GA3C Reinforcement Learning for Surgical Steerable Catheter Path Planning

Alice Segato[1], Luca Sestini[1], Antonella Castellano[2] and Elena De Momi[1]

***Abstract*— Path planning algorithms for steerable catheters, must guarantee anatomical obstacles avoidance, reduce the insertion length and ensure the compliance with needle kinematics. The majority of the solutions in literature focuses on graph based or sampling based methods, both limited by the impossibility to directly obtain smooth trajectories. In this work we formulate the path planning problem as a reinforcement learning problem and show that the trajectory planning model, generated from the training, can provide the user with optimal trajectories in terms of obstacle clearance and kinematic constraints. We obtain 2D and 3D environments from MRI images processing and we implement a GA3C algorithm to create a path planning model, able to generalize on different patients anatomies. The curvilinear trajectories obtained from the model in 2D and 3D environments are compared to the ones obtained by A* and RRT* algorithms. Our method achieves state-of-the-art performances in terms of obstacle avoidance, trajectory smoothness and computational time proving this algorithm as valid planning method for complex environments.**

I. INTRODUCTION

Keyhole-neurosurgery (KN) is a minimally-invasive procedure performed to reach targets located deep inside the brain, through a very small hole in the skull, called "burr hole" or "keyhole" [1]. Through the keyhole, catheters can be inserted into the brain for biopsy and therapies, as drug delivery or electrical stimulation. Modern KN is trying to substitute the use of a rigid needles with steerable ones, in order to increase their dexterity [2]. Path planning for steerable devices is a very challenging task: an automatic path planning algorithm for such needle has to take into consideration the maximum degree of curvature admitted by the needle, guaranteeing the clearance from anatomical obstacles by considering also the diameter of the needle.

The goal of this work is to develop a learning-based path planner, able to pre-operatively assist the surgeon, estimating optimal curvilinear trajectories (CTs). Given an environment, as the one shown in Figure 1A, the path connects a surgeon-defined entry point to a target, guaranteeing the clearance from anatomical obstacles such as blood vessels and corticospinal tracts, and complying with catheters kinematic limits. The proposed method, by solving the path planning problem with a reinforcement learning approach, permits to save computational time by avoiding subsequent optimization steps presented in previous approaches and increases planned trajectory accuracy.

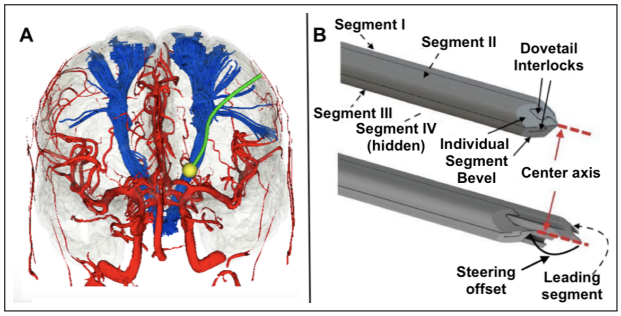

Fig. 1: (A) Representation of the 3D Environment with target structure (yellow) vessels (red), corticospinal tract (blu) and a curvilinear trajectory (green). **(B)** Illustration of the PBN catheter showing the four segments interlocked by means of Dovetail joints; the IV segments can slide on each other along the centre axis, creating an offset which allows steering of the catheter (courtesy of Leibinger, 2016).

This project is carried on in the context of EUs Horizon EDEN2020 aiming at advancing the current state of the art in neurosurgical technology, by a new biomimetic flexible probe (PBN) [3], shown in Figure 1B.

II. RELATED WORK

Path planning problem can be defined as finding a set of subsequent positions, connecting a starting point and a target point, avoiding collisions with known obstacles. When applied to steerable needles, as the PBN, path planning has to take into consideration also the kinematic constraints of the needle and its non-holonomicity. In the context of path planning applied to steerable needles, a variety of approaches has been proposed in literature.

Duindam et al. proposed a 3D motion planning for a steerable needle as a dynamical optimization problem with a discretization of the control space using inverse kinematics [4]. This approach, based on the kinematic model of the needle, is able to provide the region of feasible paths, but has little capability to take into account other crucial aspects as the obstacle avoidance. Potential field methods, originally introduced in [5], are based on the computation of a field that increases getting closer to the obstacles, which has the disadvantage of determining local minima. To address this problem, in the context of brachytherapy procedures, Li et al. [6] suggested a path-planning method with obstacle avoidance capability, based on an artificial potential field where a conjugate gradient algorithm is used. Clearance from anatomical structures can be achieved, but the method does not allow to optimize the trajectory in order to minimize its length or to meet specific kinematic constraints.

*This project has received funding from the European Unions EU Research and Innovation programme Horizon 2020 under grant agreement no 688279.

[1]Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milano , Italy alice.segato@polimi.it

[2]Neuroradiology Unit and CERMAC, Vita-Salute San Raffaele University and IRCCS Ospedale San Raffaele, Milan, Italy

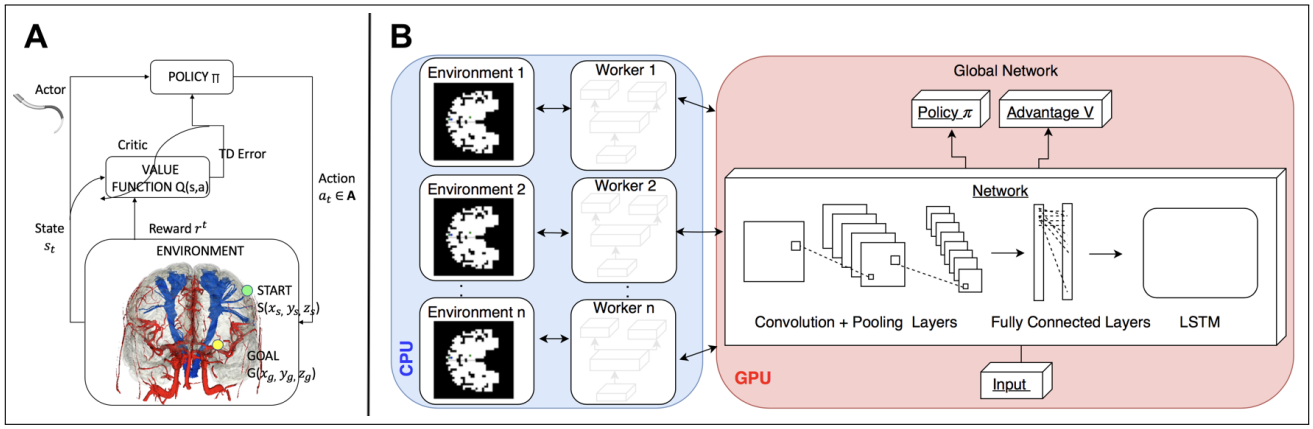

Fig. 2: (A) The actor takes as input the state (s_t) and outputs the best action ($a_t \in A$). It essentially controls how the agent behaves by learning the optimal policy (π). The critic, on the other hand, evaluates the action by computing the value function ($Q(s,a)$). **(B)** the Network used by GA3C. The pink box contains the global neural network, having the current frame (f_t) as input and the Policy π and the Advantage V as outputs, and running on the GPU. The blue box contains the n workers running in parallel, each one with a copy of the global network, and periodically interacting with it.

A. *Graph based method*

Dijkstra algorithm [7] and A* [8] are two typical graph-search methods based on the discrete approximation of the planning problem. They are "resolution-complete" algorithms, as they can determine in finite time weather a solution exists, and "resolution-optimal" since they can estimate the best path, according to length, given the specific resolution of the approximation. Park et al. presented a diffusion-based motion planning for a non-holonomic flexible needles [9]. Although graph-based methods are relatively simple to implement, they require a considerable computational time as the environment size increases [10].

B. *Sampling based method*

Based on the random sampling of the working space, sampling-based methods avoid the discretization typical of graph-based solutions, strongly reducing the computational time required to solve the path-planning problem. Rapidly-exploring Random Tree (RRT) [11] is an exploration algorithm for quick search in high-dimensional spaces, more efficient than brute-force exploration of the state space. Its enhanced versions, RRT* [12] and bidirectional-RRT [13] are "probabilistically complete" as the the probability to find an existing solution tends to one, as the number of samples goes to infinity, and "asymptotically optimal", as they can refine an initial raw path by increasing the sampling density of the volume. In neurosurgical context, Caborni et al. [14] proposed an RRT-based approach, limited to catheters moving on 2D space.

C. *Learning based method*

Graph-based and sampling-based methods, considered the standard approaches for path planning, are limited in the context of KN with flexible catheters, by the impossibility, unless using further steps, to directly optimize the trajectory in terms of obstacle clearance and kinematic constraints. Deep Reinforcement Learning (DRL) has recently been used in the path planning domain. [15], [16] demonstrate that DRL is suitable for solving path planning problems. Several studies [17], [18], [19] about applying DRL in navigation, focus on static environments, without motion or change of the environment. [20] applied the DRL approach to a grid path planning problem, with promising results on small environments. Hierarchical approaches are also widely researched in literature [21], [22], [23]. Aleksandra et al. [22] integrated sampling-based path planning with reinforcement learning agents for indoor navigation and aerial cargo delivery.
Learning based methods, and in particular reinforcement learning, are more flexible approaches with respect to graph-based and sampling-based methods, as those presented by [24], [25], [26], allowing one to directly include all the expected features (obstacle clearance, kinematic constraints meeting, minimum trajectory length) in the optimization process, without the need of subsequent refinement steps, which are time-consuming and may still not provide the optimal trajectory. The goal of this work is to explore a reinforcement learning approach in the context of KN, to overcome the above mentioned limitation of classical and modified versions of sampling-based and graph-based methods, when dealing with steerable needles.

III. METHOD

A. *Problem Statement*

The problem can be stated as follows: let us consider an agent operating on an grid, static environment composed of free cells and occupied cells, corresponding to obstacles. At every cell the agent can take an action, corresponding to a movement toward a free adjacent cell (8 possible actions in 2D, 26 in 3D). Actions moving the agent toward an occupied cell or outside the environment are considered inadmissible. Given a starting cell, S(x_s,y_s,z_s), placed on the skull, and a target cell G(x_g,y_g,z_g), placed inside the brain and defined by the coordinates of their center, the task is to find a path ($P = \{X_0, X_1, .., X_{n-1}\}, X_0 = S, X_{n-1} = G$) as an admissible sequence of free cells.

B. Environment Creation

High-resolution MR images of seven healthy subjects were acquired on a 3T Ingenia CX scanner (Philips Healthcare, Best, The Netherlands). The ethical committee of Vita-Salute San Raffaele University and IRCCS San Raffaele Scientific Institute approved the study, and all subjects provided signed informed consent prior to MR imaging. The MRI protocol included: a 3D T1-weighted sagittal Fast-Field Echo (acquisition matrix: 320 × 299; voxel size, 0.8 × 0.8 × 0.8 *mm*; thickness: 0.8/0 *mm* gap;) a 3D high-resolution time-of-flight MR angiography (TOF-MRA) (acquisition matrix: 500×399; acquired voxel size, 0.4×0.5×0.9 *mm*; reconstructed voxel size: 0.3 × 0.3 × 0.45 *mm*; thickness: 0.45/-0.45 *mm* gap) and high angular resolution diffusion MR images (HARDI) with diffusion gradients applied along 60 non-collinear directions (acquisition matrix, 128×126; voxel size, 2×2×2 *mm*; thickness, 2/0 *mm* gap). Both the ToF and the T1-weighted images were segmented by thresholding in order to obtain, blood vessels and brain models, respectively. From HARDI images, MR Tractography reconstruction of the corticospinal tracts (CST) based on a q-ball residual bootstrap algorithm were obtained using Diffusion imaging in Python (Dipy) software ([27]; [28]). The three datasets: 1) ToF for vessels, 2) DTI for CST and 3) T1 for brain cortex were registered, allowing to obtain 3D binary label maps (dimension 256 × 256 × 256 *mm*). The label maps were used to generate the 3D map, with each voxel corresponding to a cell (free=0 or occupied=1, depending on the label). In addition, from every 3D map, a 2D map was obtained by considering the central slice, parallel to the frontal plane. The term "map" indicates the constructed environment. For each of the 7 patients 1 3D and 1 2D map were generated, for a total of 7 2D and 7 3D maps, and used as the environment in the reinforcement learning model.

C. GA3C for Keyhole Neurosurgery Trajectory Planning

In reinforcement learning, an agent interacts with an environment (ε). At each time step (t), the agent receives, from the environment, its current state (s_t) and selects an action (a_t) from a set of possible actions ($\mathbf{A}$), according to its policy (π), such that:

$$\pi(s_t) = a_t \tag{1}$$

In response, the agent receives the next state (s_{t+1}) and a scalar reward (r_t), according to a predefined reward function. The goal of the agent is to determine an optimal policy π^* allowing it to take actions inside the environment, maximizing, at each t, the sum of discounted rewards $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r^t$, with γ, in range (0,1], called "discount factor". The value function V:

$$V(s) = E[R_t | s_t = s] \tag{2}$$

gives the expectation of R_t, given the current state s_t. The state-action value function Q:

$$Q(s,a) = E[R_t | s_t = s, a_t = a]) \tag{3}$$

gives the expectation of R_t according to the current state s_t, after taking an action a_t, according to the current policy (π).

In this work, the reinforcement learning problem is addressed with the Asynchronous Advantage Actor-Critic on a GPU (GA3C) algorithm [29][30], which combines Deep Q-Network (DQN) with Actor-Critic (AC) algorithms, and performs learning using an asynchronous approach.
The GA3C algorithm is actor-critic (AC) because it has a separate memory structure to explicitly represent the policy π, independently from the value function V (Figure 2A) [31]. The critique learns to approximate V, by means of a "temporal difference (TD) error", defined as:

$$TD = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$

The actor learns to approximate the policy $\pi(s_t)$.
Advantage Actor-Critic algorithms, iteratively learn the policy π performing update steps proportional to the advantage function $ADV(s_t, a_t)$, defined as:

$$ADV = Q(s_t, a_t) - V(s_t) \tag{5}$$

With respect to normal Actor-Critic, which performs update steps proportional to the value function only, using $ADV(s_t, a_t)$ helps reducing the high variance of policy networks and stabilizes the model [29].
The method is also asynchronous: it consists of a global network, whose copies are assigned to independent agents named "workers", whose number is equal to the threads available on the CPU. Every worker interacts with a copy of the environment, and periodically updates the global network. Since every worker is independent, the updates happen asynchronously, guaranteeing a very diverse experience available for training.

D. Network Structure

In Figure 2B the structure of the global network is reported. The network starts with convolutional layers to process spatial dependencies. At each t, it takes as input a "frame" (f_t), the binary map with start and target cells colored in red and green, respectively. The convolutional layers are followed by a Long Short-Term Memory (LSTM) network to process temporal dependencies between consecutive frames. Finally, the LSTM net is followed by the actor and the critic layers. The actor is implemented through a softmax layer, having as many outputs as the number of actions. The critic is implemented by a fully connected layer, consisting in one output neuron having linear activation function. Actor and critic weights on the network are defined as $\theta_\mathbf{v}$ and $\theta_\mathbf{l}$, respectively. Each worker gets a copy of the global network. For convenience, actor and critic weights for a specific worker will be named $\theta'_\mathbf{v}$ and $\theta'_\mathbf{l}$, respectively. Each worker interacts with the environment and collects experience, storing, in a buffer $\mathbf{B}$, a transition for each t ($\mathbf{B} = \{s_i, a_i, r_i, s_{i+1}\}, i = 0..,t$). Once the worker's experience history is large enough ($t = t_{max}$), the buffer $\mathbf{B}$ is used to compute the sum of discounted rewards (R_t) and advantage ($ADV(s_t, a_t)$), which, in turn, are used to calculate value loss, L_v:

$$L_v = \sum (R_t - V(s_t, \theta'_\mathbf{v}))^2 \tag{6}$$

and policy loss, L_p:

$$L_p = -log(\pi(s_t, \theta_l')) \cdot ADV(s_t, a, \theta_v', \theta_l') - \beta \cdot \mathbb{H}(\pi(s_t, \theta_l')) \quad (7)$$

L_p contains an entropy term ($\mathbb{H}$) with β in (0,1], in order to improve exploration by discouraging premature convergence to suboptimal deterministic policies[29]. The worker then uses L_p and L_v to compute the gradients $\Delta_{\theta_l'}$ and $\Delta_{\theta_v'}$:

$$\Delta_{\theta_v'} = \nabla_{\theta_v'}(L_v) \text{ and } \Delta_{\theta_l'} = \nabla_{\theta_l'}(L_p) \quad (9)(10)$$

and use them to update the global network parameters. Once the global network is updated, the worker resets its own weights to the ones of the global network, the buffer **B** is emptied, t is reinitialized to 0, and exploration is restarted. The pseudocode for the algorithm is presented in Algorithm 1.

Algorithm 1 A3C - pseudocode for each actor-learner thread.

// Assume global network actor-critic weights θ_l and θ_v
// Assume worker specific actor-critic weights θ_l' and θ_v'
// Assume global counter T
Initialize thread step counter $t \leftarrow 1$

repeat
  Reset gradients: $d\theta_l \leftarrow 0$ and $d\theta_v \leftarrow 0$
  Synchronize thread parameters: $\theta_l' = \theta_l$ and $\theta_v' = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(s_t; \theta_l')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t+1$
    $T \leftarrow T+1$
  until terminal s_t **or** if $t - t_{start} == t_{max}$

$$R_t = \begin{cases} 0, & for\ terminal\ s_t \\ V(s_t, \theta_v'), & for\ non-terminal\ s_t \end{cases}$$

  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R_t \leftarrow r_i + \gamma R_t$
    compute gradients $\Delta_{\theta_l'}$ and $\Delta_{\theta_v'}$
    accumulate gradients wrt θ_l: $d\theta_l \leftarrow d\theta_l + \Delta_{\theta_l'}$
    accumulate gradients wrt θ_v: $d\theta_v \leftarrow d\theta_v + \Delta_{\theta_v'}$
  update of θ_l using $d\theta_l$ and of θ_v using $d\theta_v$
until $T > T_{max}$

E. Reward Function

The reward function associated with each transition is shaped in order to make the agent learn to optimize its trajectory, according to three main requirements:

1) path length minimization
2) obstacle clearance maximization
3) needles kinematic constraints

The reward r_t is defined as:

$$r(s_t, a_t) = \begin{cases} k_w \text{ target reached} \\ -k_1 \cdot D_{s_{t+1}-T} - k_2 \cdot exp(\alpha/\pi) - k_3 \cdot \frac{\beta}{\pi/2} - k_{obst} \end{cases} \quad (11)$$

A positive constant reward k_w is given upon reaching the target.

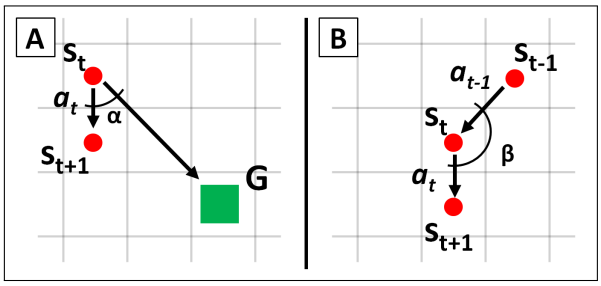

Fig. 3: **(A)** α is the angle between the current action a_t, connecting the current state s_t with the new one s_{t+1} and the vector connecting s_t and the target G. **(B)** β is the angle between the current action a_t and the previous one a_{t-1}.

In case the target is not reached, a negative reward is given according to:

- path length minimization, with a reward proportional to the distance $D_{s_{t+1}-T}$ between the new state and the target, defined as:

$$D_{s_{t+1}-T} = \sqrt{(x_{s_{t+1}} - x_g)^2 + (y_{s_{t+1}} - y_g)^2 + (z_{s_{t+1}} - z_g)^2} \quad (12)$$

- needles kinematic constraints, with two rewards given proportionally to two different angles: the angle α, between the action a_t and the vector connecting the state s_t and the target G; the angle β, between the action a_t and the previous one a_{t-1}. The two angles are shown in Figure 3.
- obstacle clearance maximization, with a constant negative reward k_{obst} given if the agent is in a cell with the minimum distance from obstacles $d_m \leq 1$ pixel/voxel .

k_w, k_1, k_2, k_3 and k_{obst} are constant values used to modulate the impact of the different terms on the total reward. In this context, the values assigned to each constant were determined using a grid-search approach and are shown in table I.

TABLE I: Reward weights

k_w	k_1	k_2	k_3	k_{obst}
6	1/32	1/5	1/2	1/2

F. Exploration Strategy

A critical aspect in reinforcement learning is balancing agent's exploration of the environment and exploitation of the gained knowledge. In this work a combination of Boltzman and Bayesian exploration was used.

Boltzman exploration [32] exploits the uncertainty contained on the estimated policy, by picking up actions according to their softmax value. An additional parameter

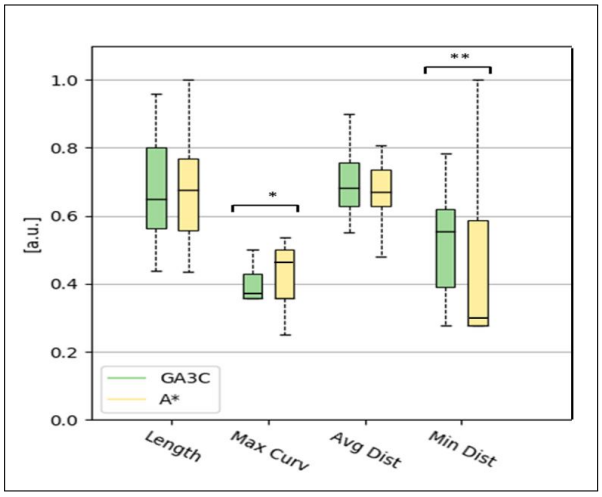

Fig. 4: The presented solution (GA3C) is tested against A* algorithm one 2D test map, not used for training, on 20 computed trajectories.The box-plots resume the obtained results in terms of path length, maximum curvature, average and minimum distance from obstacles. Each value in the box-plots is normalized in the range [0,1]. Statistical significance of Mann-Whitney U test is also reported for each feature (* $p < 0.05$, ** $p < 0.01$).

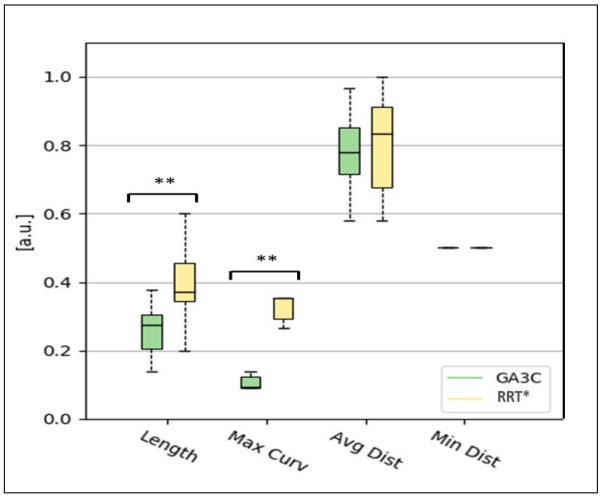

Fig. 5: The presented solution (GA3C) is tested against RRT* algorithm on the 3D test map, on 20 computed trajectories.The box-plots resume the obtained results in terms of path length, maximum curvature, average and minimum distance from obstacles. Each value in the box-plots is normalized on range [0,1]. Statistical significance of Mann-Whitney U test is also reported for each feature (* $p < 0.05$, ** $p < 0.01$)).

$\tau(t)$ is used to control the probability distribution of the softmax output, and it is annihilated over time (t), gradually turning exploration (uniform probability distribution over all the possible actions) in exploitation (high probability associated to actions with high softmax value). The Bayesian exploration exploits the agents uncertainties about its actions, as done by Bayesian Neural Networks, by introducing a dropout layer as Bayesian approximator [33]. The percentage of dropped nodes is annihilated over time, gradually turning exploration in exploitation.

G. Training Strategy

Among the 7 maps available, 6 were used during the training phase of the model. In order to train the model, 12 workers operated in parallel (one for each available thread in the CPU). At every episode a new starting point was randomly chosen among the available free cells, and a new target point was chosen. Additionally, at every episode, each agent was assigned a different map among the 6 available for training, in order to encourage its ability to learn the correct policy, independently from the specific environment, starting point or target point. The same strategy was followed to train both the 2D and the 3D models.

IV. EXPERIMENTAL RESULTS

A. Hardware Specification

We performed our experiments on a Linux machine equipped with a 6-core i7 CPU, 16GB of RAM and 1 NVIDIA Titan XP GPU with 12GB of VRAM.

B. Experimental Protocol

The same experimental protocol was followed for both the 2D and the 3D models. Experiments were carried out with "leave-one-out" cross-validation approach. The model was trained, in turn, on 6 maps, and tested on the remaining one. A total amount of 20 couples of starting points (located on the skull) and target points (located inside the target area) were identified, and a trajectory for each couple was generated using the trained model. In order to assess the quality of the proposed method, the obtained trajectories were compared to the ones obtained by means of two standard algorithms: A*[8] and RRT*[12], for 2D and 3D, respectively.

The obtained trajectories were evaluated considering their length, the minimum and average distance from obstacles, and the maximum curvature. As the number of samples for each map was small, non-parametric statistics was used [34]. To evaluate differences between each pair of methods, a pairwise comparison for each feature was run, through Mann-Whitney U test ($U < 127, p < 0.05$).

C. Evaluation

1) GA3C vs A on 2D environments:* The obtained 20 trajectories were analyzed; Figure 4 shows the results on the testing map.

2) GA3C vs RRT on 3D environments:* Analogously to the 2D case, the obtained trajectories were analyzed; the obtained results are shown in Figure 5.

Tables III and IV show the computational time required to compute the trajectories, and a smoothness index, obtained by normalizing the maximum value of curvature for each path: lower values correspond to higher smoothness of the path. Both computational times and smoothness indexes are reported with the 25% and 75% quantiles, and median value.

TABLE II: Computational times and Smoothness indexes on 2D test map

	Comput. Tme			**Smoothness**		
Case	25^{th}	Median	75^{th}	25^{th}	Median	75^{th}
GA3C	0.464s	0.699s	1.016s	0.358	0.358	0.500
*A**	0.009s	0.009s	0.013s	0.358	0.439	0.750

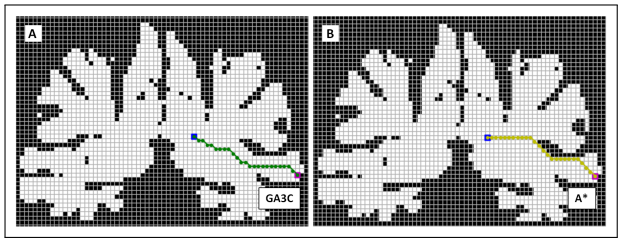

Fig. 6: Path connecting the same starting cell and target cell obtained through GA3C model **(A)** and A* algorithm **(B)** on a 2D map. Black cells correspond to obstacles, white cells correspond to free space.

TABLE III: Computational times and Smoothness indexes on 3D test map

	Comput. Tme			Smooth. Index		
Case	25^{th}	Median	75^{th}	25^{th}	Median	75^{th}
GA3C	0.087*s*	0.109*s*	0.118*s*	0.090	0.095	0.125
RRT $*$	0.05*s*	0.053*s*	0.069*s*	0.286	0.352	0.352

V. DISCUSSION

In this paper we presented a grid path planning method using GA3C Deep Reinforcement Learning, and we tested it in the context of minimally invasive neurosurgery.

When compared to A* and RRT* algorithms, the proposed method showed a superior behaviour, in terms of smoothness and safety of the generated trajectories. When applied to 2D and compared to A* (Figure 4), the proposed method was able to generate trajectories with an higher minimum distance from critical structures. The maximum curvature reached was globally lower, making the paths suitable to meet kinematic constraints of the PBN, as proved by the smoothing index in Table II. Finally, the length was in general comparable (average difference: 0.73, standard deviation: 2.32).

Figure 6 shows a comparison between trajectories generated by GA3C and A* on a 2D map, highlighting the higher smoothness and obstacle clearance of the trajectory obtained with the proposed method. When applied to 3D and compared to RRT* (Figure 5), the proposed method was able to generate trajectories with significantly lower length and significantly higher smoothness (Table III). Regarding the minimum distance value, we can observe that the paths are always at a feasible secure distance from the obstacles. However, the 3D environment introduces far more obstacles than a 2D environment and forces the path minimum distance to be lower. Figure 6 shows a comparison between trajectories generated by GA3C and RRT* on a 3D map, highlighting the higher smoothness and lower length of the GA3C generated trajectory, with respect to RRT*.

The computational times, higher with respect to A* and RRT* (Tables III and IV), are motivated by the more complex optimization of the trajectories required to the proposed model, not performed by A* and RRT*. However, when compared to methods involving subsequent refinement steps, our method performs significantly faster [26] [28].

In addition to this, the learning-based nature of the proposed approach, offers several advantages with respect to graph-based and sampling-based methods: the proposed model can be continuously improved, by retraining it with new maps, making it learn from new unseen data; it is flexible with respect to optimization strategies, allowing to menage the trade-off between different requested features (e.g. accepting to increase the insertion length, to maximize the clearance from safety regions, or vice versa), depending on the specific need; it could be trained to adapt to dynamic environments, as a real brain, where the interaction of the needle with the tissues, and the resulting deformations, may require the ability to continuously recompute the optimal trajectory.

Despite the good results on the considered maps, the models occasionally fail to generate optimal trajectories when dealing with new maps with severe differences from the ones they were trained on. This limitation is due to the reduced number of maps used for training, and could be tackled by increasing it, allowing the agent to learn how to deal with higher variability of the environment.

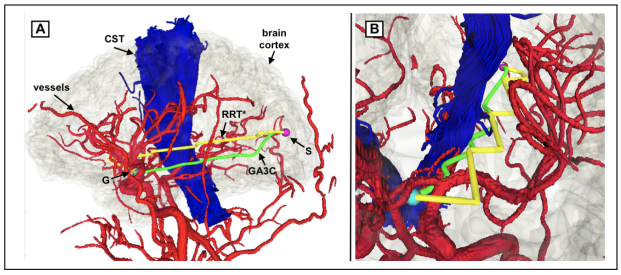

Fig. 7: Path connecting a starting cell (S) and an target cell (G) obtained through GA3C model in green and RRT* algorithm in yellow, on a Slicer model of a 3D map. Corticospinal tracts (CST) and vessels are shown in blue and red, respectively. **(A)** shows a sagittal view of the whole brain; **(B)** shows a close-up, highlighting the higher smoothness of the GA3C generated trajectory, with respect to RRT*.

VI. CONCLUSIONS

The present work proposes a novel automatic planner for steerable needles in keyhole-neurosurgery. Given a model of brain, a surgeon-defined starting point, and a target, the proposed method can provide an optimal trajectory, according to predefined features as insertion length, clearance from safety regions, as blood vessels and corticospinal tracts, and compliance to needle's kinematics limits. The model is based on a GA3C algorithm [29] [30], trained exploiting multiple workers running in parallel on continuously changing maps and targets, in order to guarantee high generalization in learning. The optimization of the trajectories, is obtained by properly shaping the reward function and tuning the weight coefficients present in it, thus determining the trade-off between different features. When tested against the standard A* and RRT* algorithms, the proposed method performed better in terms of trajectories smoothness and clearance from safety regions, with significantly increasing length. By simultaneously optimizing trajectories according to all the requested features, and not by subsequent refinements, the proposed method permits to obtain an higher accuracy, with a sensibly lower computational time.

REFERENCES

[1] L. Joskowicz, "Advances in image-guided targeting for keyhole neurosurgery: a survey paper," *Touch Briefings reports, Future directions in surgery*, vol. 2, 2006.

[2] 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.

[3] C. Burrows, R. Secoli, and F. R. y Baena, "Experimental characterisation of a biologically inspired 3d steering needle," in *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*. IEEE, 2013, pp. 1252–1257.

[4] V. Duindam, R. Alterovitz, S. Sastry, and K. Goldberg, "Screw-based motion planning for bevel-tip flexible needles in 3d environments with obstacles," in *2008 IEEE international conference on robotics and automation*. IEEE, 2008, pp. 2483–2488.

[5] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[6] P. Li, S. Jiang, J. Yang, and Z. Yang, "A combination method of artificial potential field and improved conjugate gradient for trajectory planning for needle insertion into soft tissue," *J Med Biol Eng*, vol. 34, no. 6, pp. 568–573, 2014.

[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[9] 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 *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 4600–4605.

[10] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[11] S. M. LaValle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," 2000.

[12] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," 2013.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

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

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[17] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[18] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell, *et al.*, "Learning to navigate in cities without a map," in *Advances in Neural Information Processing Systems*, 2018, pp. 2419–2430.

[19] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[20] A. I. Panov, K. S. Yakovlev, and R. Suvorov, "Grid path planning with deep reinforcement learning: Preliminary results," *Procedia computer science*, vol. 123, pp. 347–353, 2018.

[21] Y. Kato, K. Kamiyama, and K. Morioka, "Autonomous robot navigation system with learning based on deep q-network and topological maps," in *2017 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2017, pp. 1040–1046.

[22] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5113–5120.

[23] L. Zuo, Q. Guo, X. Xu, and H. Fu, "A hierarchical path planning approach based on a and least-squares policy iteration for mobile robots," *Neurocomputing*, vol. 170, pp. 257–266, 2015.

[24] 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 *2008 IEEE international conference on automation science and engineering*. IEEE, 2008, pp. 41–46.

[25] A. Kuntz, L. G. Torres, R. H. Feins, R. J. Webster, and R. Alterovitz, "Motion planning for a three-stage multilumen transoral lung access system," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3255–3261.

[26] A. Favaro, L. Cerri, S. Galvan, F. R. Y. Baena, and E. De Momi, "Automatic optimized 3d path planner for steerable catheters with heuristic search and uncertainty tolerance," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 9–16.

[27] E. Garyfallidis, M. Brett, B. Amirbekian, A. Rokem, S. Van Der Walt, M. Descoteaux, and I. Nimmo-Smith, "Dipy, a library for the analysis of diffusion mri data," *Frontiers in neuroinformatics*, vol. 8, p. 8, 2014.

[28] A. Segato, P. Valentina, A. Favaro, R. Marco, F. Andrea, E. De Momi, C. Antonella, *et al.*, "Automated steerable path planning for deep brain stimulation safeguarding fiber tracts and deep grey matter nuclei," 2019.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[30] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Ga3c: Gpu-based a3c for deep reinforcement learning," *CoRR abs/1611.06256*, 2016.

[31] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[32] L. Pan, Q. Cai, Q. Meng, L. Huang, and T.-Y. Liu, "Reinforcement learning with dynamic boltzmann softmax updates," *arXiv preprint arXiv:1903.05926*, 2019.

[33] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[34] L. VAJANI, "Probability and statistical-inference-hogg, rv, tanis, ea," 1978.