], although the algorithm is only scalable to multiple CPUs, and presents a limited speed improvement. This is a consequence of the search procedure in RRTs, which leads to a variable computational load due to an iterative growth, potentially causing the system to stall when multiple threads require the same tree to update simultaneously, and may not meet the ”single instruction multiple data operations” requirement, which the GPU is designed for.

This paper proposes a novel approach to path planning, which is tailored for a GPU-based implementation. The strategy introduced in this work employs fractal theory to create a data structure that enables efficient parallel path planning. The resulting parallelized problem has a recursive structure, is adaptable in size, is constructed procedurally, and allows a dense coverage of the entire domain, as illustrated in Fig. 1. For this, the method has been termed Adaptive Fractal Trees (AFT). Our approach presents three main advantages with respect to existing imaged-based algorithms. First, it works directly with voxels, optimizing computational performance. Second, it is capable of real-time replanning with a bounded computational time. Third, it can be used regardless of the number or complexity of the obstacles, rendering it robust and versatile, with a high success rate compared to other path planning algorithms.

The paper is structured as follows. The path planning problem for a steerable needle is formally stated in Section II. Section III provides a description of the AFT approach, together with an analysis of its specific properties for parallelization. Simulated results, together with the corresponding discussion, are presented in Section IV, leading to the conclusion of this paper in Section V.

II. Problem formulation

A. Path Planning for Programmable Bevel Tip Needles

For the purpose of path planning, only a description of the STING's distal end is necessary, since it can be assumed that the body will follow the path dictated by the tip [14]. The robot configurations form a subspace of the special Euclidian group, with $q(t) \in SE(2)$ for 2D [15] and $q(t) \in SE(3)$ for 3D [28]. The initial and target configurations are indicated by q_i and q_f, respectively. The interaction between needle and tissue, together with the robot design, lead to a set of non-holonomic constraints, valid at least locally in an infinitesimal neighborhood of time and space. Defining a direction x tangent to the insertion path, two first constraints arise from a no-slip condition, $V_y = V_z = 0$, which are the linear velocities along the y and the z axes respectively. The STING is designed to steer in 3D without duty-cycling along the insertion axis, x, hence a kinematic constraint on the rotational velocity along the insertion axis arises, $w_x = 0$. The curvatures of the resulting path along the y and z directions, defined as $k_{y,z} = \frac{w_{y,z}}{V_x}$, are determined by the bevel tip geometry. This is specifically calculated to prevent excessive stress on the needle, leading to a bounded curvature between a minimum $L_{y,z}$ and a maximum $U_{y,z}$: $L_{y,z} \leq \frac{w_{y,z}}{V_x} \leq U_{y,z}$. Considering these premises, along with a needle design that suffers from negligible torsional effects, we employed the Bishop frame [40] as the most suitable frame to describe the needle motion.

The obstacles in the configuration space correspond to either physical obstacles or virtual constraints. Due to tissue deformation, the spatial position of the obstacles may vary [41]. It is assumed that feedback from their position, as well as from the current and target configurations of the needle tip,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2528292, IEEE Robotics and Automation Letters

is available from an appropriate source (e.g. an intraoperative imaging or tracking system).

The aim of a path planner is to find a feasible path from q_i to q_f that respects all of the constraints, and optimizes a cost function. In general, the cost function to minimize is defined as a risk-based function, possibly with additional components, such as the minimization of the insertion length, as to reduce tissue damage.

B. General Path Planning Problem

More generally, we are considering a system described in implicit form by $q \in \mathbb{R}^n$, with a set of $k \leq n$ smooth linearly independent[1] nonholonomic Pfaffian constraints

$$w_i(q)\dot{q} = 0 \; i = 1, ..., k \tag{1}$$

which may also include any number of obstacles of any complexity, denoted in the configuration space by Q_{obs}. The path planning problem for this system can be equivalently formulated as a steering control problem [42].

The corresponding system can be expressed as

$$\dot{q} = \sum_{i=1}^{m} g_i(q)u_i \tag{2}$$

where $m = n - k$ and $u \in U \subset \mathbb{R}^m$ are the control inputs, with

$$span\{g_1, ..., g_m\} = span\{w_1, ..., w_k\}^{\perp} \tag{3}$$

Considering the obstacles Q_{obs} to be static, the path planning problem is then to find the input functions $u_{1,...,k}$ that steer the system from an initial q_i to a target configuration q_f, while optimizing a cost function and avoiding Q_{obs}.

III. Adaptive Fractal Trees Algorithm

The recursive nature of motion in nonholonomic systems closely resembles the topological structure of a tree. The possible motion at each step depends on the previous one, a process that reverses recursively to the initial point, or the tree origin. Despite the advantages of the parametric form (2), path planning for systems with differential constraints remains challenging; the majority of existing numerical solutions are sampling-based and rely on serial iterative computing processes, requiring often excessive, and unbounded computational time. Their parallelization to suit GPU specifications is either difficult or impossible.

By uniformly discretizing the control space, the path adopts a fractal structure. Such fractal space can be divided into subspaces, in a coarse to fine manner, as

$$T_s = T_{s1} + T_{s12} + T_{s13} + ... + T_{si} \tag{4}$$

Each subspace T_{si} can be parallel processed by the GPU. This results in a novel method for massively parallel path planning, with an efficient search. The resolution increases exponentially with each subspace, leading to fast convergence.

[1]A subset of the constraints may be locally linearly dependent. In such case, the rank of the distribution associated to the action vectors g_i increases locally, without further consequences on the path planning presented in this paper.

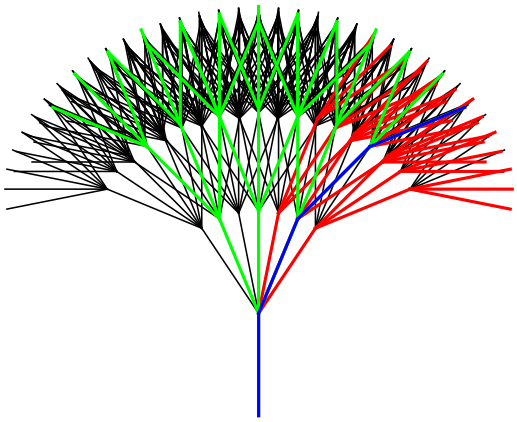

Fig. 3. Illustration of the adaptive search concept in a two-stage approach. The coarse tree (green) first explores the entire domain. The fine tree (red) is concentrated around the most promising region, providing higher resolution. The blue line highlights the most suitable path.

A. Motion Fractal Tree

Relying on the parametric form of a nonholonomic system (2), all possible paths can be mapped to an L-tree, as shown in Fig. 1. Beginning at q_i, the first set of tree ramifications corresponds to the action vectors g_i of the system, advancing by an increment that can be symbolized by δ_u in each of the s directions. Then, each branch is divided and subsequently given the motion action inputs, generating a fractal structure.

The number of required increments is determined by the needle insertion distance, and the computational time is bounded by the limited needle length. A path is then determined by a string of configurations $q_T = [i_1, i_2, ..., i_N]$, where N is the total number of increments required. The entire domain of possible motions is discretized exhaustively using a fractal tree, as illustrated in Fig. 1. A differential increment between ramifications would lead to an exact approximation of all possible paths. However, the number of paths increases exponentially with the number of ramifications and, as the discretization step decreases, the size of the path space grows, becoming infinite for a differential increment. Hence, for any given application, a specific incremental step must be selected.

This structured construction of the tree is implemented efficiently by the GPU, as explained in the following subsections. This property is in contrast with the random construction of RRTs, and it represents one of the distinctive advantages of AFT for fast computation.

B. Adaptive Discretization

By exploiting the tree property, as in (4), it is possible to break down the search into subspaces. This division has the particular property that all subspaces share the same number of motion segments and topology.

Each tree can be parametrized by three elements: l, which corresponds to the segment's length, δ_k, which describes the branch's aperture, and C, the tree's central path. The latter is either provided by a previous coarse search, or taken as a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2528292, IEEE Robotics and Automation Letters

straight line for the first generated tree. The size of a tree is therefore adaptable, depending on the construction parameters.

In this way, the path search can be executed in a coarse to fine manner, reducing the problem's complexity exponentially, and achieving high accuracy in the fine search. First, the path planner creates and searches a coarse tree T_{s1}. Then, the path that minimizes a cost function within T_{s1} is used to build a second, finer tree around it, the density of which is increased exponentially with respect to the previous one.

The adaptable search concept is illustrated in Fig. 3, where a two-stage approach is depicted. First, a coarse tree is generated covering the entire domain, in order to determine the most promising region. Then, a second tree is constructed to perform the fine search, focusing the computational resources around the region identified by the coarse tree, with a higher density of paths that minimizes the error. In general, after two or three stages, the desired resolution is reached.

C. Parallel Path Planning Algorithm

The AFT path planning algorithm is composed of three parts: (1) motion segment reconstruction, (2) collision detection and distance to target calculation, and (3) back-tracking and pooling.

A cost function is defined in order to evaluate the paths and determine the most suitable one. In this case, the cost function is composed of three parts, as

$$C(q_T) = w_1 R(q_T) + w_2 T(q_T) + w_3 D(q_T) \qquad (5)$$

where w_i represents a weighting parameter, R is a risk-based function, T is a function associated to trauma, and D represents the distance between the needle tip and the target configuration. The distance to target is defined here as the Euclidean distance.

The database of paths is generated at any time using the aforementioned tree parameters. The cache does not need to be stored in memory, which suits the GPU architecture. The initial and target configurations, as well as the obstacles, are assumed to be available from an appropriate feedback source. The tree is constructed starting from the initial point. The limited path length of steerable needles allows a fast computation of the action list.

Collision detection is then applied to the cache of paths. Medical applications require high accuracy, and the anatomical obstacles tend to present complex/irregular boundaries. Here, it is assumed that some image processing has been applied on the raw feedback data, and the voxels representing the obstacles have been identified. Our path planner then checks each voxel on the tree for possible collision, marking the path segments where this occurs. The distance to target is also computed and stored for each segment.

Back-tracing is then performed. It begins with checking whether the segments are collision-free. Then it proceeds towards the tree root, assessing possible collisions within the paths. If all segments of a path are collision-free, then it is marked as viable.

Finally, a parallel maximum pooling is executed, selecting, among the collision-free paths, the one that minimizes the cost function. This path then becomes the central line for the next stage, around which the path planner then refines the search.

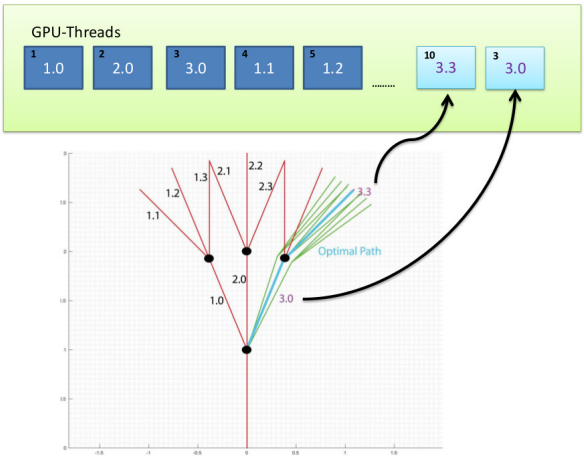

Fig. 4. Diagram of the enumeration and allocation of tree segments to the GPU threads. Each tree segment is assigned to one GPU thread. The most suitable path and corresponding threads are highlighted in cyan.

Algorithm 1 AFT Basic Algorithm

Input: q_i, q_f, Q_{obs}
Output: $q_{c_{\min}}$
Initialization
N, B, J
$q_T = \emptyset$
Recursion Loop
for $j = 1$ to J **do**
  $\mathbf{RefineTreeAround}(q_T)$
  for all ID: $i \leq N$ **do**
    $q_i \leftarrow \mathbf{MotionPlan}(i)$
    $c_i \leftarrow \mathbf{Cost}(q_i)$
  end for
  $T \leftarrow \mathbf{IndexOfMin}(c_1, c_2, \ldots, c_N)$
end for
return q_T

As a result, the method described here combines the robustness of RRTs with the parallelization possibilities of path caches, leading to an algorithm that is advantageous with respect to both. This algorithm is reported as Algorithm 1. The initial and target configurations, as well as the obstacles, are first inputted. The parameters for the tree construction, l, δ_k and C, are then determined according to the number of segments, N, and branches, B. The recursion depth, J, is also introduced, which represents the number of tree refinements (typically two). A recursion loop is then executed to generate and evaluate a tree at each step. The tree is adapted in the function $\mathbf{RefineTreeAround}(q_T)$ around path q_T, which is taken to be straight for the first iteration. All processes are executed in parallel to construct the tree and compute the cost of each path, as defined in equ.(5), which represents the function $\mathbf{Cost}(q_T)$. Subsequently, the minimum cost path is identified in the function $\mathbf{IndexOfMin}$, which is executed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2528292, IEEE Robotics and Automation Letters

by parallel reduction. This minimum cost path is used in the next iteration of the "for loop". When iteration J is reached, the path that minimizes the cost function, q_T, is determined, which is the output of the algorithm.

Fractal trees can be easily parallelized. Each tree segment can be allocated to a GPU thread, as shown in Figure 4, optimizing the use of computational resources. The cost evaluation and motion plan reconstruction are the kernel for parallel computing, which consumes the majority of computational time and space. Due to the GPU architecture, the kernel (line 5-8 of Algorithm 1) is optimized, as described in Algorithm 2.

Algorithm 2 AFT Optimized Kernel

Input: ID, q_i, q_f
Output: c_{ID}
Initialization
$q \leftarrow q_i$
$i \leftarrow 0$
$parent_{ID}$ is $root_{ID}$
$child_{ID}$ is $root_{ID}$
$\mathbf{Cost}(q_{parent_{ID}}) \leftarrow 0$
while $child_{ID} \neq ID$ **do**
  $child_{ID}, parent_{ID} \leftarrow \mathbf{Child}(parent_{ID}, ID)$
  if $child_{ID} \neq \emptyset$ **then**
    $parent_{IDcur} \leftarrow parent_{ID}$
    $q \leftarrow q + \mathbf{Action}(child_{ID})$
    continue
  end if{Calculating the last segment cost}
  $S_{last} \leftarrow \mathbf{BuildSegment}(parent_{IDcur}, parent_{ID})$
  $p_1, p_2, p_3, \ldots, p_w \leftarrow \mathbf{Dice}(S_{last})$.
  $c_{ID} = \sum_{i=1}^{N} \mathbf{Cost}(p_i)$
end while
SYNCHRONIZE GPU THREADS
while $parent_{ID}$ is not $root_{ID}$ **do**
  $parent_{ID} \leftarrow \mathbf{Parent}(parent_{ID})$
  $c_{ID} \leftarrow c_{ID} + \mathbf{Cost}(q_{parent_{ID}})$
end while

The inputs of Algorithm 2 are the ID of each segment, and the initial and target configurations. The algorithm initializes by establishing the maximum path length L and variable i, as well as as creating an array of costs $\mathbf{Cost}(q_{ID})$. To maximize efficiency, the cost for each segment is only computed once. As the ID of each segment is allocated, the cost is then calculated and stored into an array for all segments. The **BuildSegment** function builds the segment S_{last} between the $parent_{IDcur}$ and $parent_{ID}$. S_{last} is then sampled into $p_1, p_2, ..., p_w$ sub-segments, using the **Dice** function. The corresponding cost of each sample is calculated and accumulated to define the total cost c_{ID} of the whole segment S_{last}. After synchronizing the parallel threads, back tracking is then applied to calculate the cost of each path. This is executed using the function **Parent**, which determines the parent segment corresponding to each segment. In this manner, the system tracks back each node to its parent, summing the contribution of each segment to the aggregate cost, and thus obtaining the total cost associated with each path. The array of costs for each path is the output of the algorithm.

An important factor for efficient parallelization is the enumeration of each segment with an ID. Exploiting the fractal structure of the tree, parent and child IDs have a regular pattern. By travelling up and down the tree, an enumeration maps each path ID to a series of control actions.

IV. Simulation Setup

An application of AFT to minimally invasive surgery is presented in this section, in order to validate the algorithm in a statistically significant manner. In particular, simulations corresponding to 3D liver navigation are reported, as they showcase the capability of AFT to plan a path in real-time in a highly congested and complex environment.

Tissue deformation during needle insertion can lead to displacements of a few centimeters. As a consequence, target migration and variations in the spatial position of the obstacles can be significant, requiring path replanning with a high update frequency, as the needle is being inserted. In this work, it is assumed that the initial and target configurations of the steerable needle, as well as the obstacles, are available from a suitable intra-operative imaging modality, e.g. Interventional Magnetic Resonance Imaging or Ultrasound.

AFT is specifically designed to recalculate a path as the environment varies during needle insertion. The short and fixed computational time associated with AFTs allows replanning with an update frequency that can match the feedback imaging system, eliminating the need for complex low level control.

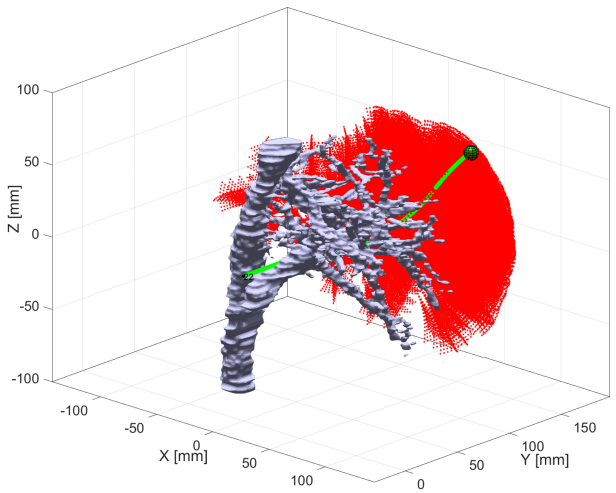

Fig. 5. Simulation of a coarse 3D search through the segmented vasculature of liver. The full tree of paths is colored red. The best path is shown in green.

The simulations reported here include a representative set of 100 different 3D path planning problems encountered during needle insertion into liver, and simulated online replanning during needle insertion, with target motion. The 100 problems correspond to different initial configurations randomly generated within a bounded domain, and three fixed targets, as shown in Figures 6,7. The online replanning is simulated in a particularly complicated needle insertion, with a target moving continuously during the insertion, with a total displacement

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2528292, IEEE Robotics and Automation Letters

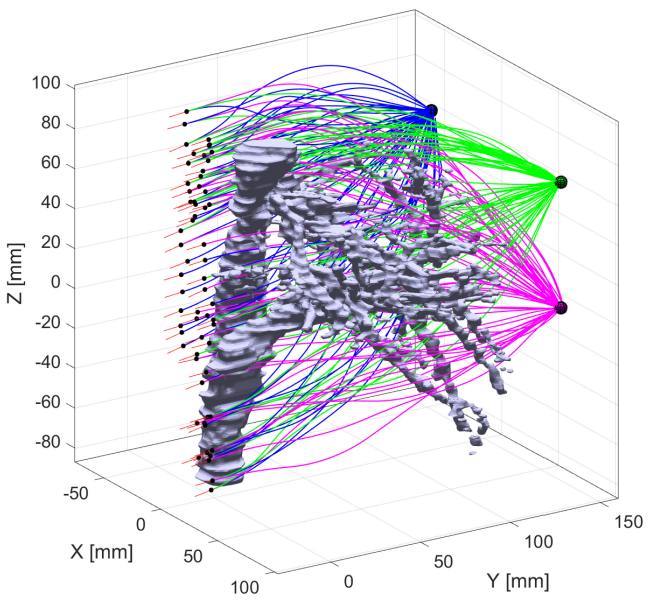

Fig. 6. Simulation results of liver path planning with the AFT algorithm, showing all of the best paths which intersect three random targets (blue, green and magenta), varying the entry position (black dots) and insertion direction.

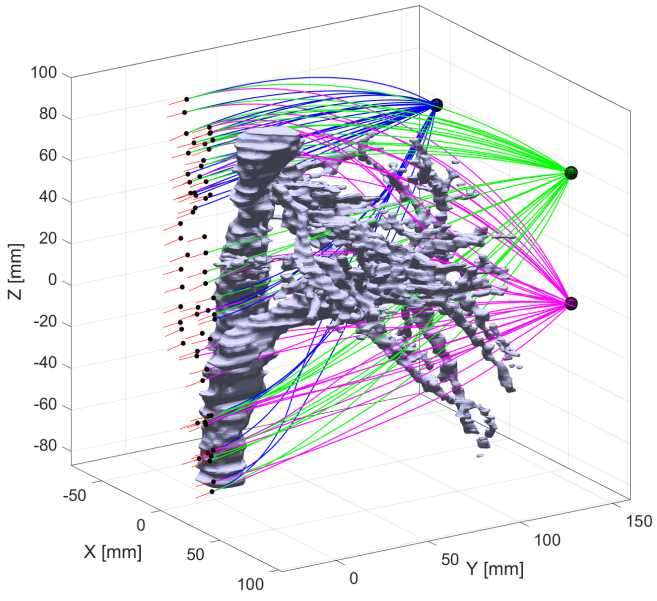

Fig. 7. Simulation results of liver path planning with the RG-RRT algorithm, showing all of the best paths that intersect three random targets (blue, green and magenta), varying the entry position (black dots) and insertion direction. For some entry points, the algorithm fails to provide suitable paths to reach the targets.

of 2 cm. The simulations are in a common 3D environment, which represents a segmented CT scan of a liver (Liver Dataset [43]) in voxel format, with a resolution of 256x256x256. This CT scan image volume was selected as it includes a high number of vessels that define a challenging obstacle map, where existing algorithms such as RRTs experience difficulties in finding a solution. It is assumed that an image processing algorithm is available to label the obstacles in the intra-operative images [44]. Similarly, it is assumed that the needle configuration can be estimated from the images [45].

In the simulations, the **Cost** function is defined to favor the shortest path that arrives closest to the target, without intersecting any obstacle. The Euclidean distance was used to measure the proximity to the target. The parameters for the simulations were as follows: maximum needle curvature = $0.014\,mm^{-1}$; search space = $100 \times 100 \times 200\,mm^3$; discretization AFT step = $2mm$; maximum insertion length = $160\,mm$; entry points randomly generated in a bounding box of $-30 \leq x \leq 0\,mm$, $-5 \leq y \leq 5\,[mm]$ and $-83 \leq z \leq 100\,[mm]$ in position, and a variation of 10 degrees with respect to vector $[0, 1, 0]$ in orientation; target positions $[x, y, z] = [57, 157, 58], [-57, 157, 58], [57, -157, -5]\,mm$; cost function parameters: $w_1 = w_2 = w_3 = 1$; number of search paths $N = 1024 * 1024 * 17$, $B = 17$ and $\delta_k = 1/280$.

In our setup, the code is implemented in Matlab 2014b© (Mathworks Inc.), Linux Ubuntu 64bit, and executed on an Intel CORE i7 CPU @ 3.2Ghz with a GTX TITANX from NVIDIA corp., with 3072 threads, a 1GHz base-clock and 12GB of memory. This GPU has an approximate computing power of 7 TFLOP and supports CUDA 7.5 API [46].

The simulation of RRTs in the same path planning problems is also reported in order to compare the performance of AFT with one of the most widely used algorithms in MIS. RRTs are implemented in the same setup, with a maximum number of iterations of 16,000. An RG-RRT implementation is adopted, as it provided faster convergence in our tests. A goal bias sampling strategy was used, with 50% of the samples on the target and the remaining 50% randomly distributed.

The performance tests conducted indicate that, with our setup, 300 million paths per second can be evaluated. Consequently, considering a typical surgical application, where a 20Hz update frequency is required, the algorithm would be capable of assessing 15 million paths per second. This computational power translates into a resolution that approaches the limits of the imaging device.

V. Results and Discussion

The results of an illustrative AFT path planning problem are shown in Fig. 5. As can be seen, a high density of paths (red) are surveyed, and the path that minimizes the cost function is selected (green), with a total computation time of just 5.2 ms.

The results of the simulation of 100 different AFT path planning problems in a prototypical scenario are shown in Figure 6. In this case, only the selected paths are displayed for clarity, showing the ability of the AFT algorithm to negotiate complex obstacles. The average final error in the feasible paths identified is 1.45 mm, with a standard deviation of 1.19 mm. The average computation time is 5.15 ms, with a corresponding standard deviation of 0.048 ms. The small variation in computation time between the different simulations is a result of the automatic speed adjustment of the GPU. However, the computation time is fixed and independent of the complexity of the obstacles. This represents a significant advantage of AFT in surgical applications.

In comparison, a standard RRTs implementation, which is taken here to be one of the best competing algorithms in the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2528292, IEEE Robotics and Automation Letters

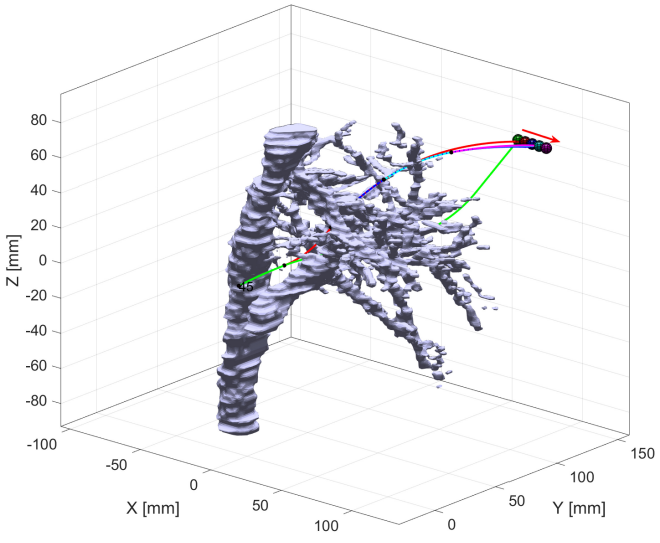

Fig. 8. Simulation results of online path replanning during needle insertion into liver, with target motion along the red arrow direction. In green the best initial path and subsequent best paths (red, blue, cyan, magenta).

literature, performs considerably worse than our proposed algorithm. The RRTs simulation performed on the same data set and with the same setup indicates that, after 16000 iterations (corresponding to an approximate computation time of 30s in our, non-optimized implementation), a path is found in 42% less cases than in AFT. The preliminary paths found using RRTs after 16,000 iterations are shown in Figure 7. As can be seen, only the simpler cases are solved, whereas the remaining cases would require a significantly higher number of iterations to reach a solution.

The relatively low success rate of RRTs in an environment with complex obstacles is a consequence of the search strategy employed by the algorithm. In RRTs, the space is sampled, and then paths linking to the tree are searched. While this strategy is successful in many environments, links to the tree can be difficult to find in the presence of complex obstacles, requiring high sampling resolution. The computational cost increases exponentially with the number of samples, hindering the use of RRTs in highly congested environments.

AFTs, on the other hand, provide a higher success rate in real-time, regardless of the number and complexity of the obstacles. Such robustness is a result of the algorithm construction and implementation, which exploits the GPU architecture to survey a high number of paths in parallel. In this regard, the AFT algorithm is particularly suited to surgical applications, where the ability to update a plan in real-time in the presence of any number of complex obstacles, would be advantageous.

The result of a simulated online replanning using AFT during needle insertion is shown in Fig. 8. As can be seen, the algorithm initially calculates a path (green). However, as the target moves during insertion, the online replanning finds a more suitable path (red), which is re-calculated and improved to account for target motion.

VI. Conclusion

Efficient three-dimensional path planning in complex environments remains challenging, especially in scenarios requiring a real-time implementation. In this work, the path planning problem can be solved in real-time, even for systems with nonholonomic constraints and complex environments, with a novel algorithm which we named Adaptive Fractal Trees (AFT). The application of AFT enables the parallelization of the path planning problem, which in turn unlocks the massive computational speedup potential of the GPU, leading to ms long path searches, regardless of the complexity of the surgical scenario. The use of AFT enables the search to be conducted in a coarse to fine manner, with a database of paths that can be procedurally produced. In this way, a perfect match between the algorithm and the hardware capabilities is achieved. In addition, the fractal tree that is generated translates into a dense, invariant and organized exploration of the entire domain. This represents an advancement with respect to existing algorithms in terms of robustness and success rate of path planning in highly constrained and complex environments. As a result, the approach described in this paper allows the path planning problem to be computed in real-time, with the resolution and update frequency necessary for many surgical applications.

VII. Acknowledgments

The authors would like to thank NVIDIA Corp. for kindly donating the GPU employed in these simulations.

Arnau Garriga Casanovas is a research engineer of the Centre for Doctoral Training (CDT) in Non Destructive Evaluation (NDE), based at Imperial College London, working in collaboration with Rolls-Royce plc, under an Engineering Doctorate scheme. The support from both the CDT in NDE, funded through EPSRC, and Rolls-Royce is gratefully acknowledged.

References

[1] D. Glozman and M. Shoham, "Image-guided robotic flexible needle steering," *Robotics, IEEE Transactions on*, vol. 23, no. 3, pp. 459–467, 2007.

[2] R. J. Webster, J. S. Kim, N. J. Cowan, G. S. Chirikjian, and A. M. Okamura, "Nonholonomic modeling of needle steering," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 509–525, 2006.

[3] N. van de Berg, D. van Gerwen, J. Dankelman, and J. van den Dobbelsteen, "Design choices in needle steering 2014;a review," *Mechatronics, IEEE/ASME Transactions on*, vol. PP, no. 99, pp. 1–12, 2014.

[4] K. Reed, A. Majewicz, V. Kallem, R. Alterovitz, K. Goldberg, N. Cowan, and A. Okamura, "Robot-assisted needle steering," *Robotics Automation Magazine, IEEE*, vol. 18, no. 4, pp. 35–46, Dec 2011.

[5] K. Reed, V. Kallem, R. Alterovitz, K. Goldberg, A. Okamura, and N. Cowan, "Integrated planning and image-guided control for planar needle steering," in *Biomedical Robotics and Biomechatronics, 2008. BioRob 2008. 2nd IEEE RAS EMBS Int. Conf. on*, oct. 2008, pp. 819 –824.

[6] V. Kallem and N. Cowan, "Image guidance of flexible tip-steerable needles," *Robotics, IEEE Trans. on*, vol. 25, no. 1, pp. 191 –196, feb. 2009.

[7] J. A. Engh, D. S. Minhas, D. Kondziolka, and C. N. Riviere, "Percutaneous intracerebral navigation by duty-cycled spinning of flexible bevel-tipped needles," *Neurosurgery*, vol. 67, no. 4, pp. 1117–1122, Oct 2010.

[8] P. Swaney, J. Burgner, H. Gilbert, and R. Webster, "A flexure-based steerable needle: High curvature with reduced tissue damage," *Biomedical Engineering, IEEE Transactions on*, vol. 60, no. 4, pp. 906–909, April 2013.

[9] P. Dupont, J. Lock, B. Itkowitz, and E. Butler, "Design and control of concentric-tube robots," *Robotics, IEEE Transactions on*, vol. 26, no. 2, pp. 209–225, April 2010.

[10] D. Rucker, B. Jones, and R. Webster, "A model for concentric tube continuum robots under applied wrenches," in *Robotics and Automation (ICRA), 2010 IEEE Int. Conference on*, may 2010, pp. 1047 –1052.

[11] S. C. Ryu, Z. F. Quek, J.-S. Koh, P. Renaud, R. Black, B. Moslehi, B. Daniel, K.-J. Cho, and M. Cutkosky, "Design of an optically controlled MR-Compatible active needle," *Robotics, IEEE Transactions on*, vol. 31, no. 1, pp. 1–11, Feb 2015.

[12] P. Qi, H. Liu, L. Seneviratne, and K. Althoefer, "Towards kinematic modeling of a multi-DOF tendon driven robotic catheter," in *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, Aug 2014, pp. 3009–3012.

[13] L. Frasson, T. Parittotokkaporn, B. Davies, and F. Rodriguez y Baena, "Early developments of a novel smart actuator inspired by nature," in *Mechatronics and Machine Vision in Practice, 2008. M2VIP 2008. 15th Int. Conference on*, dec. 2008, pp. 163 –168.

[14] L. Frasson, S. Ko, A. Turner, T. Parittotokkaporn, J. F. Vincent, and F. Rodriguez y Baena, "Sting: a soft-tissue intervention and neurosurgical guide to access deep brain lesions through curved trajectories," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 224, no. 6, pp. 775–788, 2010.

[15] S. Y. Ko, L. Frasson, and F. Rodriguez y Baena, "Closed-loop planar motion control of a steerable probe with a "'programmable bevel"' inspired by nature," *Robotics, IEEE Trans. on*, vol. 27, no. 5, pp. 970 –983, oct 2011.

[16] S. Y. Ko and F. Rodriguez y Baena, "Trajectory following for a flexible probe with state/input constraints: An approach based on model predictive control," *Robotics and autonomous systems*, pp. 509 – 521, 2012.

[17] S. Patil, J. Burgner, R. J. Webster, and R. Alterovitz, "Needle steering in 3-d via rapid replanning," *Robotics, IEEE Transactions on*, vol. 30, no. 4, pp. 853–864, 2014.

[18] S. Lazard, J. Reif, and H. Wang, "The complexity of the two dimensional curvatureconstrained shortest-path problem," in *Proceedings of the Third International Workshop on the Algorithmic Foundations of Robotics,(Houston, Texas, USA)*, 1998, pp. 49–57.

[19] D. R. Smith, "Variational methods in optimization," *Mineola, N.Y.: Dover Publications, Inc.*, 1998.

[20] O. Junge, J. E. Marsden, and S. Ober-Blöbaum, "Discrete mechanics and optimal control," in *Proceedings of the 16th IFAC World Congress*, vol. 16, no. 1, 2005, pp. 00 310–1.

[21] B. Fornberg, "A practical guide to pseudospectral methods," *Cambridge University Press*, 1998.

[22] C. I. Connolly, J. Burns, and R. Weiss, "Path planning using laplace's equation," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990, pp. 2102–2106.

[23] R. M. Murray and S. S. Sastry, "Steering nonholonomic systems in chained form," *in Decision and Control, Proceedings of the 30th IEEE Conference on. IEEE*, pp. 1121–1126, 1991.

[24] W. Park, J. S. Kim, Y. Zhou, N. J. Cowan, A. M. Okamura, and G. S. Chirikjian, "Diffusion-based motion planning for a nonholonomic flexible needle model," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 4600–4605.

[25] V. Duindam, J. Xu, R. Alterovitz, S. Sastry, and K. Goldberg, "Three-dimensional motion planning algorithms for steerable needles using inverse kinematics," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 789–800, 2010.

[26] R. A. Knepper, S. Srinivasa, and M. T. Mason , "Toward a deeper understanding of motion alternatives via an equivalence relation on local paths," *International Journal of Robotics Research*, vol. 31, no. 2, pp. 168–187, February 2012.

[27] K. M. Seiler, S. P. Singh, S. Sukkarieh, and H. Durrant-Whyte, "Using lie group symmetries for fast corrective motion planning," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 151–166, 2012.

[28] R. Secoli and F. Rodriguez y Baena, "Closed-loop 3d motion modeling and control of a steerable needle for soft tissue surgery," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 5831–5836.

[29] S. M. LaValle, "Planning algorithms," *Cambridge University Press*, 2006.

[30] M. Likhachev, D. Ferguson, G. Gordon, A. T. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

[31] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[32] J. Xu, V. Duindam, R. Alterovitz, and K. Goldberg, "Motion planning for steerable needles in 3d environments with obstacles using rapidly-exploring random trees and backchaining," in *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*. IEEE, 2008, pp. 41–46.

[33] S. Patil and R. Alterovitz, "Interactive motion planning for steerable needles in 3d environments with obstacles," *Proceedings of the 2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2010.

[34] S. Patil, J. Burgner, R. Webster, and R. Alterovitz, "Needle steering in 3-d via rapid replanning," *Robotics, IEEE Transactions on*, vol. 30, no. 4, pp. 853–864, Aug 2014.

[35] S. Patil *et al.*, "Motion planning under uncertainty in highly deformable environments," *Robotics science and systems: online proceedings*, 2011.

[36] C. Caborni, S. Y. Ko, E. De Momi, and G. Ferrigno, "Risk-based path planning for a steerable flexible probe for neurosurgical intervention," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*. IEEE, 2012, pp. 866–871.

[37] J. Kider, M. Henderson, M. Likhachev, and A. Safonova, "High-dimensional planning on the gpu," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 2515–2522.

[38] C. Park, J. Pan, and D. Manocha, "Real-time optimization-based planning in dynamic environments using gpus," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 4090–4097.

[39] J. Ichnowski and R. Alterovitz, "Parallel sampling-based motion planning with superlinear speedup," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 1206–1212.

[40] R. L. Bishop, "There is more than one way to frame a curve," *The American Mathematical Monthly*, vol. 82, no. 3, pp. 246–251, 1975.

[41] H. Rivaz, S. J.-S. Chen, and D. L. Collins, "Automatic deformable mr-ultrasound registration for image-guided neurosurgery," *Medical Imaging, IEEE Transactions on*, vol. 34, no. 2, pp. 366–380, 2015.

[42] R. M. Murray, Z. Li, and S. S. Sastry, "A mathematical introduction to robotic manipulation," *CRC Press*, 1994.

[43] 3D-Slicer. (2009) Liver segmentation tutorial. [Online]. Available: http://www.slicer.org

[44] A. de Brebisson and G. Montana, "Deep neural networks for anatomical brain segmentation," *arXiv preprint arXiv:1502.02445*, 2015.

[45] P. Chatelain, A. Krupa, and N. Navab, "3d ultrasound-guided robotic steering of a flexible needle via visual servoing," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 2250–2255.

[46] J. Nickolls and W. J. Dally, "The gpu computing era," *IEEE Micro*, vol. 30, pp. 56–69, 2010.