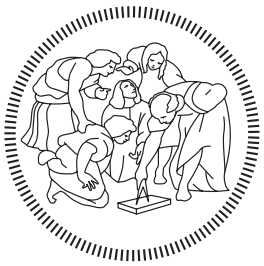
POLITECNICO DI MILANO
School of Industrial and Information Engineering
Department of Electronics, Information and Bioengineering
Master of Science in Biomedical Engineering



Surgical Path Planner for Steerable Catheters with Reinforcement Learning Approach

NearLab
Neuroengineering and medical robotics Laboratory
of Politecnico di Milano

Supervisor: Prof. Elena De Momi
Co-supervisor: Dott. Ing. Alice Segato

Master Thesis of:
Sestini Luca, 897026

Academic Year 2018-2019

To My Beloved Parents.

Index

Abbreviations

A2C Advantage Actor-Critic

A3C Asynchronous Advantage Actor-Critic

AC Actor-Critic

CST Corticospinal Tracts

DRL Deep Reinforcement Learning

EDEN2020 Enhanced Delivery Ecosystem for Neurosurgery in 2020

GA3C Asynchronous Advantage Actor-Critic on a GPU

GBM Glioblastoma Multiforme

KN Keyhole Neurosurgery

LSTM Long Short-Time Memory

MIN Minimally Invasive Neurosurgery

MIS Minimally Invasive Surgery

MR Magnetic Resonance

NNs Neural Networks

PBN "Programmable Bevel-Tip" Needle

RL Reinforcement Learning

RNN Recurrent Neural Network

RRT Rapidly Exploring Random Tree

SGD Stochastic Gradient Descent

List of Figures

List of Tables

Sommario

Negli ultimi anni, la richiesta di procedure chirurgiche meno invasive è aumentata sostanzialmente. La chirurgia mini-invasiva permette di intervenire minimizzando l'entità delle incisioni e il danneggiamento di tessuti sani, riducendo, di conseguenza i tempi di guarigione e i conseguenti rischi di infezione. Ciò risulta in procedure più sicure, con sostanziale riduzione dei rischi post-operatori, maggior comfort per il paziente e riduzione dei tempi di ricovero in ospedale. Con il termine *"Keyhole Neurosurgery"* (KN) si indica un esempio di chirurgia mini-invasiva eseguita attraverso una piccola incisione alla base del cranio. Attraverso l'incisione, un catetere può essere inserito all'interno del cranio per effettuare biopsie o trattamenti specifici. Questo lavoro si concentra su un'applicazione di KN per il trattamento del Glioblastoma multiforme, una delle forme di gliomi più aggressiva e tutt'ora difficile da curare. L'approccio standard per il suo trattamento consiste della resezione chirurgica; tuttavia questo approccio presenta importanti limiti, ed un'insoddisfacente probabilità di successo. Il trattamento attraverso somministrazione locale di specifici farmaci anti-tumorali, eseguita in modo mini-invasivo tramite KN, è stato recentemente proposto come alternativa.

Tuttavia, raggiungere lesioni collocate in zone profonde del cervello rappresenta tuttora un importante limite all'applicazione di KN per trattamento locale del glioblastoma. Recentemente, diversi prototipi di cateteri flessibili sono stati proposti; essi rappresentano un importante passo in avanti rispetto ai cateteri rigidi, comunemente utilizzati, richiedendo però, al contempo, una pianificazione del percorso più complessa, che deve tenere in considerazione una molteplicità di fattori normalmente assenti per cateteri rigidi. Gli algoritmi di pianificazione presenti in letteratura, sono generalmente limitati dall'impossibilità di ottimizzare direttamente le traiettorie, rispetto a tutti i parametri richiesti. Al contrario, utilizzano generalmente degli step successivi di raffinamento di una prima traiettoria grezza, richiedendo un alto tempo computazionale, e portando spesso a risultati sub-ottimali.

Lo scopo di questo lavoro è quello di contribuire allo sviluppo di un pianificatore di traiettorie curvilinee per cateteri flessibili, che possa assistere il chirurgo, in fase pre-operatoria. Al pianificatore è richiesto di stimare la miglior traiettoria curvilinea per raggiungere un target a partire da un'area di ingresso, definita dal chirurgo. In questo lavoro, l'ottimalità della traiettoria è valutata considerando la distanza minima da strutture vitali (vasi e tratti corticospinali) e la sua conformità rispetto ai limiti cinematici e geometrici del catetere flessibile.

Il metodo proposto è basato su un particolare approccio di *"Reinforcement Learning"* (RL) chiamato *"Asynchronous Advantage Actor-Critic"* (A3C), per la prima volta utilizzato in questo contesto, per le informazioni a noi note dalla letteratura. Il pianificatore esegue una segmentazione semi-automatica di risonanze magnetiche multi-modali appartenenti al paziente, per identificare ed estrarre le regioni anatomiche di interesse. I modelli così ottenuti, assieme al punto di ingresso e al target, vengono forniti in input al modello A3C, che restituisce in output la traiettoria ottimale. Il modello A3C è basato sull'utilizzo di reti neurali, e necessita di una fase di training in grado di garantirne la massima capacità di generalizzazione (il modello deve fornire la traiettoria ottimale dato un nuovo paziente, senza effettuare alcun'altra fase di training).

Il metodo sviluppato è in grado di stimare traiettorie sia nello spazio 2D che 3D, e considera, come strutture anatomiche da evitare, vasi sanguigni e tratti corticospinali. I test sono stati effettuati confrontando il pianificatore proposto con metodi standard presenti in letteratura. La qualità dei risultati è stata stabilita valutando i tre principali requisiti richiesti al pianificatore: minimizzazione della lunghezza della traiettoria calcolata (limite geonetrico del catetere), capacità di evitare gli ostacoli, mantenendo la maggior distanza possibile da tratti corticospinali e vasi sanguigni, e rispetto dei limiti cinematici del catetere.

I risultati mostrano che la soluzione proposta è capace di generare traiettorie migliori rispetto ai metodi presenti in letteratura, in accordo con una funzione di costo che tiene in considerazione i requisiti menzionati. In più, rispetto ai pianificatori standard, l'approccio *"learning-based"*, proprio

del RL, garantisce maggiore flessibilità al metodo proposto, rendendo possibile una sua futura estensione ad ambienti dinamici per un possibile utilizzo intra-operatorio.

Il lavoro è stato svolto nel contesto del progetto Europeo "Enhanced Delivery Ecosystem for Neurosurgery in 2020" (EDEN2020). Tra i suoi scopi, EDEN2020 punta a sviluppare cateteri in grado di rimanere in-situ per periodi prolungati, per il trattamento di malattie neuro-oncologiche croniche. La principale tecnologia sviluppata all'interno del progetto è il *"Programmable Bevel-tip Needle"* (PBN), un nuovo catetere flessibile con una particolare struttura fatta da quattro segmenti incastrati tra loro, che permettono il movimento nello spazio 3D con ottima destrezza.

Abstract

Over the last few decades, the demand for less invasive surgery for brain interventions has increased a lot. Minimally Invasive Surgery (MIS) allows to intervene minimizing the size of incisions and the damage to healthy tissues, thus reducing the wound healing time and the associated risk of infection. All this results on safer procedures, with reduced post-operative risks and discomfort for the patient, and shorter hospitalization time. Keyhole Neurosurgery (KN) is an example of MIS performed through a very small hole in the skull, called "burr hole" or "keyhole". Through the keyhole, catheters can be inserted into the brain for biopsy and therapy. This work focuses on an application of KN for glioblastoma treatment through local drug delivery, a currently investigated approach that aims at overcoming the limitations of standard surgical resection approach.

However, reaching deep located targets still represents a major challenge, which limits the application of local treatment of glioblastoma. Recently developed prototypes of steerable catheters represent an important step forward from rigid ones, but have also introduced the need for automatic path planners, able to deal with a complex optimization process of trajectories. Path planning algorithms present in literature are generally limited when applied to steerable catheters, by the impossibility to directly optimize the trajectories, according to all the requested features. Instead, they generally use subsequent refinement steps of a raw trajectory, leading to high computational time and sub-optimal results.

The aim of this work is to contribute to the development of a surgical path planner for curvilinear trajectories, able to pre-operatively assist the surgeon to define the best surgical trajectory to perform. The path planner is expected to estimate the best curvilinear trajectory for reaching a given target from an entry area, identified by the surgeon, ensuring a high level of safety and compliance with kinematic and geometric constraints of the catheter.

The developed system is based on a Reinforcement Learning (RL) approach, named Asynchronous Advantage Actor-Critic (A3C) algorithm, applied for the first time in the context of KN path planning. The path planner performs a semi-automatic segmentation of multi-modal Magnetic Resonance (MR) images of the patient, in order to identify and extract anatomical regions of interest such as anatomical obstacles and brain cortex. The obtained models of the anatomical structures, together with the entry and target points, are fed to the A3C model, which computes the optimal trajectory connecting the two points. The A3C model is based on a deep-learning approach, using neural networks, thus requiring an accurate training phase. During the training phase, the model is expected to get a solid generalization capability, in order to be able to perform optimally on new unseen patients, without any retraining phase.

The proposed method is able to deal with both 2D and 3D path planning problems, and it considers, as safety regions, blood vessels and corticospinal tracts. The proposed solution was tested against standard path planning algorithms from literature. The quality of the performances was assessed evaluating the three main requested features: insertion length minimization (geometric constraint), obstacles clearance and kinematic feasibility.

The overall study shows that, compared to standard path planning algorithms, the proposed method was able to determine better solutions, according to a cost-function defined considering all the requested features. Additionally, with respect to standard path planning algorithms, the learning-based approach guarantees a greater flexibility to the model, making it suitable to be extended to dynamic environments and used not only pre- but also intra-operatively.

This project is carried in the context of the European project Enhanced Delivery Ecosystem for Neurosurgery in 2020 (EDEN2020). Among its various goals, EDEN2020 project aims at developing steerable probes that can remain in-situ for long periods for the treatment of chronic neuro-oncological diseases. The main technology of this project, currently under development, is PBN (Programmable Bevel-tip Needle), a new multi-segment steerable probe with a programmable bevel, able to move in 3D space with great dex-

terity.

Chapter 1

INTRODUCTION

1.1 Keyhole neurosurgery framework

Surgical treatments have sensibly improved the quality of life, allowing us to live longer and healthier. However, classical open-surgery is often a trauma for the body, with large incisions and damages to healthy tissues surrounding the targeted area. This creates risks for the patient during the intervention and also after, when the long wound healing times expose him to the risk of developing infections. In addition to this, the long hospitalization times following open-surgeries, are generally associated with high costs [1]. For these reasons, over the last few decades, new techniques have been intensively investigated, leading to a massive development of MIS.

Starting from the 1970s, when Kurt Semm, in Munich and Kiel, began to extend laparoscopy from diagnosis to surgical operations [2], an increasing number of procedures, previously treated with open surgery, started to be treated non-surgically; an important boost for MIS was represented by the development of imaging techniques and the solid introduction of image guidance in surgery procedures [3]. Another factor contributing to the success of MIS is the constant upgrading of surgical instruments, which have gone from cumbersome tools to sophisticated, automatically controlled instruments. The use of minimally invasive approaches into surgical specialties, including general surgery, urology, thoracic surgery and neurosurgery, has changed not only

the performance of specific operations but, more importantly, the strategic approach to all surgeries [4].

Focusing on Minimally Invasive Neurosurgery (MIN), much work has been done in order to increase the safety and efficacy of brain and spine surgery [5]. One important application of MIS techniques is the KN, whose application to procedures such as diagnostic biopsy, Deep Brain Stimulation , and local drug delivery for tumor treatments is currently under investigation. Keyhole-neurosurgery (KN) is an example of minimally invasive surgery performed through a very small hole in the skull, called "burr hole" or "keyhole" [6]. Through the keyhole, catheters can be inserted into the brain for biopsy and therapies, using limited-sized keyhole craniotomies. The development of MIN, and in particular KN, has introduced new challenges: in complex environments as the brain, straight trajectories are often unfeasible, requiring an undesirable compromise between targeting accuracy and safeness of the path (Figure 1.1). Modern KN is trying to substitute the use of a rigid needles with steerable ones, in order to increase their dexterity in a complex environment as the brain, allowing curvilinear, safer trajectories, and thus a better targeting of the lesion, with minimized side risks of intercepting safety areas [7]. In neurosurgery, such forbidden areas are generally represented by blood vessels, lateral ventricles, Corticospinal Tracts (CST), midbrain and cerebellum [8].

1.2 Glioblastoma treatment

A Glioma is a type of tumor that starts in the glial cells of the brain or the spine. Gliomas comprise about 30% of all brain tumors and central nervous system tumors, and 80% of all malignant brain tumors [9]. There are four grades of Glioma, each associated different treatment strategies:

- Grade I: rare in adults and typically occurring in pediatric patients. It is the least aggressive form and typically grows very slowly. Grade I tumors are relatively benign, but they might put pressure on an area of the brain, leading to symptoms.

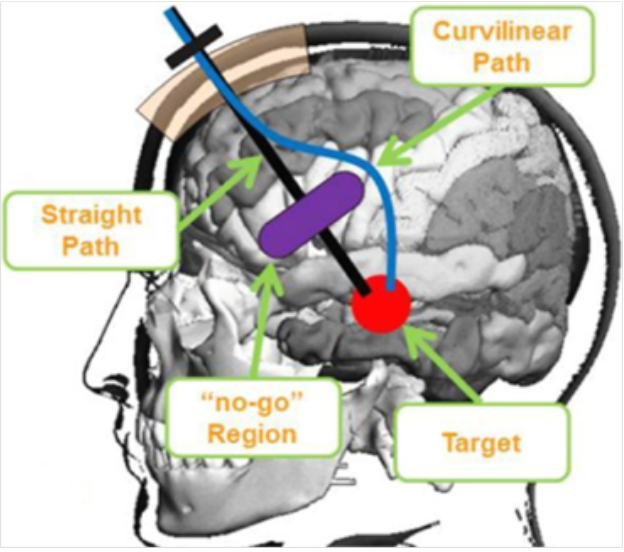


Figure 1.1: Straight path (in black) not feasible due to the presence of a safety-critical "no-go" region. Curvilinear path (in blue) instead allows to reach the target in a safe way

- Grade II: benign glioma typically found in adults, with relatively slow growth. Grade II tumors are considered benign, although they could lead to symptoms by impinging on areas of the brain, and can potentially transform into a grade III and grade IV tumors.

- Grade III: malignant tumor, for which immediate treatment is required in order to prevent growth and/or transformation into a grade IV glioma.

- Grade IV: acGBM, one of the most prevalent and malignant forms of central nervous system tumors, with a very high spreading rate.

Figure 1.2 shows the axial and sagittal views of a glioblastoma (grade IV glioma) imaged with contrast-enhanced MR.

At present, conventional therapies for these tumors are very little effective. Surgical resection is often ineffective, or even harmful. GBM has a "crab claw-like" invasion pattern, creating unclear borders between malignant and healthy tissue, thus making complete surgical resection difficult to obtain;

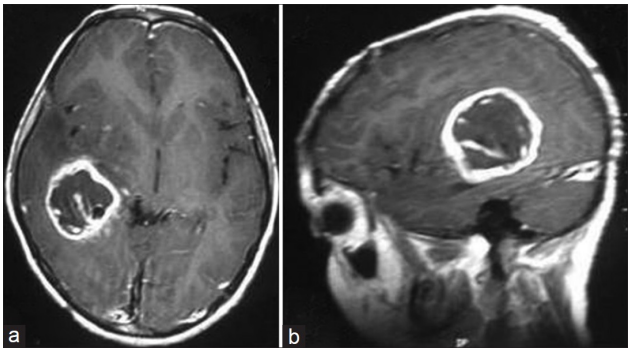


Figure 1.2: Transverse and sagittal views of a GBM in T1-weighted MR image.

additionally, the surgical procedure can stimulate the growth of malignant cells [10]. In addition to this, the central nervous system barriers (blood cerebrospinal, fluid barrier, arachnoid barrier, blood-brain barrier, blood-tumor barrier) represent a challenge to the delivery of cytotoxic drugs at therapeutic concentrations at the tumor site. This can resul in a poor cytotoxic activity and the development of drug resistance [11]. Many studies are focusing on the possibility to locally deliver drugs on the tumor, in order to maximize the efficacy of the treatment, while reducing toxicity for healthy cells [12].
Having a catheter that can move flexibly in the brain means being able to directly reach the lesion, ensuring a trajectory safe from anatomical structures as ventricles, blood vessels, corticospinal tracts, accurately targeting the tumor [13].

1.3 Steering flexible needles

In recent years many prototypes of steerable catheters have been developed for different applications in MIS.
The first steerable catheter presented is the one by Dupont [14]. It is composed by multiple pre-bent concentric stylets. The structure of steerable needles based on concentric tubes is shaped with curving segments progres-

sively decreasing in their diameters (Figure 1.3), inserted inside each other. By changing relative translation and rotation of each tube, the desired 3D trajectory can be achieved. This prototype, presented in 2012, continues to be a rich source of design, modeling, control, and sensing challenges for the research community, although some drawbacks as the overall limited curvature variability and the restrictions on achievable shapes given by the stylets preshaping.

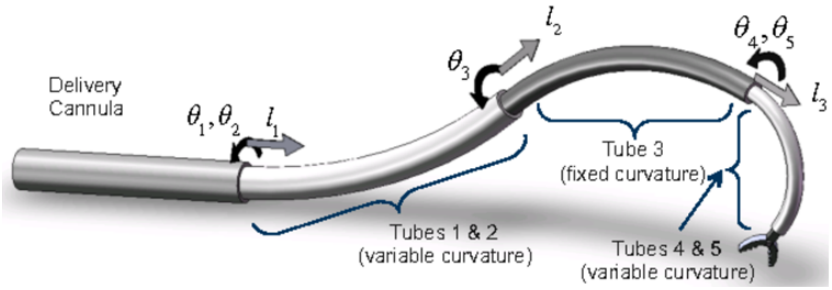


Figure 1.3: Concentric-tube robot comprising four telescoping sections that can be rotated and translated with respect to each other [14]. Each tube (1 to 5) has its own translational (l) and rotational (θ) variables (courtesy of E. Dupont, 2010).

The second solution presented is the dutycycle bevel tip one (Figure 1.4). It is based on the natural tendency of thin needles to curve toward the bevel-tip, due to the asymmetric force distribution applied by the tissue on the surface area of the tip ("Asymmetry-based steering"). In case of constant bevel angle, the direction of motion is controlled by rotating the needle at the base around its long axis, thus changing the orientation of the bevel tip with respect to the tissue. A linear relationship between duty cycle and curvature was observed across all needle materials and radii, and tissue stiffnesses. Duty-cycled rotation of a bevel tipped needle allows for proportional control of the curvature of bevel tipped needles during insertion[15]. Several different methods of needle steering have been proposed in the literature. "Base manipulation" bends the needle by applying a force perpendicular to the insertion direction. "Tissue manipulation" can move the targets and obstacles into and out of the needle's path, respectively. Each of these methods are

conceptually independent, but may be combined, thereby increasing control authority over the needle trajectory (Figure 1.4).

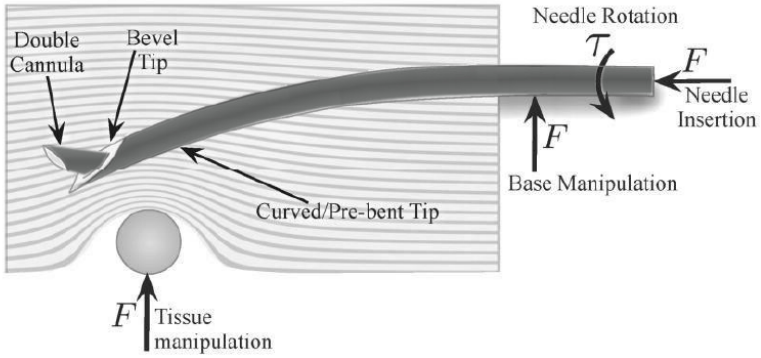


Figure 1.4: Dutycycle bevel tip solution. The needle is principally controlled by acting on the basis rotation parameter τ ("asymmetry-based steering"). Together with this other solutions have been tested: "base manipulation" exploits a force exerted at needle basis. "Tissue manipulation" exploits a force acting on the tissue (courtesy of B. Reed, 2012).

Another interesting prototype is the tendon-actuated tip implementation (Figure 1.5): the tip steering motion of the catheter is actuated in a tendon-driven manner. Two antagonistic groups of tendon actuation realize the distal tip deflecting with two-degree-of-freedom allowing it to reach a considerable large spatial workspace without catheter shaft rotation [16]. However, as many other minimally invasive surgical techniques, there remains a critical challenge to obtain sensory feedback from the end-effectors. In addition to the physical and medical constraints, such as limited workspace, biocompatibility and sterilizablity, it is even more technical demanding for the application of endoscopic surgery since the transmission path is narrow, flexible and even varying over time. Without proper position/force information at the distal end for close-loop control, existence of any positional errors such as tendon elongation and motion backlash could significantly deteriorate the system performance. As a result, it would require surgeon to continuously adjust the inputs in order to correct the errors based on visual feedback throughout the surgery. This might impair the user experience and distract

the surgeon's concentration, leading to potential prolonged operation time or safety risks [17].

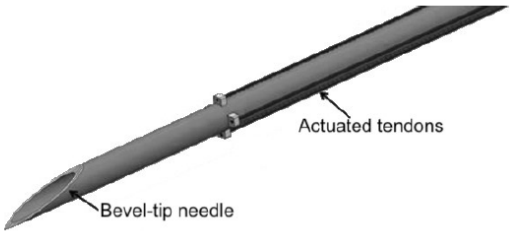


Figure 1.5: Tendon-actuated tip needle (courtesy of Konh, 2015).

Finally one of the latest developed prototypes is the multi-segment steerable probe with a programmable bevel (Figure 1.6). "Programmable Bevel-Tip" Needle (PBN) has a biologically inspired design that reproduces the multi-segment ovispositor (or egg laying channel) of certain parasitic wasps [18]. The needle is made out of four axially interlocked segments, which are able to slide along one another and are actuated independently by an actuation box able to generate four independent, linear motions. All segments possess a flat bevel tip, inclined by a certain fixed angle from the needle centre axis. By sliding on each others, the four interlocked segments create on offset. The offset is used to generate oriented shear forces with the surrounding tissues, which, in turn, are exploited to obtain steering in 3D space. Previously described solutions (concentric-tubes, duty-cycle bevel tip, tendon-actuated), when compared to straight needles present some drawbacks, including the potential to increase the extent of tissue trauma at the needle interface. The PBN, just presented above, seems to alleviate this issue [19], being able to steer without the need for duty-cycle spinning along the insertion axis or any active mechanisms at the tip.

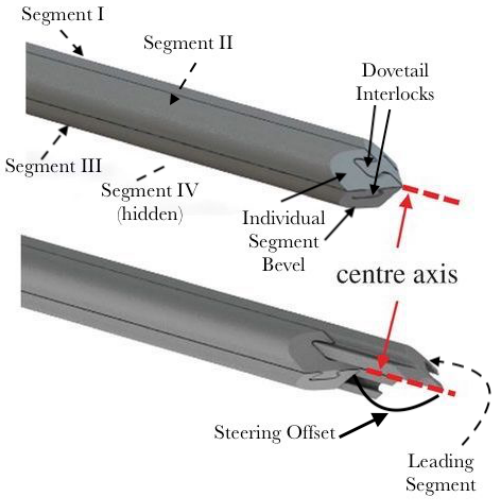


Figure 1.6: Interlocked segments steerable catheter [18]. Segments I-IV can slide on each other. The most advanced one takes the name of "leading segment": its offset with respect to the other segments is one of the variables that control needle motion. (courtesy of Leibinger, 2016).

In its current embodiment, the needle is made of plastic and can steer in three dimensions without duty-cycle spinning along the insertion axis, which has recently been shown to significantly reduce tissue deformation as a result of the insertion process [20] [21]. PBN steering is performed using a combination of pushing and rotation of the needle's segments, along their insertion axis. Figure 1.7 shows a cross section of the design, which is symmetrical about 4 axes. Each segment has two lumens: one is used as a working channel, possibly containing sensors, as optical fibers or electromagnetic trackers, or cannullas for local drug delivery; the other is employed to anchor the transmission link needed to actuate the movement.

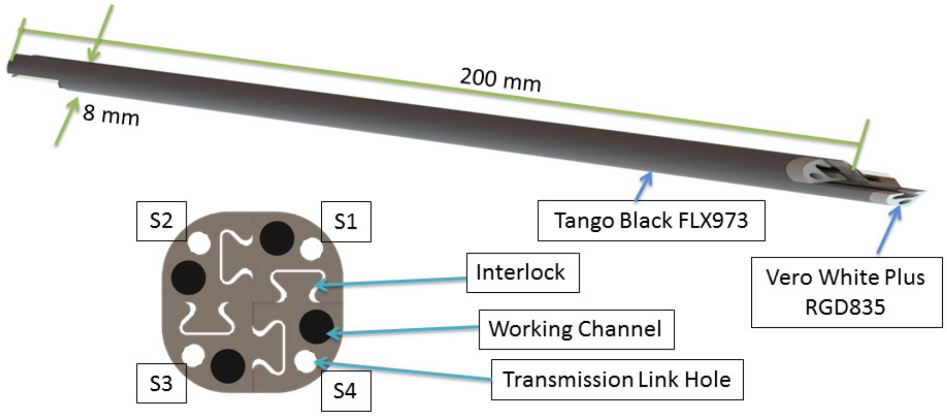


Figure 1.7: Multi-segment catheter design and front view cross-section [19]. The cross-section highlights the Dovetail interlocks, allowing reciprocal sliding, the working channels, and the transmission link holes. (courtesy of Secoli, 2018).

The detailed kinematic model of the four segments steerable needle is illustrated in Figure 1.8 and described in the following equation:

$$
\begin{aligned}
\dot{x}(t) &= \cos(\theta)\cos(\psi)v_1 \\
\dot{y}(t) &= \cos(\theta)\sin(\psi)v_1 \\
\dot{y}(t) &= -\sin(\theta)v_1 \\
\dot{\theta}(t) &= k_1\delta_{prZ}v_1 \\
\dot{\psi}(t) &= k_1\delta_{prY}v_1 \\
\delta_{prZ} &= v_2 \\
\delta_{prY} &= v_3
\end{aligned}
$$

The system defined has 3 inputs: the insertion cruise speed v_1 and the change of projected steering offsets ($\delta_{prY,Z}$) along the normal and osculating planes, represented by v_2 and v_3 respectively, which cause the needle to steer along a predetermined direction by a prescribed amount. The functions $k_{1,2}$ are experimentally derived and considered to be constant. Angles θ and ψ are defined in Figure 1.8. A high level controller, which linearises the

kinematic system by means of chained-form transformation, generates the references for a low level controller. This latter performs the actuation of the 4 segments, which work together to produce a specific tip orientation and prescribed steering offset [22].

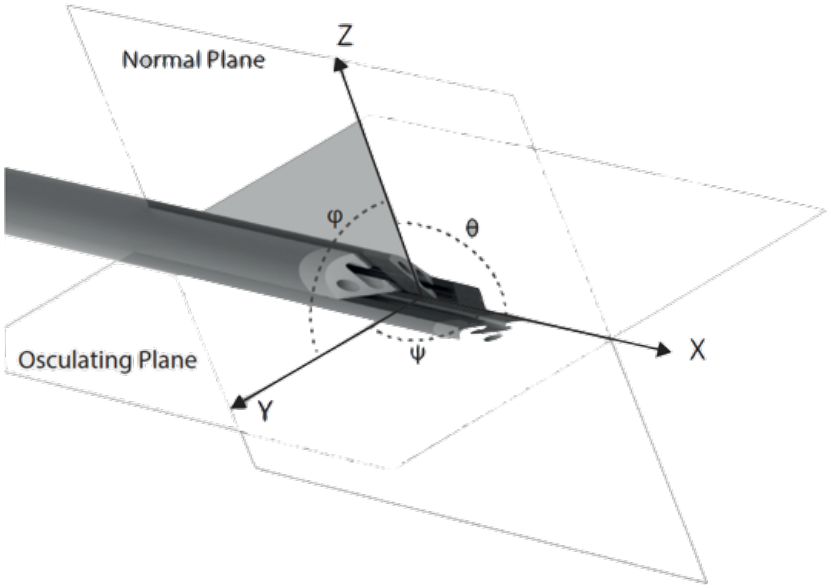


Figure 1.8: Kinematic view of a four-part needle, highlighting the osculating and normal planes, and the inclination angles θ, ψ and δ of the needle-mounted reference frame [22] (courtesy of Secoli, 2014).

1.4 EDEN2020 project overview

This work was developed in the context of EDEN2020, which aims at providing a significant change in the treatment of brain diseases, by developing an integrated technology platform for minimally invasive surgery. In doing so, the project attempts to integrate different technologies in a single surgical follow-up: pre-operative and diffusion MRI, intra-operative ultrasounds, robotic assisted catheter steering, brain drug diffusion modeling and a robotics assisted neurosurgical robotic product (Neuromate Renishaw) [23]. The project aims at achieving different objectives:

1. To engineer robotically deployable steerable catheters, that can remain

in situ for long periods for the treatment of chronic neuro-oncological diseases.

2. To enhance autonomy in monitoring robotic steerable catheter, surgeon cooperation, targeting proficiency and fault tolerance.

3. To achieve accuracy, precision and update rates in sensing and perceiving intraoperative changing brain anatomy.

4. To study in vivo diagnostic sensing in flexible access surgery.

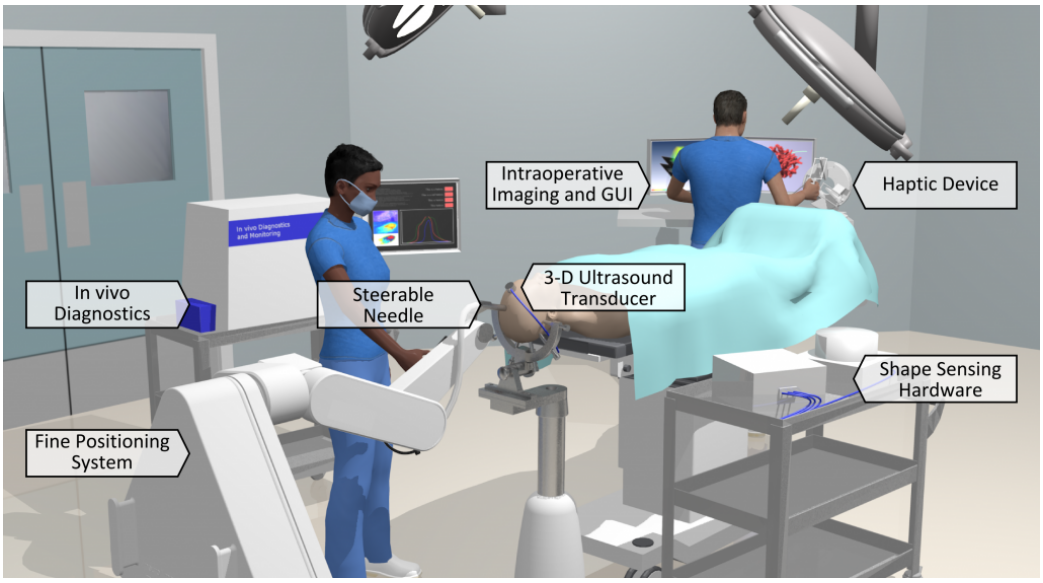


Figure 1.9: EDEN2020 platform in use, with fine positioner, and visual front end [23].

1.5 Aim of the work

Local drug delivery for treating glioblastoma, performed by means of KN, has been intensively researched in recent years, as an alternative to surgical resection. However, reaching deep located targets still represents a major challenge, which limits the application of local treatment of glioblastoma. Recently developed prototypes of steerable catheters represent an important

step forward from rigid ones, but have also introduced the need for automatic path planners, able to deal with a complex optimization process of trajectories. In fact, curvilinear planning in an environment as the brain, requires to guarantee an optimal clearance from safety regions, as blood vessels and corticospinal tracts, in order to avoid any risk of causing hemorrhages and seizures; additionally, trajectories, in order to be feasible, need to meet the kinematic and geometrical constraints of the catheter, as maximum curvature and insertion length.
In literature, while path planners for rigid catheters have been intensively studied, research on curvilinear ones is currently ongoing. The major limit of the state-of-art methods for curvilinear path planners is the impossibility to directly optimize the trajectories according to obstacle clearance, kinematic and geometrical constraints. Instead, they generally generate a raw trajectory optimized to have minimum length, and then refine it in subsequent steps to meet the other requirements. This often leads to suboptimal results.
The aim of this work was to present a path planning algorithm, able to preoperatively assist the surgeon to estimate an optimal trajectory connecting a starting point, located on the cerebral cortex and a target, located deep inside the brain. The trajectory is expected to guarantee a sufficient clearance from blood vessels and corticospinal tracts, and to meet the kinemtic and geometrical constraints of the steerable needle.

This dissertation is structured as follows:

1. Chapter 1 introduces a general background of keyhole neurosurgery and steering flexible needles, focusing on the clinical problem of glioblastoma treatment.

2. Chapter 2 summarizes the current state-of-the-art path planners for steerable needles.

3. Chapters 3 describes the overall developed system, focusing on the RL approach to the problem.

4. Chapter 4 describes the experimental protocol and summarizes the obtained results.

5. Chapter 5 discusses the results in order to assess the efficiency of the proposed curvilinear path planner.

6. Chapter 6 presents the conclusions concerning the project and describes possible future developments.

Chapter 2

STATE OF THE ART OF PATH PLANNING

2.1 Standard path planning: literature overview

2.1.1 The path planning problem

The path planning problem can be defined as the task of finding the set of subsequent positions of an agent, allowing it to reach a target point (c_{target}), starting from a starting point (c_{start}) and avoiding collisions with known obstacles. The agent in a path planning problem can be represented by a robot, a car, a catheter etc. This problem is often formalized by defining the set of possible agent configurations (U_{conf}), the space in which no configurations are allowed, due to the presence of obstacles, (U_{obst}), and the difference between these two spaces (U_{free}), which is the space of the admitted configurations [24]:

$$U_{free} = U_{conf} - Uobst \tag{2.1}$$

Therefore, solving the path planning problem means determining $\sigma(t)(t \in [0,1])$, such that:

$$\begin{cases} \sigma(t) \in U_{free} \\ \sigma(0) = c_{start} \\ \sigma(1) = c_{target} \end{cases} \tag{2.2}$$

Two important metrics to evaluate path planning algorithm approaches are the *completeness* and the *optimality*. *Completeness* refers to the capability of the algorithm of finding a solution, if one exists; *optimality* refers to the capability of the algorithm to find the optimal solution, according to the path length, among the existing ones.
In the context of path planning, a variety of approaches has been proposed in literature, many of which can be divided in three categories: artificial potential field, graph-based and search-based methods.

2.1.2 Artificial Potential Field methods

These methods try to solve the path planning problem by considering the moving agent as a point, subject to an attractive potential field generated by the target (U_{att}), and a repulsive potential field generated by the obstacles (U_{rep}) [25]. The analytical formulation for the potential fields is:

$$U_{att}(x) = \frac{1}{2}\, k_a\, \rho^2_{targ}(x) \tag{2.3}$$

$$U_{rep}(x) = \begin{cases} \frac{1}{2}\, k_r\, (\frac{1}{\rho_{obs}(x)} - \frac{1}{\rho_0})^2, & if \rho < \rho_0 \\ 0, & otherwise \end{cases} \tag{2.4}$$

where k_a and k_r are constants, x is the vector identifying the agent position, ρ_{targ} is the distance between the agent and the target, ρ is the distance from the closest obstacle and ρ_0 determines the maximum region of repulsion (far

located obstacles should not influence the agent). The related forces are obtained by computing the gradient:

$$F_{att}(x) = -\nabla U_{att}(x) \tag{2.5}$$

$$F_{rep}(x) = -\nabla U_{rep}(x) \tag{2.6}$$

From each configuration the agent is moved to the next by a force $F_{att} + F_{rep}$, which pushes it towards the target and far from the obstacles.
However, Artificial Potential Fields methods suffer from the presence of local minima of the potential field. Different solutions have been proposed in order to avoid the local minima trapping problem: Li, et al. [26] used an improved version of the conjugated gradient method in order to escape from local minima, while Park, et al. [27] located virtual obstacles around the local minimum points to prevent the robot to be trapped.

2.1.3 Graph-Based methods

In path planning context in continuous space, a common way to simplify the environment is to discretize it in a set of points (nodes), connected according to specific criteria by edges. The structure, including nodes and edges, is named "graph".
Once the space is discretized, graph-based algorithms use dynamic programming techniques, iteratively calculating the cost of the connection between nodes, and finding the best path from a start configuration to a target one minimizing this cost. Two examples of graph-based methods are Dijkstra's algorithm [28] and A* [29]. Dijkstra's algorithm aims at finding the shortest path between a node and all other nodes in the graph. As first step, all nodes are marked as unvisited and a certain *cost* is assigned to each of them (0 for c_{start}, ∞ for the others). Dijkstra's algorithm defines the *cost* of a node as a *cost-to-come*, equal to the length of the shortest obstacles-free

path, connecting the node and c_{start}, passing through visited nodes. At each iteration, the unvisited node with smallest known *cost-to-come* is considered, and the distances from all the unvisited nodes inside a neighbourhood with predefined extension is computed. *Costs-to-come* for the neighbour nodes are computed. For each node, if the new *cost-to-come* is smaller than the previously estimated, the new estimate replace the previous (a shorter path was found). After the updates, the current node is marked as visited and removed from the unvisited set. The algorithm iterates until the target is visited.

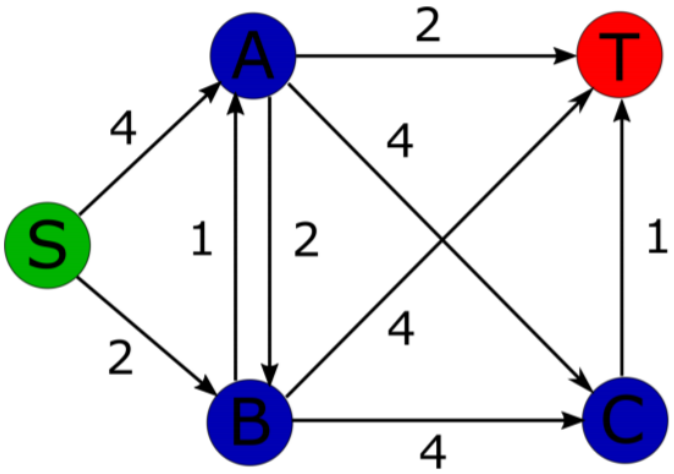


Figure 2.1: Application of Dijkstra's algorithm to a simple graph. The green node S is the start, the red one T is the target. Distances between adjacent nodes are reported.

Figure 2.1 and Table 2.1 show a simple application of Dijkstra's algorithm. A critic drawback of Dijkstra's algorithm is that it does not consider any information about the target location, thus expanding in every direction. A* is a modified version of Dijkstra's algorithm, improved to overcome the aforementioned limitation. A* algorithm takes into consideration the target location, by defining the *cost* associated with a node n as $f(n) = g(n) + h(n)$, with $g(n)$ being the *cost-to-come*, as previously defined in Dijkstra's algorithm, and $h(n)$ being an heuristic estimate of the *cost-to-go*.
Graph-based methods are *resolution-complete*, meaning that, at the chosen

Table 2.1: Steps of Dijkstra's algorithm solving the path planning problem in Figure 2.1.

Step 0	Step 1	Step 2	Step 3	Step 4
$S = 0$	$Expand\, S$	$Expand\, B_S$	$Expand\, A_B$	$End\, T_A$
$A = \infty$	$B_S = 2$	$A_B = 3$	$T_A = 5$	
$B = \infty$	$A_S = 2$	$C_B = 6$	$C_A = 7$	
$C = \infty$	$C = \infty$	$T_B = 6$		
$T = \infty$	$T = \infty$			

resolution, they are guaranteed to find a solution, if one exists. In addition, they are also *resolution-optimal*, meaning that they can find the optimal solution, among the existing ones, at the chosen resolution. Thus, optimality is guaranteed inside the limits imposed by the discretization of the domain: an higher resolution ensures an improvement in the quality of the solution, but it also increases the computational time necessary to find it. In particular, when dealing with high-dimensional spaces, the graph size and the computational time grow exponentially (*"curse of dimensionality"*) [30], requiring a reduction of the resolution, thus possibly leading to a suboptimal result.

2.1.4 Sampling-Based methods

Sampling-based methods, do not require an *a priori* discretization of the domain, but progressively sample the space, increasing the accuracy of the solution as long as the search progresses.

As opposite to graph-based methods, sampling-based methods do not explicitly characterize U_{free} and U_{obst}, but generate solutions and then check their feasibility through a collision detector. These algorithms are *probabilistic-complete*, meaning that, when the number of samples tends to ∞, they are guaranteed to find a solution, if existing.

One of the first sampling-based algorithms is Rapidly Exploring Random Tree (RRT) [31]. The algorithm starts with an initialization of the space, including only start and target points. At every iteration a new random

point is sampled in the free space, and the closest node in the growing tree is identified. If their distance is no more than a user-defined length (λ), and the connecting edge is collisions-free, the node is added to the growing tree. The research is target-oriented by occasionally sampling, as a new point, the target. If the connection satisfies the aforementioned requisites, the target is reached and the research is stopped.

RRT-connect is an enchanced version of RRT, which involves the parallel growth of two trees, one rooted on the starting point and one on the target point [32], ensuring a faster convergence. The research is focused by occasionally sampling, as a new point, the nearest point on the other tree. If the connection satisfies the aforementioned requisites, the two trees are connected (and so the starting and target node), and the research is stopped. RRT and RRT-connect are both *probabilistic-complete*, but they are not *asymptotically optimal*, meaning that they do not guarantee an asymptotical convergence to an optimal solution, as the number of samples goes to ∞.

RRT* is a modified version of RRT, improved in order to be *almost-surely asymptotically optimal* [33]. The main difference with RRT, is on how a new point is connected to the growing tree. While RRT and RRT-connect look for the closest point of the growing tree, RRT* looks for the nearby node (nodes inside a circle centered on the new point a with user-predefined radius) which minimizes the *cost-to-come*. The radius is a crucial parameter, determining a trade-off between efficiency of the search and improvement of the tree, and so of the solution. Figure 2.2 shows a comparison between RRT and RRT* exploration.

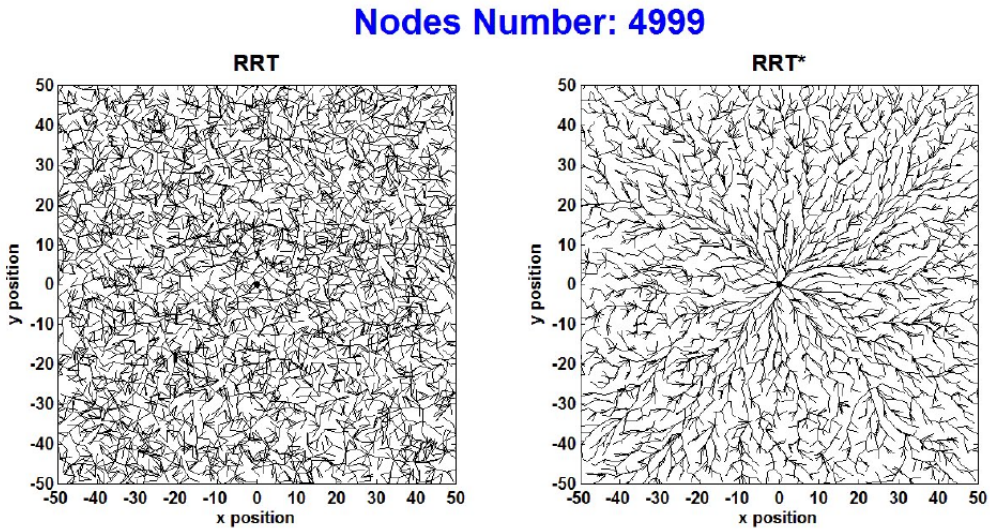


Figure 2.2: Comparison between RRT and RRT* growth of the tree. RRT*, by connecting the new point to the nearby one which minimizes the *cost-to-come*, generates smoother branches (courtesy of Dong, 2015).

2.2 Path planning algorithms for steerable needles

The application of path planning to steerable needles, in the context of MIS, introduces new crucial requirements that the computed trajectories are supposed to meet. Clearance from obstacles, not considered by the aforementioned classical path planning algorithms, becomes an essential requirement. In neurosurgery applications, as local drug delivery for glioblastoma treatment, the needle must keep an acceptable distance from cerebral blood vessels, and import brain structures as Thalamus, Ventricles, Pons and Corticospinal tracts. In addition, the obtained trajectories must be feasible for the catheter, by taking into consideration kinematic and geometrical constraints as maximum curvature and insertion length, and its non-holonomicity (the property of having a number of controllable degrees of freedom is smaller than the number of degrees of freedom of the needle). Many of the aforementioned path planning approaches have been implemented as path planners for steer-

able needles. In the context of brachytherapy procedures, Li et al. [26] 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. 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 [34]. This approach, based on the kinematic model of the needle, is able to provide the region of feasible paths, but little capability to take into account other crucial aspects as the obstacle avoidance. Graph-based and sampling based methods have been extensively explored, integrating them with different strategies in order to guarantee obstacle clearance and the compliance to kinematic constraints. Patil et al. [35] proposed an RRT-based algorithm, combined with a reachability-guided sampling heuristic (RG-RRT, Figure 2.3). This method differs from standard RRT in the choice of the new random point at each iteration: while in RRT the new point is sampled randomly from U_{free}, in RG-RRT, the allowed space for sampling is limited to the region of space accessible to the needle, according to its kinematic constraints. A similar approach was investigated by Caborni et al. [36], and tested in a neurosurgical context, limited to 2D space. Obstacle clearance represents a limitation for RG-RRT methods. The computed trajectories, in fact, are not directly optimized to guarantee clearance from safety regions, but only evaluated and ranked by means of a risk-based cost function. Favaro et al. [37] proposed a path planning algorithm exploiting multiple refinement steps, in order to refine the trajectories in order to meet the requested features (Figure 2.4). The algorithm generates a raw trajectory by means of a sampling-based approach, and iteratively add sampling points to improve it by minimizing its length. Then obstacle clearance is evaluated by considering an additional safety-margin: trajectories regarded as not safe are discarded. Finally, a smoothing phase is accomplished, in order to make trajectories compliant with needle kinematic constraints. Finally the multiple candidates obtained at this phase are ranked according to a cost function, and the best path is

selected.

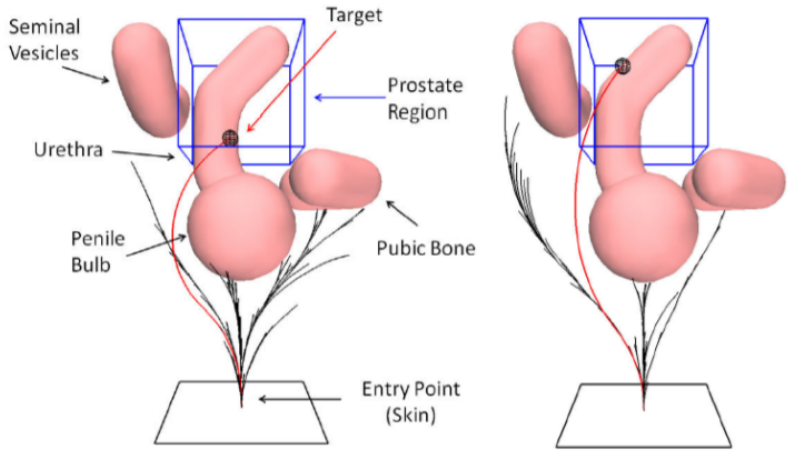


Figure 2.3: Simulation of the RG-RRT planning strategy. Given an entry point in the skin and a target in the prostate region, multiple feasible paths are obtained. Notice how all path meet the kinematic constraints of the needle thanks to the Reachability-guided approach (courtesy of Patil et al., 2010).

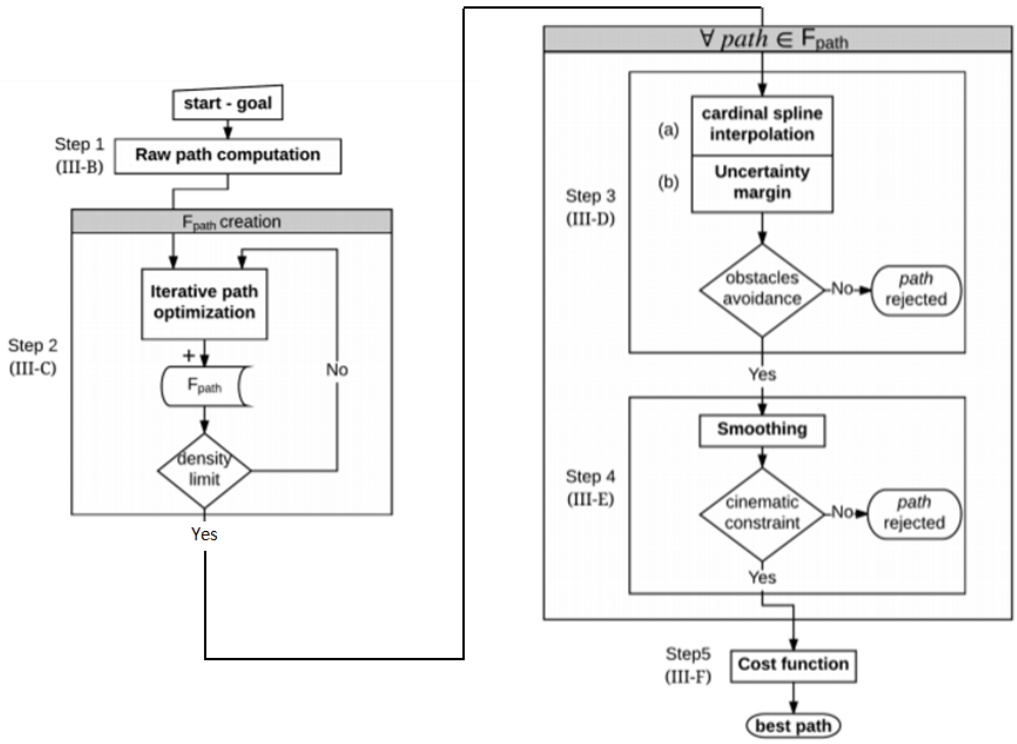


Figure 2.4: Path planning algorithm involving the computation of a first, raw trajectory, optimized in order to have minimum length, and then refined through multiple steps in order to meet obstacle clearance and smoothness requirements. (courtesy of Favaro et al., 2018).

2.2.1 Learning-based methods

Graph-based and sampling-based methods, considered the standard approaches for path planning, present some limitations when applied to steerable catheters, related to the impossibility to directly optimize the trajectory in terms of obstacle clearance and kinematic constraints. Learning-based methods, as typical of machine-learning approaches, allow to work around the problem, without requiring the explicit definition of a series of subsequent steps necessary to solve the problem. The models, instead, require a training phase during which they directly learn from data, with a *"black-box"* approach. Deep Reinforcement Learning (DRL) has recently been used in

the path planning domain.
[38, 39] demonstrate that DRL is suitable for solving path planning problems. Several studies [40, 41, 42] about applying DRL in navigation, focus on static environments, without motion or change of the environment, with promising results. [43] applied the DRL approach to a grid path planning problem, with promising results on small environments.

2.3 Thesis Objective

When applied to steerable needles, path planning algorithms need to consider important requisites as clearance from safety regions and compliance to catheter kinematic and geometrical constraints. The solutions present in literature are generally limited by the impossibility to directly optimized trajectories according to all the requested features. RG-RRT-based methods, as the on by Carboni et al. [36], for example, considers obstacle clearance only in a separate ranking phase, and not as part of an optimization process; other methods, as the one by Favaro et al. [37], start from a sampling-based method to obtain a raw trajectory, and then perform subsequent refinement steps to meet all the requirements. This, in general, increases the computational time, and often leads to sub-optimal solutions, being a refined version of an initial sub-optimal (with respect to obstacle clearance and compliance to kinematic constraints) raw solution. In addition to this, most of the proposed algorithms are limited to 2D, and often deals with simplified environments with geometrical objects.

The herein proposed algorithm is a novel path planner for steerable needles, suitable for both 2D and 3D spaces. It aims at overcoming the current limits found in literature by using a Deep Reinforcement Learning approach. The algorithm takes an input a model of the brain, a target and a starting point, and produces as output a trajectory connecting the two. The brain model is obtained via semi-automatic segmentation of multimodal MR images, and it includes blood vessels and corticospinal tracts. The optimality of the trajectory is evaluated considering the three main requested features:

clearance from obstacles, compliance to kinematic constraints and insertion length.

Chapter 3

MATERIALS AND METHODS

3.1 Workflow overview

As we have seen, the development of steerable needles has introduced many new possibilities in MIS. In glioblastoma treatment, KN offers the possibility to perform local drug delivery, currently investigated as an alternative to surgical resection, or in support to it; in addition to the specific clinical context-related advantages, MIS guarantees, in general, smaller damages to healthy tissues, reducing the risk of the intervention and the patient hospitalization time. Steerable catheters potentially allow to minimize the trade off between targeting accuracy and clearance from safety areas, imposed by traditional rigid catheters.To perform these interventions, a pre-operative phase is necessary to carefully plan the path that catheter should follow to reach the target. The aim of this project is to develop a fully automatic pre-operative planning system that allows to plan safe curvilinear trajectories, compliant to kinematic and geometric constraints of the catheter.
The proposed method is based on a Deep Reinforcement Learning approach. Differently from current solutions presented in literature the method is based on a typical machine-learning *"black-box"* approach, which does not require to define an explicit series of subsequent steps in order to obtain the trajectory. The method instead, requires the development of a dataset, on which the DRL model is trained. Once trained, the model is ready to perform on

new unseen data. Figure 3.1 summarizes the main steps involved in training. Once the model is trained it performs on new patients following the workflow summarized in Figure 3.2. Each step will be analyzed in detail in the next paragraphs.

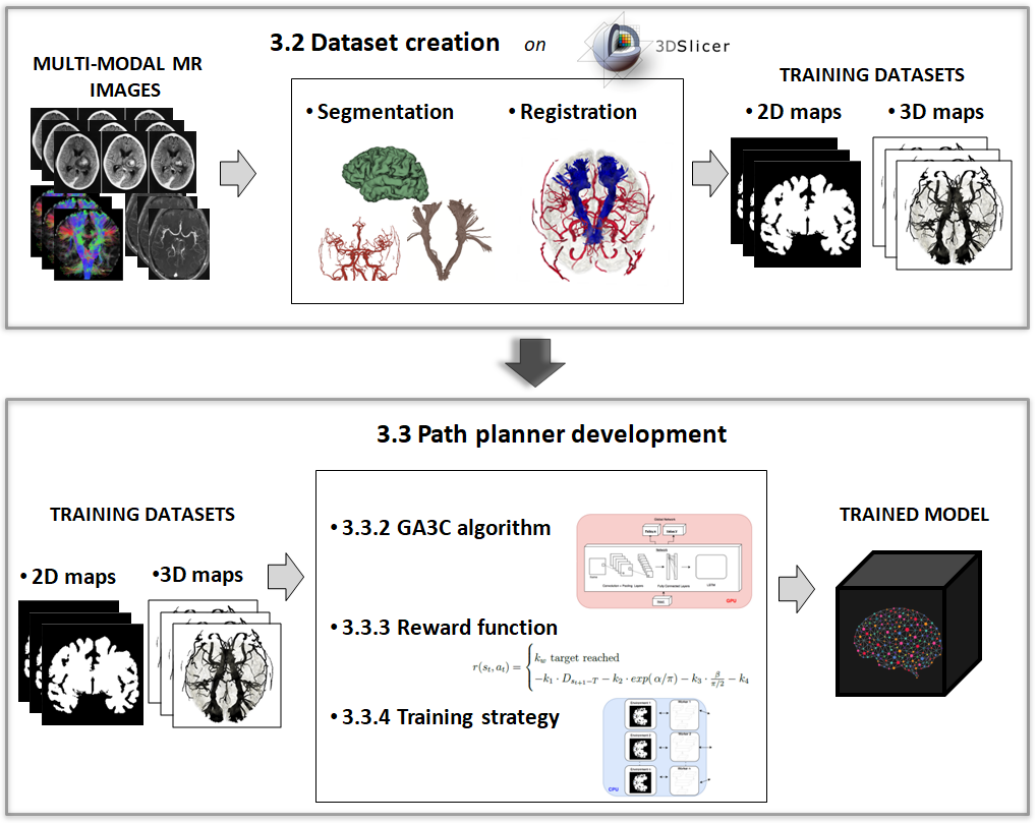


Figure 3.1: Training workflow showing: the dataset creation, starting from multi-modal MR images, involving segmentation and registration of the obtained brain cortex, blood vessels and corticospinal tracts models; 2D and 3D maps creation; development of the path planner, involving GA3C network and algorithm definition, reward function implementation, according to the requested constraints, definition of a training strategy. At the end of training the trained learning-based model is ready to be used on unseen data.

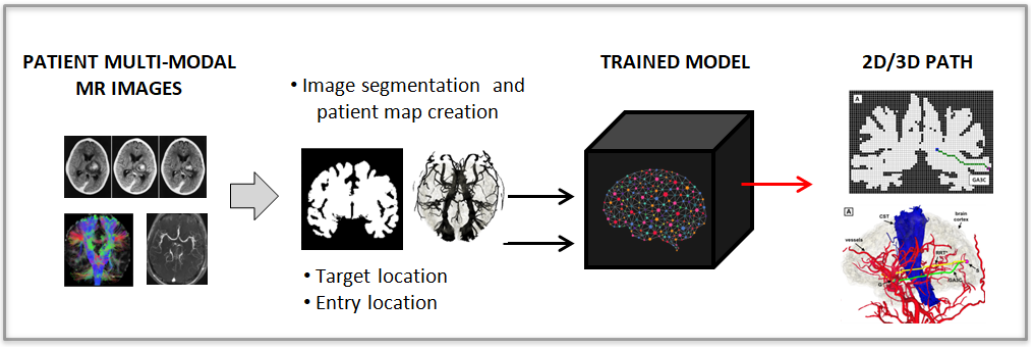


Figure 3.2: Path planner workflow: given a new patient, multi-modal MR images are acquired and then segmented and registered to obtain 2D/3D maps. The map is fed to the trained model, together with target and entry location. The model directly provides the 2D/3D path optimized in order to ensure clearance from safety areas and to be compliant to needle kinematic and geometric constraints.

3.2 Dataset creation

The training dataset was built using high-resolution multi-modal MR images. The MR images considered included:

- a 3D T1-weighted sagittal Fast-Field Echo (Figure 3.3).
- a 3D high-resolution time-of-flight MR angiography (TOF-MRA, Figure 3.4).
- a high angular resolution diffusion MR images (HARDI) with diffusion gradients applied along 60 non-collinear directions (Figure 3.5).

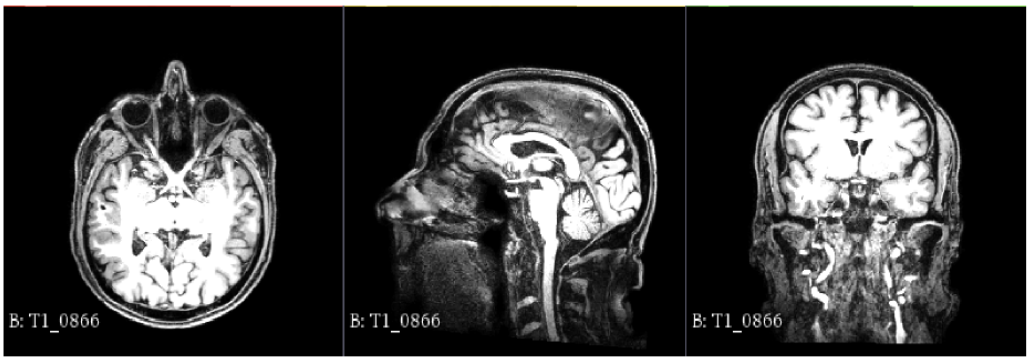


Figure 3.3: T1 weighted MR images. From left to right transverse, sagittal and coronal views are shown.

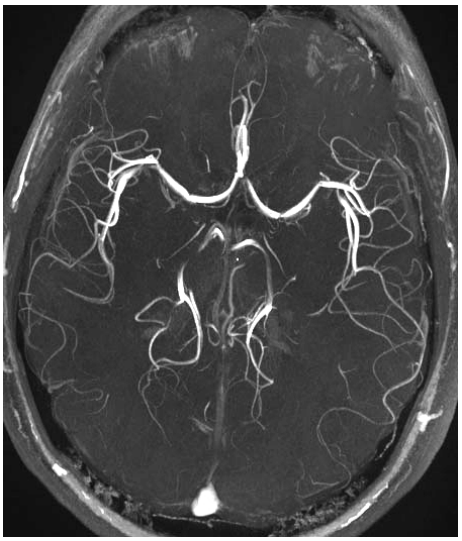


Figure 3.4: Time of Flight MR angiography, taken at the level of the Circle of Willis.

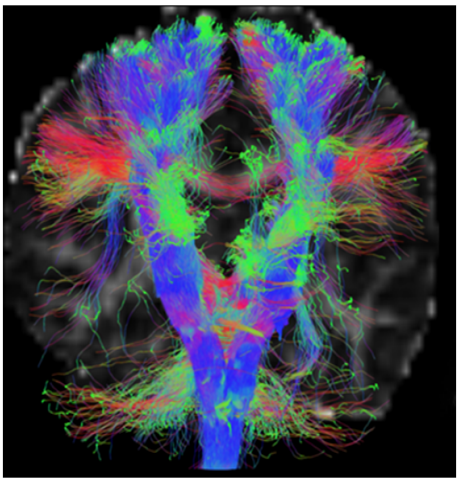


Figure 3.5: High Angolar-Resolution Diffusion Imaging highlighting corticospinal tracts (coronal view).

The ToF and the T1-weighted images were segmented by thresholding in order to obtain, blood vessels and brain models, respectively. The goal of image segmentation is to divide the starting image into a set of semantically meaningful, homogeneous, and non-overlapping regions. The segmentation result is a label map, classifying each pixel/voxel as belonging, or not, to a certain structure, which can be used to build a 3D model of the segmented structure.
Segmentation was performed by means of thresholding on voxels values. Small isolated voxel clusters ($< 100\,voxels$) were discarded, having no semantic meaning. Figure 3.6 shows the result of the segmentation step on a ToF and on a T1-weighted MR image, extracting the vessels and the brain cortex, respectively.

Figure 3.7: 3DSlicer rendering of corticospinal tracts extracted from HARDI MR image.

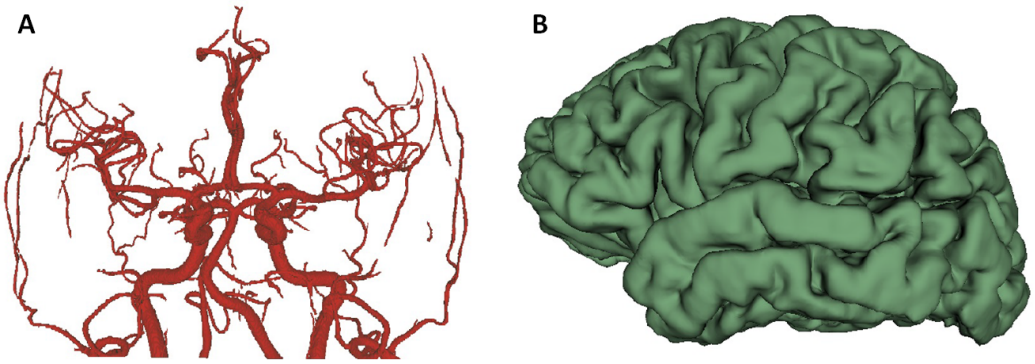


Figure 3.6: **(A)** 3DSlicer rendering of brain blood vessels extracted by thresholding segmentation of ToF image. **(B)** 3DSlicer rendering of brain cortex extracte by thresholding segmentation of T1-weighthed MR image.

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 [44] [45]. Figure 3.7 shows the result of the reconstruction of the CSTs. The three models: 1) brain cortex, 2) blood vessels and 3) CST were registered, allowing to obtain a unique model (Figure 3.8).

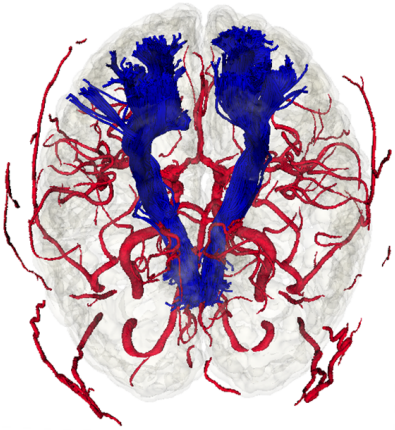


Figure 3.8: 3DSlicer rendering of brain model obtained registering brain cortex, blood vessels and corticospinal tracts models.

From the model, 3D binary label maps were created (dimension 256 $\times$ 256 $\times$ 256 mm), having zeros corresponding to regions accessible to the catheter, and ones otherwise (safety structures: blood vessels and corticospinal tracts).The label maps were used to generate the 3D maps, with each voxel corresponding to a cell (free=0 or occupied=1, depending on the label). The term "map" refers to the constructed environment. In addition, from every 3D map, a 2D map was obtained by considering the central slice, parallel to the frontal plane. Figures 3.9 and 3.10 show examples of a 2D and 3D maps, respectively.

Figure 3.9: Example of 3D map, obtained by binarizing the model shown in Figure 3.8.

Figure 3.10: Example of 2D map, obtained by considering the central slice, parallel to the frontal plane of the corresponding 3D map.

3.3 Path planner development

In collaboration with radiologists of San Raffaele Hospital, and taking into consideration PBN characteristics we identified implicit and explicit rules used by neurosurgeons when selecting a best possible trajectory, to be translated, in a second step, in numerical constraints and implemented in the path planner:

- **Maximization of clearance from anatomical obstacles.** The catheter should not intercept any vital or risky structure in his path. Although ensuring a minimal distance would work fine for this aim, the path planner is requested to optimize the paths maximizing the distance from blood vessels and corticospinal tracts, in order to better deal with possible uncertainties.

- **Minimization of path curvature.** In order to be feasible for the real catheter, the path should be compliant to its kinematic constraints. In order to ensure this, the path planner was requested to optimize the paths minimizing its curvature.

- **Minimization of path length.** In order to be feasible for the real catheter, the path should be compliant to its geometric constraints, in particular diameter and length. While diameter is a fixed parameter, taken into consideration when considering clearance from obstacles, the length constraint was explicitely addressed by requiring the path planner to minimize the insertion length.

- **Limitation of the hemisphere.** Trajectories crossing the two hemispheres would unnecessarily increase the risk of intercepting vital structures as the ventricles, without bringing any particular benefit. For this reason, choosing target and starting point located in different hemispheres was not allowed. This allowed us to consider one hemisphere at time, easing the training process.

The above mentioned constraints were implemented by properly shaping a *"reward function"*, whose role, in RL context, will be described in the following section.

3.3.1 Reinforcement Learning background

RL is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Reinforcement learning is one of the three basic machine

learning paradigms, alongside supervised and unsupervised learning. It differs from supervised learning in that labelled input/output pairs need not be presented; it also differs from unsupervised learning, which aims at finding previously unknown patterns in data sets without pre-existing labels, since it involves a continuous interactions with an external, unknown environment.
Reinforcement learning, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms.
Basic reinforcement is modeled as a Markov Decision Process, and it includes:

- An **agent**, which takes actions.
- An **environment**, the world where the agent acts.
- A **state** s_t, the concrete and immediate situation in which the agent finds itself.
- An **action** a_t, which makes the agent interact with the environment and change its state. At each t the agent decides which one to choose among a set of possible actions $\mathbf{A}$.
- A **reward** r_t, the feedback which measure the success or failure of an agent's action.

At every time step, the agent, which is in a state s_t, chooses an action a_t, according to its policy (π), such that:

$$\pi(s_t) = a_t \tag{3.1}$$

As a response, the agent receives a new state s_{t+1} from the environment, and a reward r_t (Figure 3.11). 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 of the discount factor determines

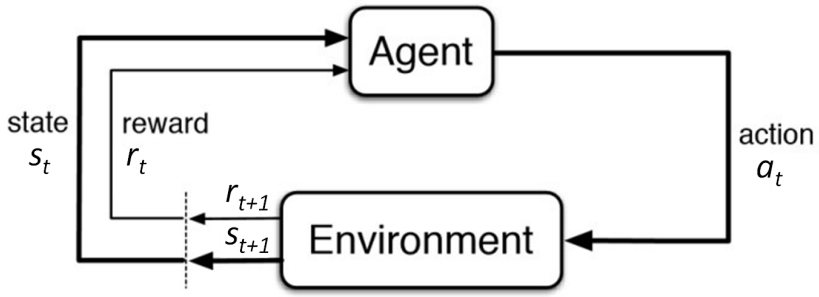


Figure 3.11: Reinforcement Learning basic flowchart showing the agent, at current state s_t, taking an action a_t and getting from the environment a new state s_{t+1} and a reward r_t.

how forward-looking the agent is: if $\gamma = 0$ the agent gives importance only to the immediate reward; if $\gamma = 1$ the agent gives the same importance to all future rewards [46].

Determining the sum of discounted rewards for a specific state can be difficult, especially for high-dimensional problems, typical when dealing with reinforcement learning. However, an estimation of it can be given by the value functions [46]. The state value function V:

$$V(s) = E[R_t|s_t = s] = E[\sum_{t'=t}^{T} \gamma^{t'-t} r^t | s_t = s] = E[r_t + \gamma V(s_{t+1})|s_t = s] \quad (3.2)$$

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

$$Q(s,a) = E[R_t|s_t = s, a_t = a] = E[\sum_{t'=t}^{T} \gamma^{t'-t} r^t | s_t = s, a_t = a] = \\ = E[r_t + \gamma \max_{a'} Q(s_{t+1}, a')|s_t = s, a_t = a] \quad (3.3)$$

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). Notice the three equivalent definitions of the value functions: the third, in particular, highlights the

separate contribute of the current reward r_t, and the estimate of the value function associated with the new state s_{t+1}. This definition has the form of a Bellman equation [47], and has a pivotal role in RL, which will be clarified later in this section.

EXPLORATION TECHNIQUES

A critical aspect in reinforcement learning is balancing agent's exploration of the environment and exploitation of the gained knowledge. Differently from the other machine learning approaches, in RL, the agent is responsible to build its own training dataset, while training on it. At each t the agent is required to take an action, and decide whether to exploit the knowledge gained so far, or to pick up a more uncertain action to increase its knowledge about the environment.
The two extreme exploration strategies consist in total exploration (random choice of actions) and total exploitation (choice, at each t, of the action maximizing the Q-value). A simple combination of the two is the epsilon-greedy approach [46], which involves choosing, at each t, an optimal action or a random one, according to a certain predefined probability. Boltzman exploration [48] is a more sophisticated exploration technique, which exploits the information contained on the estimated Q-values, by assigning to each action a weighted probability. The values are estimated by means of a softmax layer, so that the probability of each one to be chosen is proportional to its Q-value. An additional parameter $\tau(t)$ is used to control the probability distribution of the softmax output, and it is annihilated over time, gradually turning exploration (uniform probability distribution over all the possible actions) in exploitation (high probability associated to actions with high q-values). The probability distribution associated with each action is specified in eq. 3.4.

$$P(a) = \frac{exp(Q(s,a)/\tau(t))}{\sum_{i=1}^{N} exp(Q(s,a_i)/\tau(t))} \tag{3.4}$$

Another exploration strategy is the Bayesian one, which exploits the agents uncertainties about its actions, as done by Bayesian Neural Networks [49], by introducing a dropout layer as Bayesian approximator [50]. The percentage of dropped nodes is annihilated over time, gradually turning exploration in exploitation.

VALUE-FUNCTION BASED and POLICY GRADIENTS BASED ALGORITHMS

Stated that the aim of the agent is to determine the optimal policy π^*, maximizing the sum of discounted rewards R_t, different approaches can be exploited, roughly classifiable in two categories: value-function learning and policy gradients learning.
Value-function learning involves the iterative estimation of a value function, in general the state-action one ($Q(s, a)$). Optimal policy π^* can then be extracted from the estimated state-action function $Q(s, a)$, knowing that:

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{3.5}$$

where $Q^*(s, a)$ is the real state-action value function, under the optimal policy. A popular value-function based algorithm is one-step Q-learning [46] (Algorithm 1). In Algorithm 1, operation 6 corresponds to the update rule, with α being the "learning rate", and multiplying a term defined in RL context as *"Temporal-Difference error"* (TD error). In it, it is possible to recognize the expression of R_t defined in equation 3.3. The TD error corresponds to the difference between the current estimation of $Q(s_t, a_t)$ and a new, refined version which better approximates R_t. The TD error is a crucial element of RL, and will be presented again later in this section.
The Q-learning algorithm presented in Algorithm 1 is in a "tabular form". In tabular methods the state-value function $Q(s, a)$ is stored in a table, having one entry for each state-action couple. This is feasible in problems with small discrete state and action spaces; in many cases of practical interest,

Algorithm 1 Tabular one-step Q-learning pseudocode

// Randomly initialize $Q(s, a), \forall s \in \mathbf{S}, \forall a \in \mathbf{A}$

// Initialize episodes counter $T \leftarrow 0$

// Initialize steps counter $t \leftarrow 0$

repeat
  select initial state s_0
  repeat
    select action a_t according to exploration policy (e.g. ϵ-greedy)
    take action a_t, get new state s_{t+1} and reward r_t
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))]$
    $s_t \leftarrow s_{t+1}$
    $t \leftarrow t + 1$
  until $t > t_{max}$
  $T \leftarrow T + 1$
until $T > T_{max}$

however, the state and action spaces are continuous, making it unfeasible to use tables. In those cases the value functions must be approximated, using some sort of more compact parameterized function representation. A major breakthrough in reinforcement learning, was represented by the introduction of Neural Networks (NNs) [51]. NNs can deal with high dimensional continuous state and action spaces; in addition, they have great generalization capability, making training faster and more efficient.

Deep Reinforcement Learning (DRL) combining deep-learning methods, involving NNs, and reinforcement learning algorithms, has become the standard way to deal with reinforcement learning problems. In value-based deep reinforcement learning methods, the action value function is approximated by a neural network. Let $Q(s, a; \theta)$ be an approximate action-value function, with θ the weights of the neural network. The updates to θ can be derived from a variety of reinforcement learning algorithms. In one-step deep Q-learning (function approximation version of the tabular one-step Q-learning, described in Algorithm 1), the weights θ of the action value function $Q(s, a; \theta)$

are learned by iteratively minimizing a loss function L defined as:

$$L(\theta_\mathbf{t}) = E(r_t + \gamma \max_{a'} Q(s_t, a'; \theta_\mathbf{t}) - Q(s_t, a_t; \theta_\mathbf{t}))^2 \tag{3.6}$$

corresponding to a squared version of the TD error. The loss $L(\theta_\mathbf{t})$ is then used to update the weights. A basic optimization algorithm is the Stochastic Gradient Descent (SGD), with update rule defined as:

$$\theta_{\mathbf{t+1}} = \theta_\mathbf{t} - \alpha \nabla_{\theta_t} L(\theta_t) \tag{3.7}$$

with α being the learning rate. More sophisticated algorithms for optimization of stochastic objective functions are generally used, as SGD with Nesterov momentum or ADAM [52].

Policy-gradient based algorithms, differently from the previously mentioned algorithms, parametrize the policy function π as $\pi(s;\theta)$, and update the parameters θ by performing gradient ascent on $E(R_t)$. One example of policy-gradient based methodes is the REINFORCE family of algorithms [53]. Standard REINFORCE algorithms update the policy parameters θ in the direction of $\nabla_\theta log\pi(s_t;\theta)R_t$, which is an unbiased estimate of $E(R_t)$. It is possible to reduce the variance of this estimate while keeping it unbiased by subtracting a learned function of the state $b_t(s_t)$, known as "baseline", from the return [53]. The resulting gradient is $\nabla_\theta log\pi(s_t;\theta)(R_t - b_t(s_t))$. A learned estimate of the value function $V(s_t)$ is commonly used as of the baseline $b_t(s_t)$. When an approximation of $V(s_t)$ is used as baseline, the quantity $R_t - b_t(s_t)$, used to scale the policy gradient, can be seen as an approximation of the advantage function, defined as:

$$ADV(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{3.8}$$

and it results on a much lower variance of the estimated policy gradient. The presented approach, including the baseline estimation by means of the state value function, directly introduces us to the next DRL model, the actor-critic.

ACTOR-CRITIC METHOD

Algorithms using parametrized policies, as the basic REINFORCE algorithms, are called "Actor-only methods". When using only the value function approximations, as DQN, algorithms are regarded as "Critic-only methods". The combination of the two solutions takes the name of Actor-Critic (AC) approach.
In AC methods the critic learns to approximate an estimation of the return R_t, and use it to update the policy approximation of the actor (Figure 3.12). Actor and critic are implemented by means of two independent neural networks, having weights named $\theta_{\mathbf{p}}$ and $\theta_{\mathbf{v}}$, respectively. Among all the different algorithms included in the AC family, Advantage AC has a relevant role. Advantage Actor-Critic (A2C) specifically uses an estimate of the advantage function $ADV(s_t, a_t)$ as an unbiased estimate of $E(R_t)$ with subtracted baseline $b_t(s_t)$, in order to update the actor parameters $\theta_{\mathbf{p}}$. Parallelly, the critic updates its parameters $\theta_{\mathbf{v}}$, exploiting the TD error, as done by Critic-only methods.

3.3.2 GA3C algorithm

The Asynchronous Advantage Actor-Critic method, popularly called A3C, is a highly computational efficient way of utilizing the A2C approach [54]. Analogously to the previously described A2C algorithm, A3C maintains a separate memory structure to explicitly represent the policy independently from the value function, with weights named $\theta_{\mathbf{p}}$ and $\theta_{\mathbf{v}}$, respectively, and uses an estimate of the advantage function in order to update the actor parameters estimating the policy. The computational efficiency comes from the asynchronous nature of the algorithm. The model consists of a global network, processing the state in input and outputting the state value function $V(s_t; \theta_{\mathbf{v}})$ (critic) and the policy $\pi(s_t; \theta_{\mathbf{p}})$ (actor); copies of the global network are assigned to independent agents, named "workers", whose num-

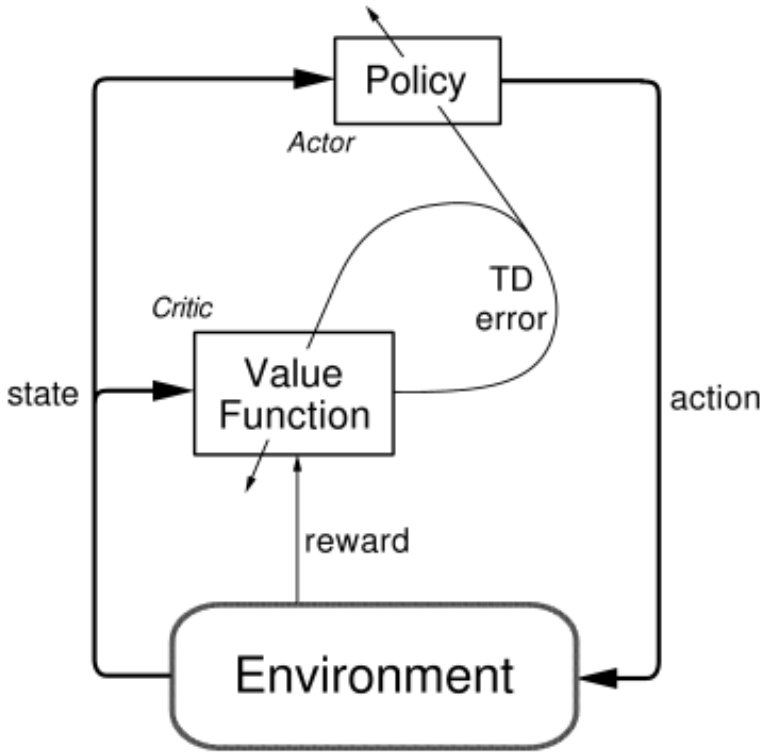


Figure 3.12: Schematic representation of Actor-Critic algorithm: at each t the agent takes an action from a certain state and gets a new state and a reward. The reward is used by the critic to compute a TD error, which in turn is used to update the critic itself and the actor. (courtesy of Sutton & Barto, 1992).

ber is, at maximum, equal to the number of threads available in the CPU. Every worker interacts with a copy of the environment, and periodically updates the global network. Multiple actors-learners acting in parallel increment exploration efficiency, by exploring different parts of the environments, eventually using different exploration techniques. Additionally, by running multiple independent workers in parallel, the overall changes being applied to the weights of the global network are likely to be less correlated in time, than a single agent applying online updates. In addition to stabilizing learning, the asynchronous action of the workers guarantees a reduction in training time, almost linearly proportional to their number.
In this work, the reinforcement learning problem is addressed with the Asynchronous Advantage Actor-Critic on a GPU (GA3C) algorithm [55], a hybrid CPU/GPU implementation of the standard A3C algorithm.

PATH PLANNING PROBLEM FORMULATION IN REINFORCEMENT LEARNING CONTEXT

The path planning problem is formulated in RL terms. The agent is the tip of the catheter. Since the needle advances and curves with a continuous trajectory, the path travelled by the object can be an be approximated as a *"follow the leader"* path. In this configuration, the distal part of the needle follows its tip and remarks its travelled points.
The agent operates on an grid, static environment composed of free cells and occupied cells, corresponding to obstacles. The grid environment is the binary label map obtained after segmentation step, describe in section 3.2. Every cell is a state, described by its coordinates in the image reference frame. From each 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, 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. The constraints described in section 3.3 are imple-

mented by properly defining the reward function, and will be described later in this section.

NETWORK STRUCTURE

In Figure 3.13 the structure of the global network is reported.

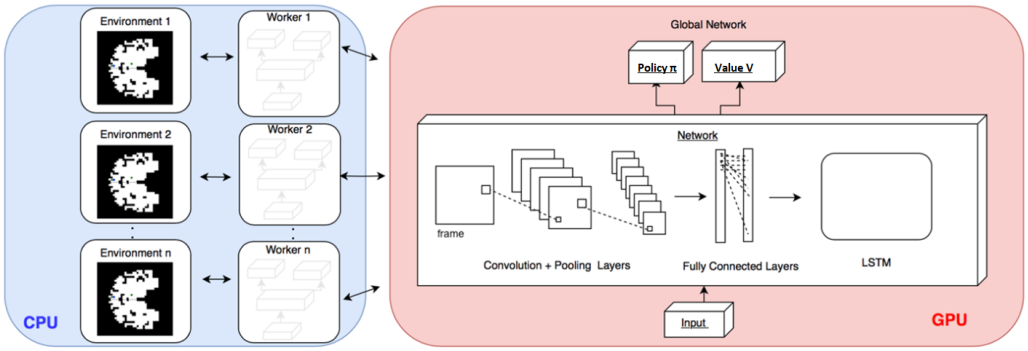


Figure 3.13: Architecture of the network used by GA3C. The pink box contains the global neural network, having the current frame (f_t) as input and the policy π and the value V as outputs, and running on the GPU. The blue box contains the n workers running in parallel, each one with a copy of the global network, and periodically interacting with it.

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 activation function used was leaky Rectified Linear Unit (leaky ReLU) [56].
The convolutional layers are followed by a Long Short-Time Memory (LSTM) network to process temporal dependencies between consecutive frames [57]. LSTM are a specific kind of Recurrent Neural Network (RNN). RNNs are a kind of NN able to model time dependencies, due to the presence of connections between output and input, which allow information to persist. Figure 3.14 shows a representation of a RNN, highlighting the real network structure, and the dynamic representation.

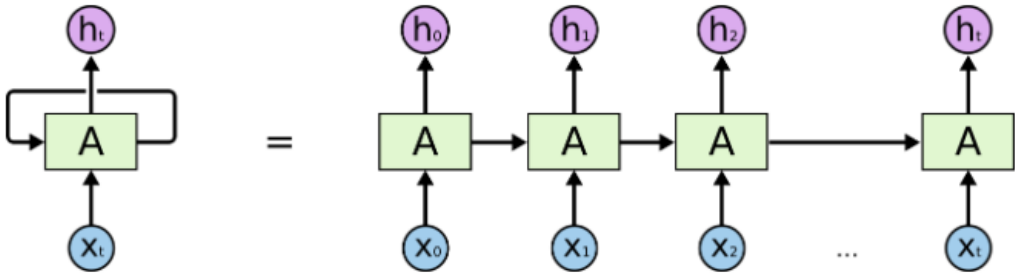


Figure 3.14: RNN network structure: real architecture and dynamic representation. A is the basic cell equal at each t. x_t are the inputs, h_t are the outputs at each t.

Multiple tasks as speech recognition, language modelling and translation, require the network to handle both long- and short- time dependencies. Basic RNNs often struggle with long-term dependencies [58]. LSTM networks are explicitly designed to avoid the long-term dependency problem. Figure 3.15 shows a comparison between the dynamic representations of RNNs and LSTM networks. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, the repeating module have a simple structure, such as a single layer with hyperbolic tangent activation function (*tanh*) (Figure 3.15A). The output at previous timestep t h_{n-1} is concatenated with the current input x_t, and fed to the *tanh* layer, which produces the new output h_t. LSTMs also have this chain like structure, but the repeating cell has a different structure. Instead of having a single neural network layer, there are four, interacting with each other. In LSTM networks every cell gets the new external input x_t, and two inputs from the previous one: the previous output h_{t-1}, as for RNNs, and an additional term C_{t-1}, called *"cell state"*. The cell state term flows through all the cells, undergoing updates involving "forgetting" obsolete information and "learning" new information. The two operations are regulated by structures called *"gates"*, implemented as layers with sigmoid activation function (σ), producing a scalar value in range [0,1], followed by a pointwise multiplication operation ($\bigotimes$). The updated version of the cell state, C_t is filtered by the third gate, which decides what parts of it to output as h_t. In this

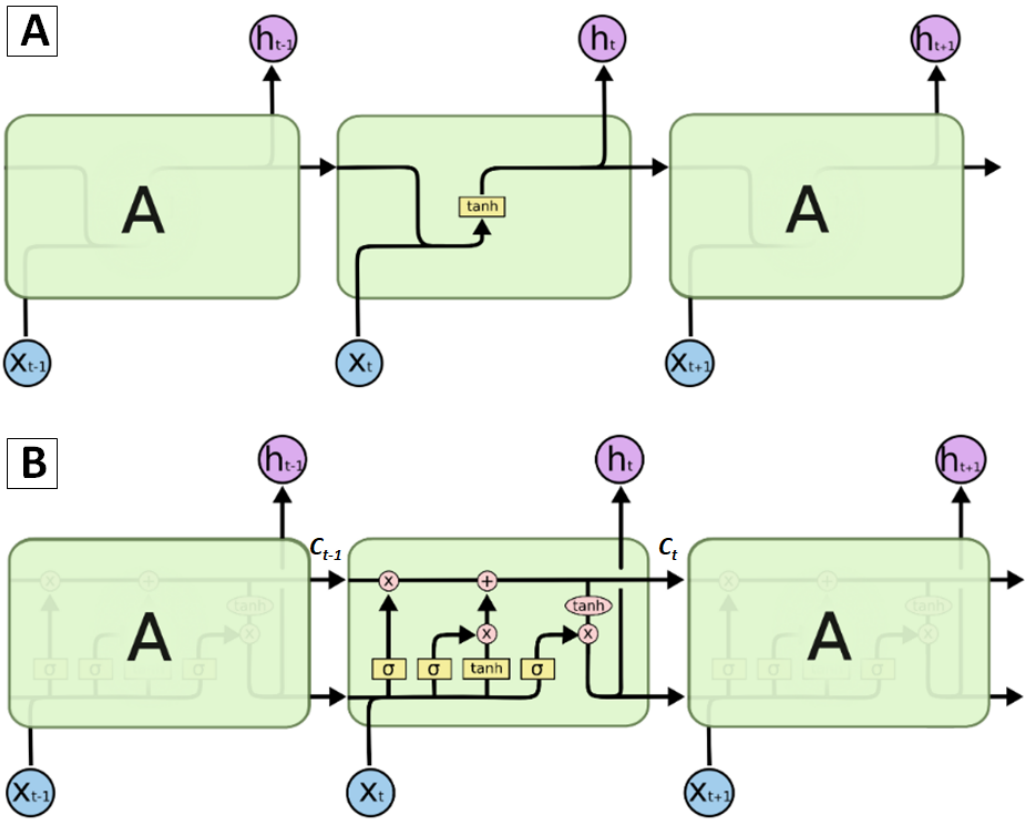


Figure 3.15: **(A)** Dynamic representation of RNN, highlighting the network structure: a single layer with *tanh* activation function taking as input the concatenation of the previous output h_{t-1} and the current input x_t. **(B)** Dynamic representation of LSTM network, highlighting the network structure: multiple layers interact with each other, in order to update the cell state term C_t, responsible to keep memory of previous events. Layers with σ activation function are responsible of "learning" and "forgetting" of the cell state term.

work, the LSTM network can capture the movements of the agent throughout different frames, forming a global view of the trajectory, necessary to ensure the respect of kinematic constraints. The standard strategy for modelling temporal dependencies, is passing as input the last n frames (n defined depending on the context, e.g. $n = 4$ in [51]): this introduces a trade-off between the extension of the modelled time-dependencies and the computational cost associated with it (bigger n allows to have a better modelling of time dependencies, but also increases the amount of data to be processed at each iteration). LSTM networks allow to model time dependencies among several frames, without requiring an increase of the input dimension.
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.
During training, 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 ($ADV(s_t, a, \theta'_{\mathbf{v}}, \theta'_{\mathbf{l}})$), which, in turn, are used to calculate value loss, L_v:

$$L_v = \sum (R_t - V(s_t, \theta'_{\mathbf{v}}))^2 \tag{3.9}$$

and policy loss, L_p:

$$L_p = -log(\pi(s_t, \theta'_{\mathbf{l}})) \cdot \ ADV(s_t, a, \theta'_{\mathbf{v}}, \theta'_{\mathbf{l}}) - \beta \cdot \mathbb{H}(\pi(s_t, \theta'_{\mathbf{l}})) \tag{3.10}$$

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 [54]. The worker then uses L_p and L_v to compute the gradients

$\Delta_{\theta'_l}$ and $\Delta_{\theta'_v}$:

$$\Delta_{\theta'_v} = \nabla_{\theta'_v}(L_v) \tag{3.11}$$

$$\Delta_{\theta'_l} = \nabla_{\theta'_l}(L_p) \tag{3.12}$$

and use them to update the global network parameters. In this work ADAM optimizer was used to perform gradient descent optimization.
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 2.

3.3.3 Reward function

The reward function is the part of the model in which the rules described in paragraph 3.3 are translated in mathematical constraints and implemented. The three main requirements that the algorithm is expected to fullfil are:

1. path length minimization
2. obstacle clearance maximization
3. compliance with needle kinematic constraints

The reward function $r(s_t, a_t)$ is defined as:

$$r(s_t, a_t) = \begin{cases} k_w \text{ target reached} \\ -k_1 \cdot D_{s_{t+1}-T} - k_2 \cdot exp(\,\alpha/\pi) - k_3 \cdot \frac{\beta}{\pi/2} - k_4 \end{cases} \tag{3.13}$$

A positive constant reward k_w is given upon reaching the target.
In case the target is not reached, a negative reward is given according to:

Algorithm 2 A3C - pseudocode for each actor-learner worker.

// Assume global network actor-critic weights $\theta_{\mathbf{l}}$ and $\theta_{\mathbf{v}}$

// Assume worker specific actor-critic weights $\theta'_{\mathbf{l}}$ and θ'_{v}

// Assume global counter T

Initialize thread step counter $t \leftarrow 1$

repeat

  Reset gradients: $d\theta_{\mathbf{l}} \leftarrow 0$ and $d\theta_{\mathbf{v}} \leftarrow 0$

  Synchronize thread parameters: $\theta'_{\mathbf{l}} = \theta_{\mathbf{l}}$ and $\theta'_{\mathbf{v}} = \theta_{\mathbf{v}}$

  $t_{start} = t$

  Get state s_t

  repeat

    Perform a_t according to policy $\pi(s_t; \theta'_{\mathbf{l}})$

    Receive reward r_t and new state s_{t+1}

    $t \leftarrow t + 1$

    $T \leftarrow T + 1$

  until terminal s_t **or** if $t - t_{start} == t_{max}$

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

  for $i \in \{t-1, \ldots, t_{start}\}$ **do**

    $R_t \leftarrow r_i + \gamma\, R_t$

    compute gradients $\Delta_{\theta'_{\mathbf{l}}}$ and $\Delta_{\theta'_{\mathbf{v}}}$

    accumulate gradients wrt $\theta_{\mathbf{l}}$: $d\theta_{\mathbf{l}} \leftarrow d\theta_{\mathbf{l}} + \Delta_{\theta'_{\mathbf{l}}}$

    accumulate gradients wrt $\theta_{\mathbf{v}}$: $d\theta_{\mathbf{v}} \leftarrow d\theta_{\mathbf{v}} + \Delta_{\theta'_{\mathbf{v}}}$

  update of $\theta_{\mathbf{l}}$ using $d\theta_{\mathbf{l}}$ and of $\theta_{\mathbf{v}}$ using $d\theta_{\mathbf{v}}$

until $T > T_{max}$

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

$$D_{s_{t+1}-T} = \sqrt{(x_{s_{t+1}} - x_g)^2 + (y_{s_{t+1}} - y_g)^2 + (z_{s_{t+1}} - z_g)^2} \tag{3.14}$$

- needle's 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 3.16. The reward associated with angle α makes the agent learns to move always toward the target, avoiding curved trajectories when straight ones are feasible. The reward associated with angle β is implemented to avoid abrupt changes in the direction. The two rewards, combined, allow the agent to learn how to move in a smooth, efficient way in the environment.

- obstacle clearance maximization, with a constant negative reward k_{obst} given if the agent is in a cell with minimum distance $d_m \leq 1$ pixel/voxel. This reward makes the agent learn to keep a minimum distance from obstacles. The negative reward can be extended to cells close to the ones adjacent to obstacles, in order to increase the minimum distance from obstacles that the agent should respect.

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, which allow to menage the trade-off in the optimization process. In this context, the values assigned to each constant were empirically determined and are shown in Table 3.1.

Table 3.1: Reward weights

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

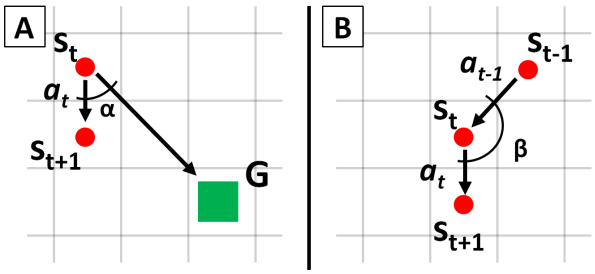


Figure 3.16: **(A)** α is the angle between the current action a_t, connecting the current state s_t with the new one s_{t+1} and the vector connecting s_t and the target G. **(B)** β is the angle between the current action a_t and the previous one a_{t-1}.

3.3.4 Training strategy

According to the constraint defined in paragraph 3.3, training could be performed for each hemisphere separately. This allows to sensibly shorten and ease the training process, since the workers have to learn ho to move in a reduced environment. At every episode a new starting point was randomly chosen among the available free cells, and a new exit point was chosen inside a predefined target area. The target area was determined by manually identifying a cell, and considering, as part of the target area, all the nearby cells contained in a circle/sphere with ray λ. Additionally, at every episode, each agent was assigned a different map among the ones available for training, in order to encourage its ability to learn the correct policy, independently from the specific environment, starting point or exit point, thus maximizing its generalization and abstraction capabilities.

Chapter 4

EXPERIMENTAL SETUP AND RESULTS

4.1 Experimental setup

4.1.1 Environment creation

The maps for training and testing phase were obtained from multi-modal MR images of 7 healthy subjects, 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 $\times$ 299; voxel size, 0.8 $\times$ 0.8 $\times$ 0.8 mm; thickness: 0.8/0 mm gap;) a 3D high-resolution time-of-flight MR angiography (TOF-MRA) (acquisition matrix: 500$\times$399; acquired voxel size, 0.4$\times$0.5$\times$0.9 mm; reconstructed voxel size: 0.3 $\times$ 0.3 $\times$ 0.45 260 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$\times$126; voxel size, 2$\times$2$\times$2 mm; thickness, 2/0 mm gap).

Both the ToF and the T1-weighted images were segmented by thresholding

using 3DSlicer, an open source software for biomedical imaging processing [59]. Blood vessels and brain models were extracted. 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 ([44], [45]). The three models : 1) brain cortex, 2) blood vessels and 3) corticospinal tracts were registered in 3DSlicer, by means of a roto-translation operator and the 3D binary label maps were obtained (dimension $256 \times 256 \times 256\ mm$). From the label maps, 3D maps and 2D maps were built, as described in paragraph 3.2. 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.

4.1.2 Training setup

Among the 7 2D and 3D maps, 6 2D maps and 6 3D maps were used during the training phase of the model; 1 2D map 1 3D map were used for the testing phase. Each map was split in two hemispheres (Figure 4.1), according to the training strategy defined in section 3.3.4. 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 exit point was chosen inside a predefined target area. The target area ray λ was set equal to 3 for 2D, and to 2 for 3D. At every episode, each agent was assigned a different map among the 6 available for training.
The same strategy was followed to train both the 2D and the 3D models.

4.1.3 Hardware and Software specifications

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. The GA3C algorithm was implemented in Tensorflow, an open source machine learning library for research and production [60]. Training performances were visualized by Tensorboard, a Tensorflow visualization tool.

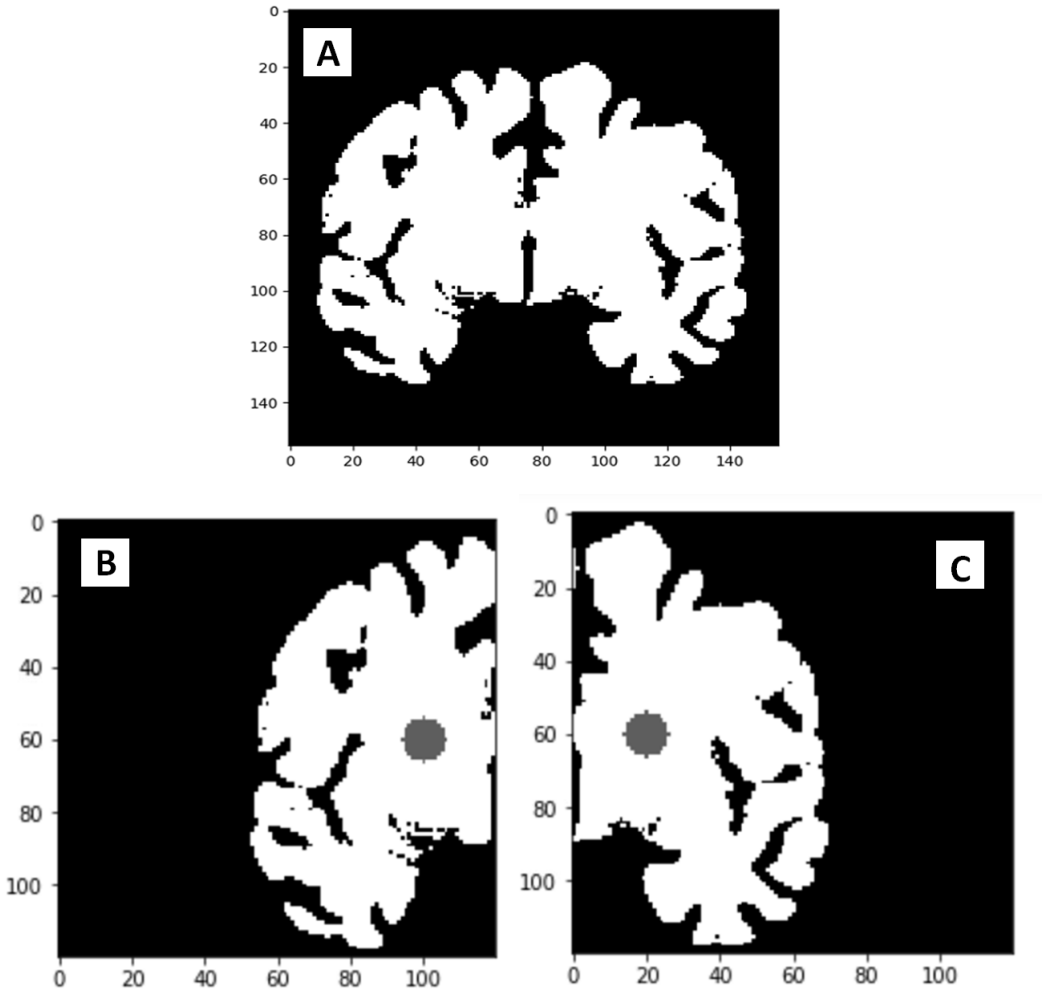


Figure 4.1: **(A)** Original 2D binary map. **(B)** and **(C)**: hemisphere maps obtained from **A** and used during training. The grey regions represent the target areas for each map, inside which, at each iteration, a new target is chosen.

4.1.4 Experimental protocol

The same experimental protocol was followed for both the 2D and the 3D models. Experiments were carried out on the predefined testing map, not used during training. 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*[29] and RRT*[33], for 2D and 3D, respectively.

The obtained trajectories were evaluated considering their length (l), the minimum (d_m) and average distance ($\bar{d}$) from obstacles, and the maximum curvature (k_M).

Additionally, a cost function F_c, accounting for the above mentioned parameters, was defined as:

$$F_c = \exp(\, w_1 \cdot \frac{l}{D_{S-T}} - w_2 \cdot \frac{d_m}{d_M} - w3 \cdot \frac{\bar{d}}{d_M} + w_4 \cdot k_M) \tag{4.1}$$

where D_{S-T} is the Euclidean distance between start cell and target cell. Coefficients w_1, w_2, w_3, w_4 are the parameters weights, and are shown in Table 4.1. The cost function, containing all the main requested features, was used during testing stage, to assess the quality of the obtained trajectories. As the number of samples for each map was small, non-parametric statistics was used [61]. To evaluate differences between each pair of methods, a pairwise comparison for each feature was run, through Mann-Whitney U test ($U < 127, p < 0.05$).

Table 4.1: Cost function weights

w_1	w_2	w_3	w_4
0.5	1.5	1.5	1.5

4.2 Results evaluation

4.2.1 Training performances

The training performances were monitored by considering the value, at increasing iterations number, of:

- policy loss L_p (defined in eq. 3.10)
- value loss L_v (defined in eq. 3.9)
- path length l
- reward r

Figures 4.2 and 4.3 show the training performances of 2D and 3D models, respectively. Each plot shows a smoothed version of the specific quantity for each worker. The spikes on the loss functions are due to Adam optimization with mini-batches, and are not significant when evaluating training performances. For both 2D and 3D models training, a stable value is reached with very few iterations with respect to standard RL algorithms, thanks to the parallel, asynchronous action of the workers.

4.2.2 Testing performances and comparison with state-of-the-art

GA3C vs A* on 2D environments

In order to assess the quality of performances on 2D environments, the proposed GA3C method was compared with the standard A* algorithm, following the experimental protocol defined in section 4.1.4. Figure 4.4 shows an example of a trajectory obtained through GA3C model and A*. Figure 4.5 summarizes the obtained results. Table 4.2 shows 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

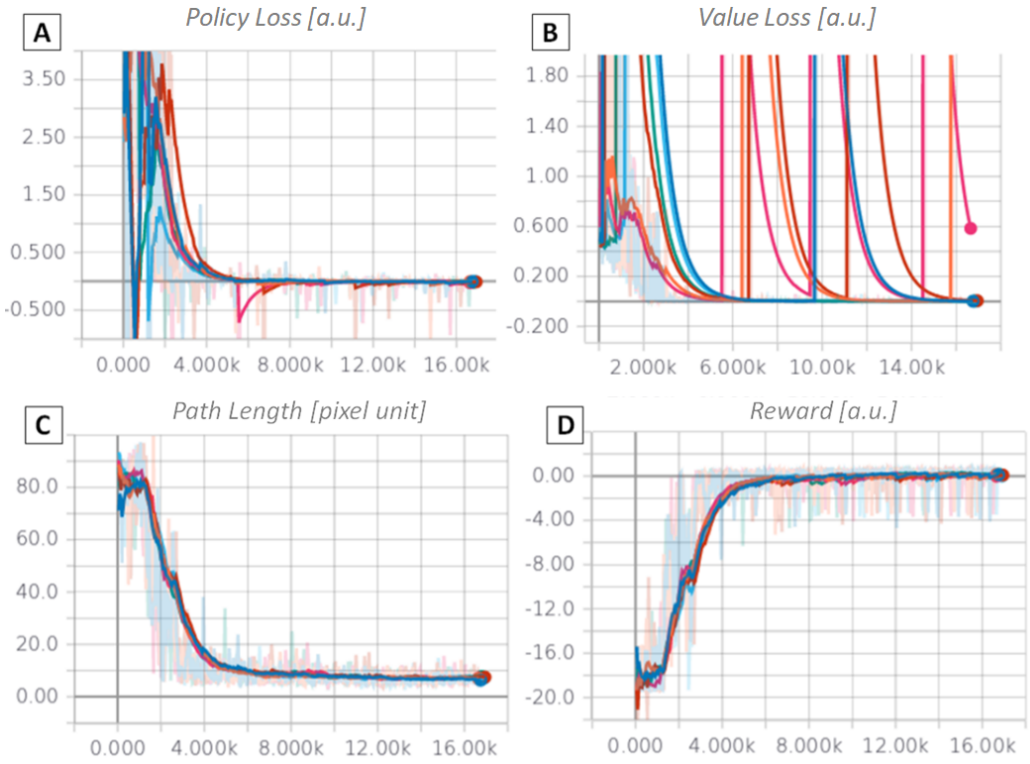


Figure 4.2: Training performance assessment on 2D model. **(A)** Policy loss, expressed in arbitrary unit of measurement. **(B)** Value loss, expressed in arbitrary unit of measurement. **(C)** Path length, expressed considering, as unit of measurement, the size of a pixel. **(D)** Reward, expressed in arbitrary unit of measurement. Every quantity is observed at increasing iterations number.

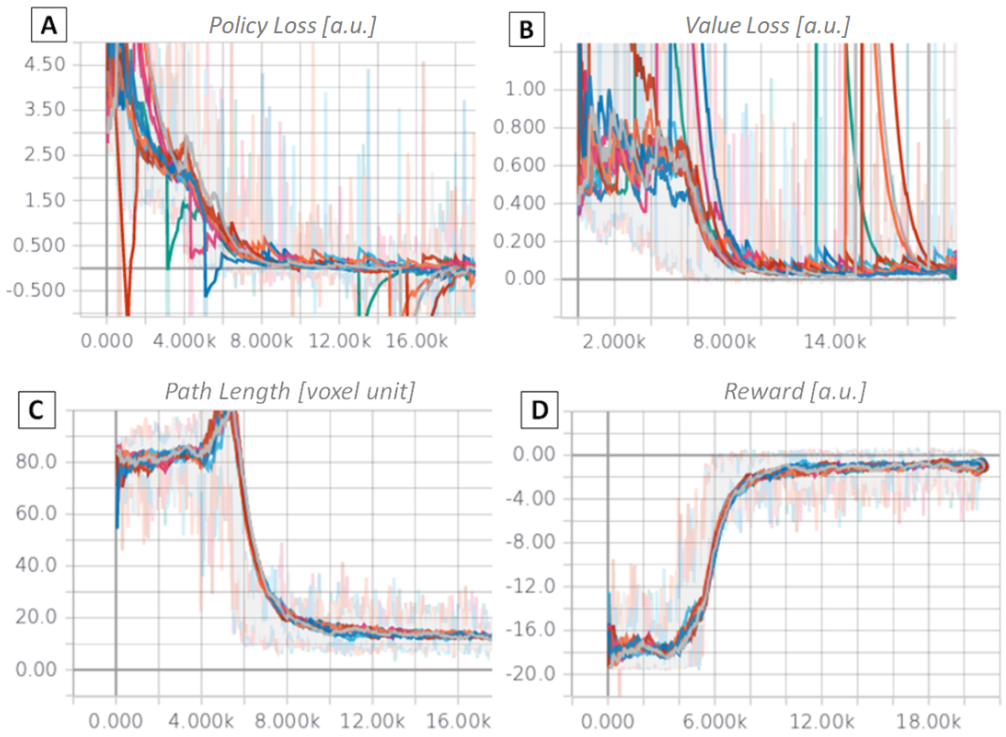


Figure 4.3: Training performance assessment on 3D model. **(A)** Policy loss, expressed in arbitrary unit of measurement. **(B)** Value loss, expressed in arbitrary unit of measurement. **(C)** Path length, expressed considering, as unit of measurement, the size of a voxel. **(D)** Reward, expressed in arbitrary unit of measurement. Every quantity is observed at increasing iterations number.

and smoothness indexes are reported with the 25% and 75% quantiles, and median value.

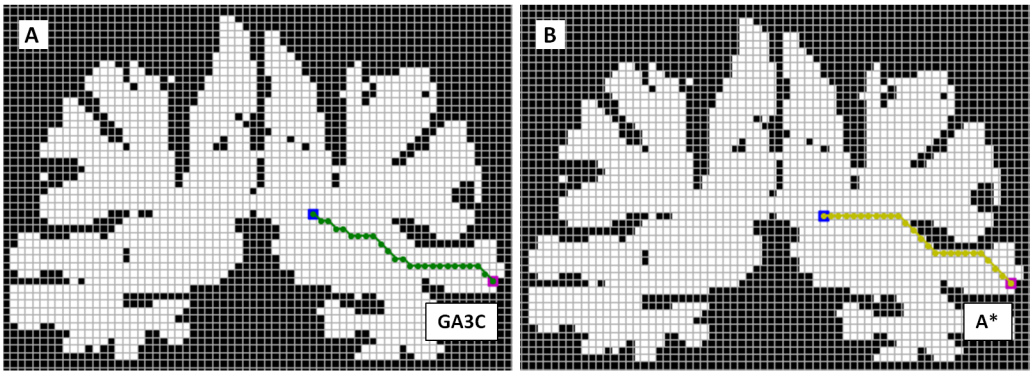


Figure 4.4: Path connecting the same starting cell and exit cell obtained through GA3C model **(A)** and A* algorithm **(B)** on a 2D map. Black cells correspond to obstacles, white cells correspond to free space.

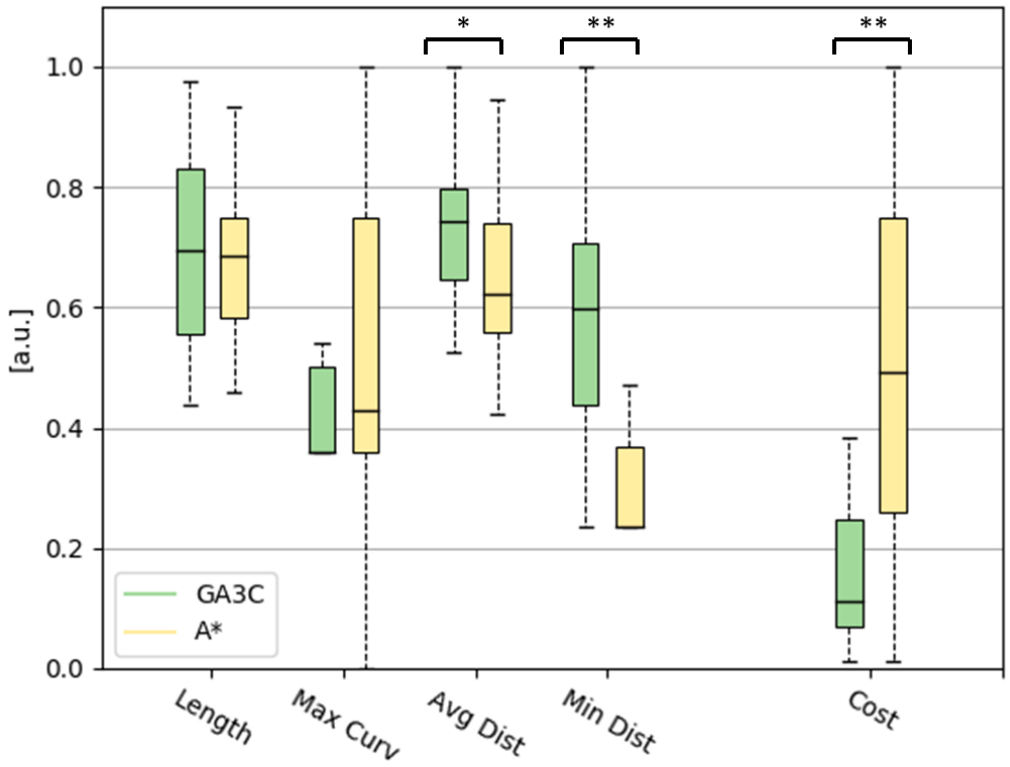


Figure 4.5: The presented solution (GA3C) is tested against A* algorithm one 2D test map, not used for training, on 20 computed trajectories.The box-plots resume the obtained results in terms of path length, maximum curvature, average and minimum distance from obstacles, and cost. Each value in the box-plots is normalized in the range [0,1]. Statistical significance of Mann-Whitney U test is also reported for each feature (* $p < 0.05$, ** $p < 0.01$).

Table 4.2: 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

GA3C vs RRT* on 3D environments

In order to assess the quality of performances on 3D environments, the proposed GA3C method was compared with the standard RRT* algorithm, following the experimental protocol defined in section 4.1.4. Figure 4.6 shows an example of a trajectory obtained through GA3C model and RRT*. Figure 4.7 summarizes the obtained results. Table 4.3 shows 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.

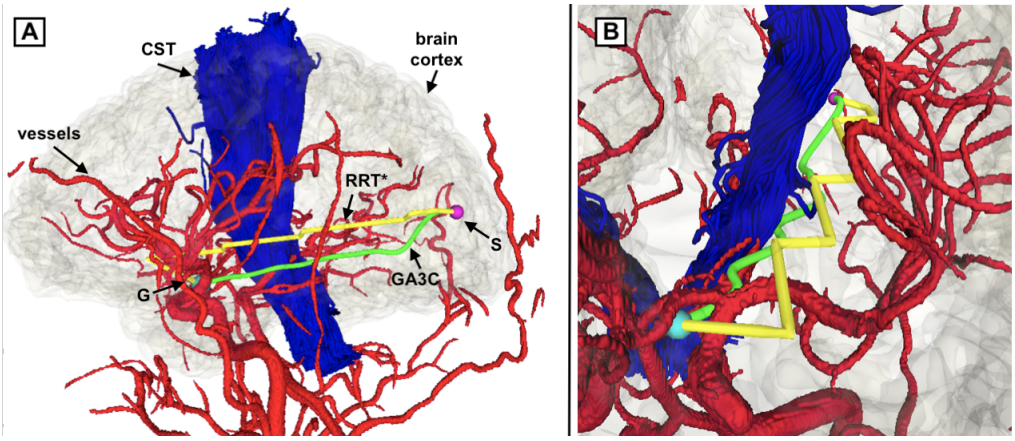


Figure 4.6: Path connecting a starting cell (S) and an exit cell (G) obtained through GA3C model in green and RRT* algorithm in yellow, on a Slicer model of a 3D map. Corticospinal tracts (CST) and vessels are shown in blue and red, respectively. **(A)** shows a sagittal view of the whole brain; **(B)** shows a close-up, highlighting the higher smoothness of the GA3C generated trajectory, with respect to RRT*.

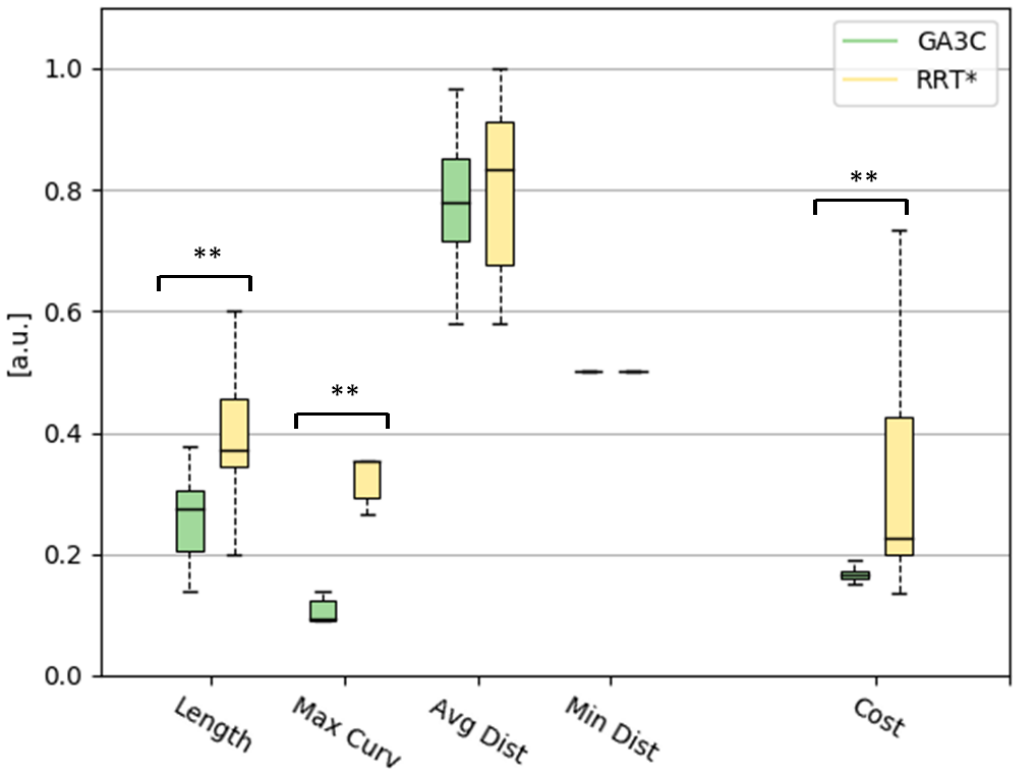


Figure 4.7: The presented solution (GA3C) is tested against RRT* algorithm on the 3D test map, on 20 computed trajectories.The box-plots resume the obtained results in terms of path length, maximum curvature, average and minimum distance from obstacles, and cost. Each value in the box-plots is normalized on range [0,1]. Statistical significance of Mann-Whitney U test is also reported for each feature (* $p < 0.05$, ** $p < 0.01$)).

Table 4.3: 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	$0.087s$	$0.109s$	$0.118s$	0.090	0.095	0.125
*RRT**	$0.05s$	$0.053s$	$0.069s$	0.286	0.352	0.352

Chapter 5

DISCUSSION

In this work 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, according to the cost function defined in eq. 4.1, as shown in Figures 4.5 and 4.7, rightmost column. Breaking down the cost function, Figures 4.5 and 4.7 show the behaviour of the proposed method with respect to the critical parameters on the test map. When applied to 2D and compared to A*, the proposed method was able to generate trajectories with an higher average and 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 4.2. Finally, the length was in general comparable (average difference: 0.73, standard deviation: 2.32). Figure 4.4 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*, the proposed method was able to generate trajectories with significantly lower length and significantly higher smoothness (Table 4.3). Regarding the minimum distance value, we observed that the paths are always at a feasible secure distance from the ob-

stacles. However, the 3D environment, at the given resolution, introduces far more obstacles than a 2d environment and forces the path minimum distance to be lower. Figure 4.6 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*.
Training results, shown in Figures 4.2 and 4.3, show a fast convergence of path length and reward values toward asymptotic values, with very small improvement after that. The 3D model takes slightly longer to converge, and the asymptotic reward value reached is lower than the one of the 2D case. This is due to the higher complexity of the 3D problem, mostly related to the much higher number of actions (26 in 3D against 8 in 2D), that substantially increase the challenges in learning the optimal policy π^*.
The computational times, higher with respect to A* and RRT* (Tables 4.2 and 4.3), 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 [37] [45].
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 presented new maps, the models occasionally fail to generate trajectories when dealing with new maps with severe differences from the ones it was trained on. This is more evident in 3D environments, for the above mentioned reasons.

Chapter 6

CONCLUSION AND FUTURE WORK

The present work proposes a novel automatic planner for steerable needles in the context of keyhole-neurosurgery. Given multi-modal MR images of the patient, a surgeon-defined starting point, and a target, the proposed path planner 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 DRL approach, and uses a GA3C algorithm [54] [55] to solve the path planning problem. The model is trained exploiting a unique approach involving multiple workers running in parallel, on different CPU threads, getting independent experience, and periodically updating a global network. This approach guarantees a more stable learning and lower computational times, with respect to standard DRL approaches. The optimization of the trajectories according to the three main requested features, is obtained by properly shaping the reward function and tuning the weight coefficients present in it, thus determining the trade-off between different features optimization.
When tested against the standard A* and RRT* algorithms, the proposed method performed better in terms of cost function, generating smoother trajectories with a sensibly higher clearance from safety regions, and with

comparable 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. When dealing with maps with severe differences from the ones used in training phase, the algorithm occasionally fails to provide trajectories, mainly in 3D environments. This limit could be overcome by increasing the number of maps used for training, and eventually running more workers in parallel to speed-up the training phase. The clearance from safety regions could be improved by introducing rewards proportional to the values of a distance map, making the reward function associated with it continuous, and the optimization process more accurate. The kinematic feasibility of the trajectories could be also improved by introducing in the reward function an inverse kinematic model of the catheter [62], accurately defining the region of space of admissible trajectories.

The most stimulating aspects of the proposed learning-based approach are its flexibility and the possibility to extend it to dynamic environments. The flexibility is given by the reward function, which allows to easily implement all the requested features in it: more constraints could be added, without increase of the computational complexity of the model. The possibility to extend the model to dynamic environments is probably the most stimulating possible future development. The path planner could be trained to be able to respond to environment changes, as the ones that could take place during an insertion (due to needle-tissues interaction), making it suitable to work not only pre-operatively, but also intra-operatively, in combination with recently developed MR intraoperative systems [63].

Bibliography

[1] K Fitch, T Engel, and A Bochner. «Cost Differences Between Open and Minimally Invasive Surgery.» In: *Managed care (Langhorne, Pa.)* 24.9 (2015), pp. 40–48.

[2] K Bhattacharya. «Kurt Semm: a laparoscopic crusader». In: *Journal of minimal access surgery* 3.1 (2007), p. 35.

[3] Terry M Peters. «Image-guidance for surgical procedures». In: *Physics in Medicine & Biology* 51.14 (2006), R505.

[4] Mack MJ. «Minimally Invasive and Robotic Surgery». In: *JAMA* 285.5 (2001), pp. 568–572.

[5] Chris S. Karas and E. Antonio Chiocca. «Neurosurgical robotics: a review of brain and spine applications». In: *Journal of Robotic Surgery* 1.1 (2007), pp. 39–43.

[6] Robert Reisch Et al. «The Keyhole Concept in Neurosurgery». In: *World Neurosurgery* 79.2 (2013), S17.e9–S17.e13.

[7] N. Abolhassani, R. Patel and M. Moallem. «Needle insertion into soft tissue: A survey». In: *Medical engineering and physics* 29.4 (2007), pp. 413–431.

[8] E. De Momi Et al. «Multi-trajectories automatic planner for StereoElectroEncephaloGraphy (SEEG)». In: *Medical engineering and physics* 9.6 (2014), pp. 1087–10971.

[9] McKinsey L Goodenberger and Robert B Jenkins. «Genetics of adult glioma». In: *Cancer genetics* 205.12 (2012), pp. 613–621.

[10] L Nam et al. «Drug delivery nanosystems for the localized treatment of glioblastoma multiforme». In: *Materials* 11.5 (2018), p. 779.

[11] Yuan-Yuan Xu et al. «Development of targeted therapies in treatment of glioblastoma». In: *Cancer biology & medicine* 12.3 (2015), p. 223.

[12] Marc Fakhoury. «Drug delivery approaches for the treatment of glioblastoma multiforme». In: *Artificial cells, nanomedicine, and biotechnology* 44.6 (2016), pp. 1365–1373.

[13] Max J Cotler et al. «Steerable Microinvasive Probes for Localized Drug Delivery to Deep Tissue». In: *Small* (2019), p. 1901459.

[14] P. E. Dupont, J. Lock, B. Itkowitz and E. Butler. «Design and control of concentric-tube robots». In: *IEEE Transactions on Robotics* 26.2 (2010), pp. 209–225.

[15] Kyle B. Reed Et al. «Robot-Assisted Needle Steering». In: *IEEE Robot Autom* 18.4 (2011), pp. 35–46.

[16] P. Qi, H. Liu, L. Seneviratne and K. Althoefer. «Towards kinematic modeling of a multidof tendon driven robotic catheter». In: *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE* (2014), pp. 3009–3012.

[17] Zhenglong Sun Et al. «Modeling and motion compensation of a bidirectional tendon-sheath actuated system for robotic endoscopic surgery». In: *Computer Methods and Programs in Biomedicine* 119.2 (2015), pp. 77–87.

[18] A. Leibinger, M. J. Oldfield and F. R. y Baena. «Minimally disruptive needle insertion: a biologically inspired solution». In: *Interface focus* 6.3 (2016).

[19] R. Secoli and F. Rodriguez y Baena. «Experimental validation of curvature tracking with a programmable bevel-tip steerable needle». In: *International Symposium on Medical Robotics* (2018).

[20] M. Oldfield Et al. «Method to Reduce Target Motion Through Needle-Tissue Interactions». In: *Ann Biomed Eng* 43.11 (2015), pp. 2794–803.

[21] C. Burrows Et al. «Multi-target Planar Needle Steering with a Bio-inspired Needle Design.» In: *Advances in Italian Mechanism Science* (2016), pp. 51–60.

[22] R. Secoli and F. Rodriguez y Baena. «Closed-loop 3D Motion Modeling and Control of a Steerable Needle for Soft Tissue Surgery». In: *2013 IEEE International Conference on Robotics and Automation (ICRA)* (2013).

[23] Eden2020. URL: `http://www.eden2020.eu/about/objectives-approach-and-impact`.

[24] Alessandro Gasparetto et al. «Path planning and trajectory planning algorithms: A general overview». In: *Motion and operation planning of robotic systems*. Springer, 2015, pp. 3–27.

[25] Qidan Zhu, Yongjie Yan, and Zhuoyi Xing. «Robot path planning based on artificial potential field approach with simulated annealing». In: *Sixth International Conference on Intelligent Systems Design and Applications*. Vol. 2. IEEE. 2006, pp. 622–627.

[26] Pan Li et al. «A combination method of artificial potential field and improved conjugate gradient for trajectory planning for needle insertion into soft tissue». In: *J Med Biol Eng* 34.6 (2014), pp. 568–573.

[27] Min Gyu Park and Min Cheol Lee. «A new technique to escape local minimum in artificial potential field based path planning». In: *KSME international journal* 17.12 (2003), pp. 1876–1885.

[28] Edsger W Dijkstra. «A note on two problems in connexion with graphs». In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[29] Peter E Hart, Nils J Nilsson, and Bertram Raphael. «A formal basis for the heuristic determination of minimum cost paths». In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[30] Richard E Bellman. *Adaptive control processes: a guided tour*. Vol. 2045. Princeton university press, 2015.

[31] Steven M LaValle. «Rapidly-exploring random trees: A new tool for path planning». In: (1998).

[32] James J Kuffner and Steven M LaValle. «RRT-connect: An efficient approach to single-query path planning». In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[33] Sertac Karaman and Emilio Frazzoli. «Incremental sampling-based algorithms for optimal motion planning». In: *arXiv preprint arXiv:1005.0416* (2010).

[34] Vincent Duindam et al. «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.

[35] Sachin Patil and Ron Alterovitz. «Interactive motion planning for steerable needles in 3D environments with obstacles». In: *2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*. IEEE. 2010, pp. 893–899.

[36] Chiara Caborni et al. «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.

[37] Alberto Favaro et al. «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.

[38] Volodymyr Mnih et al. «Human-level control through deep reinforcement learning». In: *Nature* 518.7540 (2015), p. 529.

[39] Timothy P Lillicrap et al. «Continuous control with deep reinforcement learning». In: *arXiv preprint arXiv:1509.02971* (2015).

[40] Piotr Mirowski et al. «Learning to navigate in complex environments». In: *arXiv preprint arXiv:1611.03673* (2016).

[41] Piotr Mirowski et al. «Learning to navigate in cities without a map». In: *Advances in Neural Information Processing Systems*. 2018, pp. 2419–2430.

[42] Lei Tai, Giuseppe Paolo, and Ming 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.

[43] Aleksandr I Panov, Konstantin S Yakovlev, and Roman Suvorov. «Grid path planning with deep reinforcement learning: Preliminary results». In: *Procedia computer science* 123 (2018), pp. 347–353.

[44] Eleftherios Garyfallidis et al. «Dipy, a library for the analysis of diffusion MRI data». In: *Frontiers in neuroinformatics* 8 (2014), p. 8.

[45] Alice Segato et al. «Automated steerable path planning for Deep Brain Stimulation safeguarding fiber tracts and deep grey matter nuclei». In: (2019).

[46] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[47] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2012.

[48] Ling Pan et al. «Reinforcement Learning with Dynamic Boltzmann Softmax Updates». In: *arXiv preprint arXiv:1903.05926* (2019).

[49] Charles Blundell et al. «Weight uncertainty in neural networks». In: *arXiv preprint arXiv:1505.05424* (2015).

[50] Yarin Gal and Zoubin Ghahramani. *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*. 2016.

[51] Volodymyr Mnih et al. «Playing atari with deep reinforcement learning». In: *arXiv preprint arXiv:1312.5602* (2013).

[52] Diederik P Kingma and Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014).

[53] Ronald J Williams. «Simple statistical gradient-following algorithms for connectionist reinforcement learning». In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[54] Volodymyr Mnih et al. «Asynchronous methods for deep reinforcement learning». In: *International conference on machine learning*. 2016, pp. 1928–1937.

[55] Mohammad Babaeizadeh et al. «GA3C: GPU-based A3C for deep reinforcement learning». In: *CoRR abs/1611.06256* (2016).

[56] Bing Xu et al. «Empirical evaluation of rectified activations in convolutional network». In: *arXiv preprint arXiv:1505.00853* (2015).

[57] Sepp Hochreiter and Jürgen Schmidhuber. «Long short-term memory». In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[58] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. «Learning long-term dependencies with gradient descent is difficult». In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[59] Steve Pieper, Michael Halle, and Ron Kikinis. «3D Slicer». In: *2004 2nd IEEE international symposium on biomedical imaging: nano to macro (IEEE Cat No. 04EX821)*. IEEE. 2004, pp. 632–635.

[60] Mart Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `http://tensorflow.org/`.

[61] L VAJANI. *PROBABILITY AND STATISTICAL-INFERENCE-HOGG, RV, TANIS, EA*. 1978.

[62] V. Duindam Et al. «Three- dimensional motion planning algorithms for steerable needles using inverse kinematics». In: *The International Journal of Robotics Research,* 29.7 (2010), pp. 789–800.

[63] Christian Senft et al. «Intraoperative MRI guidance and extent of resection in glioma surgery: a randomised, controlled trial». In: *The lancet oncology* 12.11 (2011), pp. 997–1003.

Ringraziamenti

Ed eccoci ai ringraziamenti, la parte più impegnativa da scrivere per una persona di poche parole come me, ma allo stesso tempo la più bella.

Ringrazio la Professoressa Elena De Momi per la fiducia e le enormi opportunità che mi ha offerto durante questo percorso. Grazie per le sfide che mi ha permesso di affrontare, perché mi hanno fatto scoprire tanto di me che non conoscevo.

Un ringraziamento sentito alla mia correlatrice Alice, per il bel viaggio che abbiamo condiviso. Grazie per avermi insegnato qualcosa di nuovo ogni giorno.

Grazie a J e Pizzi. Nonostante le diverse strade che abbiamo preso abbiano provato ad allontanarci, io mi sento ancora come quando eravamo lì, tutti e tre insieme, dalla Marisa. Siete la cosa più bella che ho vissuto in questi anni al Poli.

Grazie a tutti i miei amici, siete troppi per nominarvi tutti. Ricordo ancora il giorno prima di partire per Milano: sembrava fosse la fine di tutto, e invece è stato soltanto un nuovo meraviglioso inizio. Niente ci ha mai separato e niente lo farà mai.

Grazie anche a chi mi ha fatto fare le cinque tutte le sere quando il giorno dopo dovevo scrivere la tesi, perché la mattina mi alzavo stanco, ma col sorriso.

Grazie Sofia, perché io e te siamo cresciuti insieme. L'album che mi regalasti cinque anni fa tra qualche lacrima, è ancora lì a farmi compagnia.

Grazie ai miei coinquilini, a quelli che sono passati per poco e se ne sono andati, e a quelli che ci sono sempre stati. Grazie Ghezzo, Vinz, Leo e Samu, siete stati una meravigliosa famiglia milanese.

Grazie nonne Nada e Popa, e nonni Gigi e Gino, grazie a chi è qui a guardarmi e a chi lo fa da un po' più lontano. Non mi sono mai sentito solo grazie a voi.

Grazie Mamma e Babbo, mi conoscete da una vita e sapete che il grazie che dedico a voi è quello più pieno e sentito, senza bisogno di aggiungere altro.