



A hybrid inductive learning-based and deductive reasoning-based 3-D path planning method in complex environments

Alice Segato¹ · Francesco Calimeri² · Irene Testa¹ · Valentina Corbetta¹ · Marco Riva^{3,4} · Elena De Momi¹

Received: 22 December 2020 / Accepted: 18 April 2022
© The Author(s) 2022

Abstract

Traditional path planning methods, such as sampling-based and iterative approaches, allow for optimal path's computation in complex environments. Nonetheless, environment exploration is subject to rules which can be obtained by domain experts and could be used for improving the search. The present work aims at integrating inductive techniques that generate path candidates with deductive techniques that choose the preferred ones. In particular, an inductive learning model is trained with expert demonstrations and with rules translated into a reward function, while logic programming is used to choose the starting point according to some domain expert's suggestions. We discuss, as use case, 3-D path planning for neurosurgical steerable needles. Results show that the proposed method computes optimal paths in terms of obstacle clearance and kinematic constraints compliance, and is able to outperform state-of-the-art approaches in terms of safety distance-from-obstacles respect, smoothness, and computational time.

Keywords Path planning · Artificial intelligence · Machine learning · Surgical robotics · Steerable needles · Answer set programming

1 Introduction

Moving agents, such as mobile robots, are increasingly being employed in many complex environments. Starting from the initial applications of mobile robots to manufacturing industries, a variety of robotic systems have been developed, and they have shown their effectiveness in performing different kinds of tasks, including smart home environments (Khawaldeh et al., 2016), airports (Zhou et al., 2016),

shopping malls (Kanda et al., 2009), and manufacturing laboratories (Chen et al., 2014).

Nowadays, path planning is fruitfully employed in many fields, such as entertainment, medicine, mining, rescuing, education, military, space, agriculture, robots for elderly persons, automated guided vehicles, for transferring goods in a factory, unmanned bomb disposal robots and planet exploration robots (Robert et al., 2008). Apart from robotic applications, path planning finds use in planning the routes on circuit boards, obtaining the hierarchical routes for networks in wireless mobile communication, planning the path for digital artists in computer graphics, reconnaissance robots and in computational biology to understand probable protein folding paths (Raja & Pugazhenthii, 2012).

In recent years, path planning has been also widely used in surgery (Adhami & Coste-Manière, 2003). In current clinical practice, a growing number of minimally invasive procedures rely on the use of needles, such as biopsies, brachithery for radioactive seeds placement, abscess drainage and drug infusion (Shi et al., 2016). With respect to standard open surgeries, the small diameter of the needle allows to access the targeted anatomy inflicting limited tissue damage and thus reducing the risks for the patient and speed up the recovery.

This project has received funding from the European Union's EU Research and Innovation programme Horizon 2020 under Grant Agreement No. 688279.

✉ Alice Segato
alice.segato@polimi.it

- ¹ NearLab, Department of Electronics, Information and Bioengineering Department (DEIB), Politecnico di Milano, 20133 Milan, Italy
- ² Department of Mathematics and Computer Science, University of Calabria, 87036 Rende, Italy
- ³ Department of Medical Biotechnology and Translational Medicine, Università degli Studi di Milano, Milan, Italy
- ⁴ IRCCS Humanitas Research Hospital, Rozzano, Milan, Italy

Over the last two decades, different research groups have focused their efforts on the development of needles able to autonomously steer inside the tissue. These needles can perform curvilinear trajectories planned to maximize the distance from sensitive anatomical structures to be avoided and reach targets otherwise inaccessible via rectilinear insertion paths (Wang et al., 2011). Accurate placement of the needle tip inside tissue is challenging, especially when the target moves and anatomical obstacles must be avoided. Moreover, the complex kinematics of steerable needles (Favaro et al., 2020) make the path planning challenging requiring the aid of automatic or semi-automatic path planning solutions. In our opinion, one of the most challenging path planning scenarios in surgery is neurosurgery. The complexity of planning within the brain is closely linked to the crowded anatomical structures and the relative delicacy of tissues that cannot be damaged without the risk of causing permanent damage or even death. In addition, due to the consistency of brain matter, there are few catheters capable of tracing curvilinear trajectories and those that exist have a minimal degree of curvature (Audette et al., 2020). These reasons significantly reduce the number of feasible trajectories that guarantee a high level of safety. According to this, the planning space in the brain is considered a complex environment. Furthermore, in the majority of cases, reaching the target, located deep in the brain, is very difficult due to the kinematic limitations of the catheter and the various anatomical regions to be avoided. Therefore, during the planning phase, several poses on the surface of the skull have to be evaluated to guarantee the best entry pose for reaching the target.

In general, moving agents in a static or dynamic known environment means finding one or more admissible paths from a starting configuration to a target configuration, avoiding obstacles and some movement possibilities, identified as kinematic constraints. The path planning problem is part of a larger class of “scheduling” and “routing problems”, and it is known to be non-deterministic polynomial-time hard and complete (Obitko, 1998). In general, path planning algorithms performance can be evaluated over two main characteristics: “completeness” and “optimality”. An algorithm is said to be complete if it can find a solution in a definite interval of time, provided that the solution exists, or report a failure otherwise. An algorithm is optimal if no other algorithm uses less time or space complexity. Given a path, path length is defined as the total distance covered by the moving agent from the starting position to the target, path safety represents the distance from the path to the nearest obstacle, and computation time is the time required to compute a path.

In this work, we propose a framework that couples inductive and deductive techniques in order to improve path planning performances. In particular, an inductive learning model, relying on demonstrations performed by expert operators, is in charge of generating a set of paths as can-

didate solutions; a deductive reasoning module selects then the “best” starting point, according to explicit knowledge modeled over domain experts suggestions. Interestingly, this kind of coupling allows us to transfer to the automated path planner some of the knowledge available at human level: the inductive learning module “catches” via demonstrations expert capabilities that are hard to explicitly express (e.g., visual-spatial, bodily-kinesthetic), while the deductive module formally encodes what has been elaborated by the experts upon long-lasting practice (e.g., domain knowledge, best practices). Furthermore, the deductive technique based on a declarative formalism grants several advantages: on the one hand, it makes the knowledge easy to maintain and update; on the other hand, it allows us to provide the final user with a highly customizable tool with real time visual feedback, as she can decide what is important for choosing the best path, and to what extent, for each case at hand.

Eventually, we assess the viability of the proposed approach in a use case, namely 3-D path planning for neurosurgical steerable needle, proving that it stands or even outperform state-of-the art solutions.

The remainder of the paper is structured as follows: in Sect. 2 an overview of the path planning approaches proposed in literature is presented. Section 3 outlines our path planning method. Section 4 describes the experimental protocol used. Section 5 presents the comparison between the presented solution, the expert manual path definition and a state-of-the-art path planning method for the specific neurosurgical application. Discussion and Conclusions are reported in Sect. 6.

2 Related works

Several approaches for path planning have been proposed in literature: graph-based, sampling-based, optimisation-based, heuristic-based, learning-based, reasoning-based methods. These methods are described below, and summarised in Table 1, according to optimality (i.e. an algorithm is known to be and “optimal” since it can estimate the best path, according to a certain criteria, given the specific resolution of the approximation); completeness (i.e. an algorithms is known to be “complete”, as it can determine whether a solution exists in finite time); scalability (i.e. an algorithms is known to be “scalable” as it can plan a path in a reasonable time even if the search space increase in size); computational time (i.e. the execution time to obtain a solution); the ability to plan within a complex environment (i.e. the environment is composed of many obstacles with elaborate shapes, narrow passages and tangled locations); and the ability to obtain a smooth path (i.e. able to minimise the along the curvature).

Dijkstra algorithm (Dijkstra, 1959) and A* (Hart et al., 1968) are graph-based methods based on the discrete

Table 1 Related work (Color table online)

Method	Optimal	Complete	Scalable	Computation time	Complex environment	Smooth path
Graph-based	✓	✓	×	↑	×	×
Sampling-based	×	×	✓	↓	✓	×
Optimisation-based	✓	✓	×	↑	✓	✓
Heuristic-based	×	×	✓	↑	×	×
Learning-based	×	×	✓	↓	✓	✓
Reasoning-based	✓	✓	×	↓	×	×
ILDR	✓	✓	✓	↓	✓	✓

✓=yes, ×=no, ↑=high, ↓=low

approximation of the planning problem. Many methods represent the environment as a square graph, or as an irregular graph (Kallem et al., 2011), or a Voronoi diagram (Garrido et al., 2006). A search is performed in order to find an optimal path. These algorithms are known to be “resolution-complete”, as the one proposed in Fu et al. (2021) that provides more guarantees also in terms of efficiency, and “resolution-optimal”. This approach may also be used for identifying a restricted area where further optimisation refinements can be performed (Huang et al., 2009). Notably, even though graph-based methods are relatively simple to implement, they require considerable computational time as the environment size increases (Bellman, 1966) or becomes complex. Tangent graph-based planning methods for a given environment build a graph by selecting collision-free common tangents between the obstacles. These methods allow more accurate path planning around curved obstacles without errors caused by polygonal approximation; however, these methods are not always suitable when considering the kinematics limitations of a moving agent and require additional optimisation steps (Tovar et al., 2007) to obtain a smooth path.

Sampling-based methods are based on a random sampling of the working space, with the aim of significantly reducing the computational time. Rapidly-exploring Random Tree (RRT) (LaValle & Kuffner Jr., 2000) is an exploration algorithm for quick search in high-dimensional spaces, more efficient than brute-force exploration of the state space. In fact, this class of methods is scalable and capable of planning in a complex environment. Its enhanced versions, RRT* (Jordan & Perez, 2013; Favaro et al., 2018) and bidirectional-RRT (Karaman & Frazzoli, 2011) are “probabilistically complete” since the probability to find an existing solution tends to one, as the number of samples goes to infinity, and “asymptotically optimal”, as they can refine an initial raw path by increasing the sampling density of the volume.

Paths computed with the approaches mentioned above can be further refined using Bezier curves (Hoy et al., 2015), splines (Lau et al., 2009), polynomial basis functions (Qu et al., 2004), or with optimisation-based methods such as evolutionary algorithms, simulated annealing, and particle

swarm (Besada-Portas et al., 2010) to obtain a smooth path. These approaches have the advantage of working properly in complex environments, as demonstrated in Favaro et al. (2021); however, they require higher computational time than the sampling-based methods.

Artificial Intelligence (AI) has been increasingly used while dealing with path planning tasks (Erds et al., 2013) in the last decade. Heuristic-based techniques such as greedy (Sniedovich, 2006) and genetic algorithm (Whitley, 1994) are two examples of AI approaches; they belong both to the class of optimisation procedure. Greedy algorithms for path planning are often used in combination with other approaches. This kind of algorithms fails to find the optimal solution, as it takes decisions merely on the basis of the information available at each iteration step, without considering the overall picture. Genetic algorithms are also used to generate solutions for path optimisation problems based on operations like mutation, crossover and selection. For this kind of algorithms, the most relevant limit is computational time, as it significantly increases with the search space. Completeness depends on the heuristic function.

Learning-based methods are more flexible than graph-based and sampling-based methods (Xu et al., 2008); indeed, they allow one to directly include all expected constraints and optimality criteria (obstacle clearance, kinematic constraints, minimum path length) in the optimisation process, without the need for subsequent refinement steps, which are time-consuming and may still not lead to the optimal path (Segato et al., 2020). Mnih et al. (2015) and Lillicrap et al. (2015) showed that Deep Reinforcement Learning (DRL) is suitable for solving path planning problems, and several studies (Mirowski et al., 2016, 2018; Tai et al., 2017) applied DRL in path planning. Panov et al. (2018) made use of the DRL approach to a grid path planning problem with promising results on small environments. Inverse Reinforcement Learning, also known as an Inductive Learning (IL) technique (Michalski, 1983) that is a process where the learner discovers rules by observing examples, has been applied to a wide range of domains, including autonomous driving (Wulfmeier et al., 2016), robotic manipulation (Crooks et al., 2016) and grid-world planning (Nguyen et al., 2017). In

general learning-based method are not optimal or complete, the computational time is not high, and it is not increasing when the search space increase. They perform well in complex environment even if it is dynamic because they don't need prior information about obstacles, as we demonstrated in our previous work (Segato et al., 2021a).

Reasoning-based approaches for path planning have been successfully designed, providing high-level methods like in Lifschitz (2002). They are optimal and complete. Portillo et al. (2011) successfully solved the path planning problem, Gómez et al. (2021) and Erdem et al. (2015) encoded multi-agents pathfindings and Erdem et al. (2012) used a deductive reasoning-based approach to control and plan the actions of multiple housekeeping robots whose aim is to tidy up a house in the shortest possible time and to avoid collisions between themselves and other obstacles. Reasoning-based approaches have the capability of explicitly representing domain knowledge; however, a path planning system for complex environments based only on a deductive reasoning-based method or similar approaches might be insufficient, as current implementations cannot handle an excessive increase of the search space and generalise on different environments (Erdem et al., 2012).

3 Materials and methods

In this work, we propose a novel approach for path planning in 3-D complex environments that combines IL and Deductive Reasoning (DR); we refer to it as the *ILDR* method. In this way we can exploit all the advantages of the first technique (scalability, low computation time, capacity to plan in a complex environment taking into account kinematic constraints of the robot providing a smooth path if necessary) and the second technique (completeness, optimality, capability of explicitly encoding medical knowledge, solid theoretical bases coupled with a fully declarative nature) which allows to produce formal specifications that are already executable without the need for additional algorithmic coding, thus fostering fast prototyping and easing the interaction with domain experts. The novel aspects of this approach is that it not only includes all the fundamental requirements of the path planning task, but take also into account the expert's knowledge to fully understand the decision-making process that guides the optimal path selection.

To test viability and effectiveness of our approach, we have chosen keyhole neurosurgery as a case study. In this context, path planning is crucial when the pathological target (e.g., a tumor or left subthalamic nucleus (LSTN)) is located in deep brain areas and cannot be safely reached by a flexible probe. Thus, the optimization criteria involved in finding the best path in this case scenario are: the maximisation of the minimum and average distance with to obstacles, so as to

avoid delicate structures in the brain and the minimisation of the length and curvature of the path, so as to limit damage to the brain matter.

3.1 Problem statement

3.1.1 Moving agent

Let us consider an "agent", showed in Fig. 1a, moving in a 3D complex environment. The agent in this work is the tip of a steerable needle, a new biomimetic flexible probe (PBN) (Burrows et al., 2013), that can translate and rotate in space. Its kinematic constraints are the "diameter", d_{out} , and the "maximum curvature", k_{max} .

3.1.2 Environment

The "3D complex environment" (Env) is showed in Fig. 1b. The "configuration space", $C-space$, is the set of all the possible t "agent configurations", \mathbf{T}_{agent_t} , with $t \in C-space$. The agent configurations are described by poses, denoted as 4×4 transformation matrices:

$$\mathbf{T}_{agent_t} = \begin{pmatrix} \mathbf{R}_{agent_t} & \mathbf{p}_{agent_t} \\ \mathbf{0} & 1 \end{pmatrix}$$

where $\mathbf{p}_{agent_t} \in \mathbb{R}^3$ is the tip position and $\mathbf{R}_{agent_t} \in SO(3)$ is the orientation relative to a world coordinate frame.

The "obstacle space", $C_{obst} \in C-space$, is the space occupied by obstacles. The "free space" $C_{free} \in C-space$, is the set of possible configurations ($\mathbf{T}_{agent} = \mathbf{T}_{free} \in C_{free}$; $\mathbf{T}_{agent} \neq \mathbf{T}_{obst} \in C_{obst}$). Moreover we can define: the "start configurations" $\mathbf{T}_{start_k} \in C_{free}$ with $k \in 1..N$, the "target space" $C_{target} \in C_{free}$, that is a subspace of "free space" which denotes where we want the needle to move to, and the "target configuration" $\mathbf{T}_{target} \in C_{target}$.

3.1.3 Actions

The agent at every t -th time step can take an action $a_t = (x_t, y_t, z_t, \alpha_t, \beta_t, \gamma_t)$, moving towards the \mathbf{T}_{target} , where x, y and z are the axes and α, β, γ the angles around the axes respectively. Actions moving the agent toward an \mathbf{T}_{obst} or outside the Env are considered inadmissible.

3.1.4 Path planning problem

The path planning problem, described in Fig. 2a, can be formulated as follows: given the "start configurations" \mathbf{T}_{start_k} $k = 1 : N$ and a "target configuration", \mathbf{T}_{target} , the task is to find a path, $Q = \{\mathbf{T}_{agent_0}, \mathbf{T}_{agent_1}, \dots, \mathbf{T}_{agent_{n-1}}\}$, $\mathbf{T}_{agent_0} = \mathbf{T}_{start_k}$, $\mathbf{T}_{agent_{n-1}} = \mathbf{T}_{target}$, and

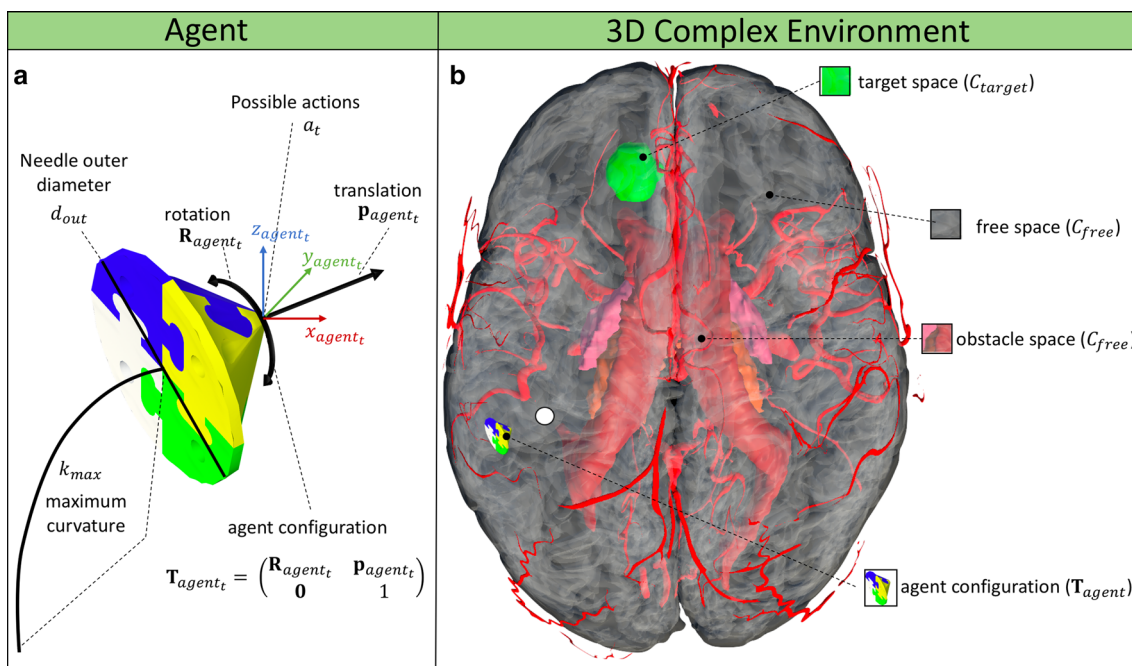


Fig. 1 Agent and 3D complex environment. **a** (on the left) The moving agent kinematic constraints are the needle diameter (d_{out}) and the maximum curvature (k_{max}) that it can perform. At t th time step it can translate (\mathbf{p}_{agent_t}) and rotate (\mathbf{R}_{agent_t}), performing an action a_t , from its

configuration, \mathbf{T}_{agent_t} . **b** (on the right) The 3D complex environment is represented by a brain structure, the obstacle space (C_{obst}), the free space (C_{free}), the agent configuration (\mathbf{T}_{agent_t})

\mathbf{T}_{agent_t} and $\mathbf{T}_{agent_{t+1}}$ are connected by straight segments, as an admissible sequence of “agent configurations”.

3.2 Inductive learning-based and deductive reasoning-based (ILDR) architecture

The herein proposed approach to path planning in 3-D complex environments, referred to as the ILDR method, is summarized in Fig. 2b. The best path is obtained by means of a learning approach that generates a set of path candidates and an expert classifier that selects the “preferred” one, according to a set of rules set by the domain expert.

3.2.1 Inductive learning model

As shown in Fig. 3, the inductive component iteratively trains the moving agent to generate optimal paths, $\{Q_k^{IL}\}$, thanks to expert’s manual demonstrations, $\{Q^{Manual}\}$.

3.2.2 Generator

With several successful applications in robot control, RL that searches for optimal policies by interacting with the environment becomes one potential solution to learn how to plan a path. Segato et al. (2020), based on DRL, demonstrated its capability of directly learning the complicated policy of planning a path for steerable needle.

In the proposed approach, the agent interacts with the environment, Env . At each time step (t), the agent updates its current state (\mathbf{T}_{agent_t}) and selects an action (a_t), according to the policy (π), such that:

$$\pi(\mathbf{T}_{agent_t}, Env) = a_t \tag{1}$$

In response, the agent receives the next state ($\mathbf{T}_{agent_{t+1}}$) and observations (explained in the next paragraph). 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 cumulative extrinsic reward $r^{ex}(\mathbf{T}_{agent_t}, a)$. The latter is associated to the real, full state of the system and to the agent’s observations. In this way, the moving agent successfully learns to follow the best path in accomplishing its task and its constraints of maximum curvature, k_{max} , and diameter, d_{out} .

Observations The observations collected step-by-step by the agent while exploring the environment are: the length (d_{tot}) of the path, the minimum (d_{min}) and average (d_{avg}) distances from the obstacle space $\{C_{Obst}\}$, the maximum curvature of the path (c_{max}), the target angle (α_{target}) and the computational time (T).

For each path $Q^G = \{\mathbf{T}_{agent_0}, \mathbf{T}_{agent_1}, \dots, \mathbf{T}_{agent_{n-1}}\}$ where $\mathbf{T}_{agent_0} = \mathbf{T}_{start_k}$ $k = 1 : N$, the observations are:

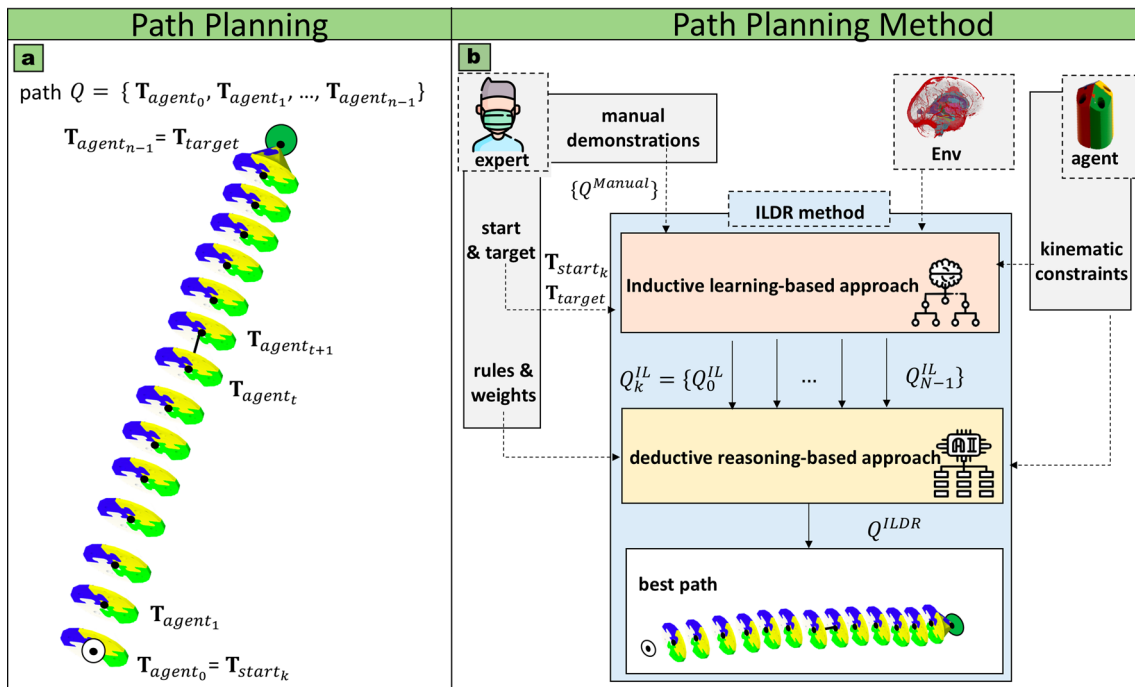


Fig. 2 Path planning problem and method. **a** Schematic representation of the 3D path planning problem elements, start configuration ($\mathbf{T}_{start_k} = \mathbf{T}_{agent_0}$) and target configuration ($\mathbf{T}_{target} = \mathbf{T}_{agent_{n-1}}$). The task is to find the optimal path ($Q = \{\mathbf{T}_{agent_0}, \mathbf{T}_{agent_1}, \dots, \mathbf{T}_{agent_{n-1}}\}$). At every t th time step \mathbf{T}_{agent_t} and $\mathbf{T}_{agent_{t+1}}$ are connected by straight segments, as an admissible sequence of “agent configurations”, taking into account the “obstacle space”, C_{obst} . **b** The proposed path planning algorithm exploits two merged approaches: inductive learning-based

and deductive reasoning-based approaches. The expert indicates N start configurations, (\mathbf{T}_{start_k} with $k \in N$), the target configuration, (\mathbf{T}_{target}) the rules and performs a set of demonstrations. While the agent acts and observes in the virtual environment, the expert’s demonstrations and the agent’s observations feed the inductive approach that generates a set of optimal trajectories $\{Q_k^{IL}\}$ with k equal to the N start configurations. The rules and the kinematic constraints feed the deductive approach that extracts the best path Q^{ILDR} .

- Path length (d_{tot}): the distance, d , between any two positions can be calculated based on the Euclidean distance:

$$d(\overline{\mathbf{p}_{agent_t} \mathbf{p}_{agent_{t+1}}}) = \|\mathbf{p}_{agent_t} - \mathbf{p}_{agent_{t+1}}\| \quad (2)$$

with $t \in 1, \dots, n$, $n = \|\mathcal{Q}^G$ and

$$d_{tot} = \sum_{t=0}^{n-1} d(\overline{\mathbf{p}_{agent_t} \mathbf{p}_{agent_{t+1}}}) \quad (3)$$

- Minimum distance (d_{min}): given the the line segment ($\overline{\mathbf{p}_{agent_t} \mathbf{p}_{agent_{t+1}}}$) and the m obstacles represented by the occupied configurations $\mathbf{T}_{obst_j} = \begin{pmatrix} \mathbf{R}_{obst_j} & \mathbf{p}_{obst_j} \\ \mathbf{0} & 1 \end{pmatrix}$ with $j \in 1, \dots, m$, $m = \|C_{obst}$, the minimum distance of the path from the nearest obstacle indicating the level of safety is the minimum length between the line segments and the closest obstacle, such that:

$$d_{min} = \min\{d(\overline{\mathbf{p}_{agent_t} \mathbf{p}_{agent_{t+1}}}, \mathbf{p}_{obst_j})\} \forall t, \forall j \quad (4)$$

- Average distance (d_{avg}) of the path from all the nearest m obstacles: it is calculated over the whole length of the path with respect to all the obstacles, such that:

$$d_{avg} = \frac{1}{n \cdot m} \sum_{t=1}^n \sum_{j=1}^m d(\overline{\mathbf{p}_{agent_t} \mathbf{p}_{agent_{t+1}}}, \mathbf{p}_{obst_j}) \forall t, \forall j \quad (5)$$

- Maximum curvature (c_{max}) of the path: curvature c of a path in a 3-D space is the inverse of the radius r_t of a sphere passing through 4 positions of the path ($\mathbf{p}_{agent_t}, \mathbf{p}_{agent_{t+1}}, \mathbf{p}_{agent_{t+2}}, \mathbf{p}_{agent_{t+3}}$) computed for each t (the method is explained comprehensively in “Appendix A.1”). Subsequently, the maximum curvature can be extracted as follows:

$$c_t = \frac{1}{r_t} \quad (6)$$

$$c_{max} = \max_{0 \leq t < n-3} c_t \quad (7)$$

- Target angle (α_{target}): given the 3-D unit vector representing the agent direction $\overline{\mathbf{p}_{agent_t} \mathbf{p}_{agent_{t+1}}}$ and the one representing the target direction $\overline{\mathbf{p}_{agent_t} \mathbf{p}_{target}}$, the target angle is defined as follows:

$$\alpha_{target} = \arccos(\overline{\mathbf{p}_{agent_t} \mathbf{p}_{agent_{t+1}}} \cdot \overline{\mathbf{p}_{agent_t} \mathbf{p}_{target}}) \quad (8)$$

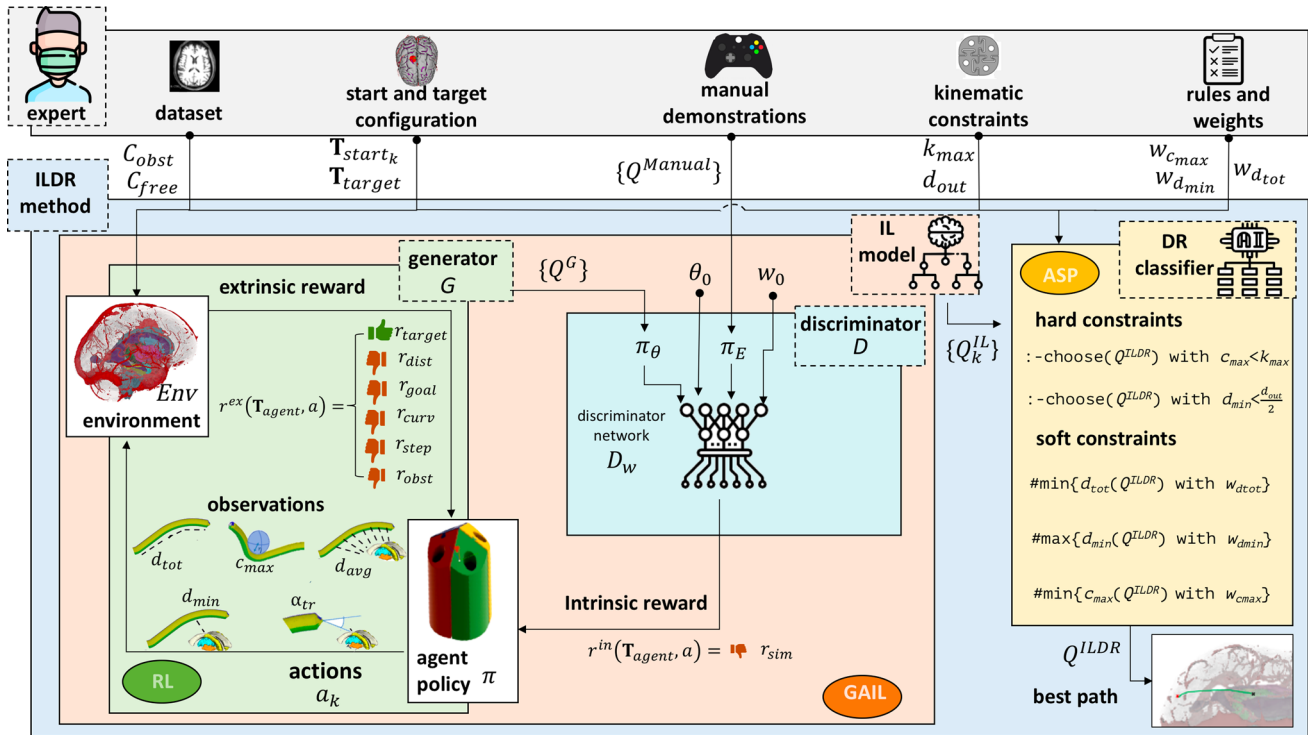


Fig. 3 ILDR architecture. The expert gives in input the dataset (C_{free} and C_{obst}), the start and target configuration (\mathbf{T}_{start} and \mathbf{T}_{target}) and the kinematics constraints (d_{out} and k_{max}). The IL model is trained through a loop that starts with paths generated by an expert ($\{Q^{Manual}\}$) and paths ($\{Q^G\}$) performed by the network generator (G) based on a Reinforcement Learning (RL) approach. With a Generative Adversarial Imitation Learning (GAIL) approach a discriminator (D) with its network (D_w), takes in input the expert and generator policies (π_E and π_θ) the starting value of the policy’s parameters and of the discriminator (θ_0 and w_0). During the training phase, it makes a comparison of these two paths generating an intrinsic reward ($r^{in}(\mathbf{T}_{agent}, a)$) based

on the similarity score (r_{sim}), updating the agent’s policy π . The loop continues until the generator, moving in the environment (Env) with actions (a_t), collecting observations (d_{tot} , d_{min} , d_{avg} , c_{max} and α_{tr}) and computing an extrinsic reward ($r^{ex}(\mathbf{T}_{agent}, a)$), can produce a path similar to the expert’s demonstrations and that respects the kinematic constraints. Once the IL model is trained, it generates a set of optimal paths; the weights ($w_{d_{tot}}$, $w_{d_{min}}$ and $w_{c_{max}}$) and the kinematic constraints are taken as input by the DR classifier that extracts the best path with an approach based on Answer Set Programming (ASP). Applying hard and soft constraints obtaining the best path Q^{ILDR}

Algorithm 1 Reward function - pseudocode for each episode

```

1: for  $t$  in  $S_{max}$  do
2:   Compute action  $a_t$ 
3:   Collect Observations  $d_{min_t}$ ,  $d_{avg_t}$ ,  $d_{tot_t}$ ,  $c_t$  and  $\alpha_{target_t}$ 
4:   addReward( $-1/S_{max}$ )
5:   if  $\mathbf{T}_{agent_t} = \mathbf{T}_{target}$  then
6:     addReward(1)
7:     newEpisode()
8:   else if  $\mathbf{T}_{agent_t} = \mathbf{T}_{obst}$  then
9:     addReward(-1)
10:    newEpisode()
11:   if  $d_{min_t} < d_{safe}$  then
12:     addReward(-0.001)
13:   if  $c_t > k_{max}$  then
14:     addReward(-0.001)
15:   if  $\alpha_{target_t} \neq NaN$  then
16:     addReward( $-\alpha_{target_t} / \alpha_{max} * 0.0001$ )

```

Reward function (Algorithm 1) the reward function associated with each time step t is shaped in order to make the agent

learn to optimise the path, according to three main requirements:

1. path length minimisation
2. obstacle clearance maximisation
3. moving agent’s kinematic constraints respect.

The reward $r^{ex}(\mathbf{T}_{agent}, a)$ is defined as:

$$r^{ex}(\mathbf{T}_{agent_t}, a_t) = \begin{cases} r_{goal_t} & \text{if } \mathbf{T}_{agent_t} = \mathbf{T}_{target} \\ r_{obst_t} & \text{if } \mathbf{T}_{agent_t} = \mathbf{T}_{obst} \\ r_{step_t} + r_{dist_t} + r_{curv_t} + r_{target_t} & \text{otherwise} \end{cases} \quad (9)$$

- A positive constant reward, r_{goal_t} , is given upon reaching the target.
- A negative constant reward, r_{obst_t} , is given if an obstacle collision is detected.
- A negative reward, $r_{step_t} = -\frac{1}{S_{max}}$, is given at each step t of the agent in order to obtain a reduction in the com-

Algorithm 2 Path planning with GAIL

Input: Expert paths $\{Q^{Manual}\} \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0

```

1: for  $i=0,1,2,\dots$  do
2:   Generated paths  $\{Q^G\} \sim \pi_{\theta_i}$ 
3:   Update the  $D_w$  parameters from  $w_i$  to  $w_{i+1}$  with the gradient  $\nabla_w$ :
4:
    $E_{\pi_{\theta_i}} [\nabla_w \log (D_w (\mathbf{T}_{agent}, a))] + E_{\pi_E} [\nabla_w \log (1 - D_w (\mathbf{T}_{agent}, a))]$ 
5:   Take a policy step from  $\theta_i$  and  $\theta_{i+1}$  with Trust Region Policy (TRPO) cost function  $\log(D_{w_{i+1}}(\mathbf{T}_{agent}, a))$  with the gradient  $\nabla_{\theta}$ :
6:
    $E_{\pi_{\theta_i}} [\nabla_{\theta} \log \pi_{\theta}(a|\mathbf{T}_{agent})Q(\mathbf{T}_{agent}, a)] - \lambda H(\pi_{\theta})$ 
7:   Compute reward  $r^{in}(\mathbf{T}_{agent}, a) = -\log(1 - D_w(\mathbf{T}_{agent}, a))$ 
8:   return  $\pi$ 

```

Algorithm 3 Classifying Path with ASP

Input: $\{Q_k^{LL}\}, k_{max}, r, w_{dot}, w_{dmin}$ and w_{cmax} . **Output:** Q^{LLDR} .

```

1: for  $x$  in  $\{Q_k^{LL}\}$  do
2:   Compute  $d_{min_x}, d_{dot_x}$  and  $c_{max_x}$ 
3:   if  $:-$  choose( $Q_x^{LLDR}$ ), curvature( $c_{max_x}$ ), maxCurve( $Q_x^{LLDR}, K_{max_x}$ ),  $k_{max} < c_{max_x}$ . then
4:     if  $:-$  choose( $Q_x^{LLDR}$ ), radius( $r$ ), distObst( $Q_x^{LLDR}, d_{min_x}$ ),  $d_{min} < r$ . then
5:       Sort  $\{Q^{LLDR}\}$  by:
6:         #min $\{d_{dot_x} @ w_{dot} \cdot Q_x^{LLDR}$ : choose( $Q_x^{LLDR}$ ), length( $Q_x^{LLDR}, d_{dot_x}$ )}.
7:         #max $\{d_{min_x} @ w_{dmin} \cdot Q_x^{LLDR}$ : choose( $Q_x^{LLDR}$ ), distObst( $Q_x^{LLDR}, d_{min_x}$ )}.
8:         #min $\{c_{max_x} @ w_{cmax} \cdot Q_x^{LLDR}$ : choose( $Q_x^{LLDR}$ ), maxCurve( $Q_x^{LLDR}, c_{max_x}$ )}.
return  $Q_0^{LLDR}$ 

```

putational time (T), where S_{max} corresponds to a fixed threshold representing the maximum number of steps t .

- A negative constant reward, r_{dist_t} , is added whenever the minimum value of d_{min_t} is lower than a predefined safe distance ($d_{safe} = \frac{d_{out}}{2}$), corresponding to the occupancy of the moving agent. This reward aims to maximise the d_{min_t} to reduce the risk of collision and increase the safety rate of the path.
- A negative constant reward, r_{curv_t} , is assigned if the current path curvature c_t overcomes the maximum value of the curvature of the moving agents specified by the user (k_{max}).
- Finally, a negative reward, $r_{target_t} = -\frac{\alpha_{target_t}}{\alpha_{max}} r_{deg}$, is added to minimise α_{target_t} in order to further minimise c and T parameters. The value of this reward is proportional to the ratio between α_{target_t} and the maximum angle α_{max} .

The parameters of the reward function are reported in Table 2. The optimal parameters' value have been obtained by fine-tuning with grid-search procedure. Sub-optimal parameter configurations caused the agent to learn and apply inappropriate actions, e.g. moving too close to obstacles, going in the opposite direction with respect to the target, choosing non-optimal paths in terms of distance.

In this way, with the generator, G , we are able to obtain new paths, Q^G .

Table 2 Training parameters

r_{goal_t}	r_{obst_t}	r_{curv_t}	r_{dist_t}	S_{max}	α_{max}	r_{deg}
1	-1	-0.001	-0.001	5000	180	-0.001

3.2.3 Discriminator

In GAIL (Ho & Ermon, 2016) the task of the discriminator, D , is to distinguish between the paths, $\{Q^G\}$, generated by G with a RL approach, and the demonstrated paths $\{Q^{Manual}\}$. When D cannot distinguish $\{Q^G\}$ from $\{Q^{Manual}\}$, then G has successfully matched the demonstrated path, $\{Q^{Manual}\}$, reaching a level equal to or higher than one of the expert thanks to the additional observations collected and extrinsic reward received.

As showed in Algorithm 2, the proposed path planning approach receives in input the expert's paths, $\{Q^{Manual}\}$, the path generated from the Generator, $\{Q^G\}$, and the initialization of the policy's parameters, θ_0 , and of the discriminator, w_0 . The path $\{Q^G\}$ fits a parameter policy π_{θ} (where θ represent weights), while the manual demonstration, $\{Q^{Manual}\}$, fits an expert policy, π_E . The discriminator network D_w (w , weights) learns to distinguish the generated policy, π_{θ} , from the expert one, π_E . The parameters w of D_w are updated in order to maximise Eq. 10:

$$\begin{aligned} & \max_{\nabla_w} E_{\pi_\theta} [\nabla_w \log (D_w (\mathbf{T}_{agent}, a))] + \\ & E_{\pi_E} [\nabla_w \log (1 - D_w (\mathbf{T}_{agent}, a))] \end{aligned} \quad (10)$$

where ∇_w is the gradient, $E[\]$ is the expectation value with respect to a policy, π_θ , or to the expert policy, π_E , $D_w (\mathbf{T}_{agent}, a)$ is the Discriminator network that evaluates the state (\mathbf{T}_{agent}) and the action (a).

TRPO (Schulman et al., 2015) assures that the change of parameter θ between $\pi_{\theta+1}$ and π_θ is limited:

$$\begin{aligned} \theta = & \min_{\nabla_\theta} E_{\pi_\theta} [\nabla_\theta \log \pi_\theta(a|\mathbf{T}_{agent}) \cdot Q(\mathbf{T}_{agent}, a)] \\ & - \lambda H(\pi_\theta) \end{aligned} \quad (11)$$

where ∇_θ is the gradient, $H(\pi_\theta) = E_{\pi_\theta} [-\log \pi(a|\mathbf{T}_{agent})]$ is the causal entropy and its value by $\lambda \in \mathfrak{R}, \lambda > 0$.

Finally the discriminator, D , updates the agent’s policy π to be close to the expert policy π_E :

$$\pi = \arg \max_{\pi_\theta} E_{\pi_\theta} [-\log(1 - D_w(\mathbf{T}_{agent}, a))] \quad (12)$$

using an intrinsic reward, defined as $r^{in}(\mathbf{T}_{agent}, a) = r_{sim}$. The similarity reward r_{sim} proportional to $-\log(1 - D_w(\mathbf{T}_{agent}, a))$ is an increasing reward when the results approach of D_w approaches $r_{sim} = 1$, i.e. D_w is not able to

Algorithm 4 Calculation of radius curvature

- 1: Input: four successive position of the path $\mathbf{p}_{agent_1}(x_1, y_1, z_1), \mathbf{p}_{agent_2}(x_2, y_2, z_2), \mathbf{p}_{agent_3}(x_3, y_3, z_3), \mathbf{p}_{agent_4}(x_4, y_4, z_4) \in Q = \{\mathbf{T}_{agent_0}, \mathbf{T}_{agent_1}, \dots, \mathbf{T}_{agent_{n-1}}\}$.
- 2: Output: Ray of the sphere r_i that passes through the input positions.
- 3: $A = \det \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix}$
- 4: **if** $A == 0$ **then**
- 5: $r_i = 0$ ▷ The four positions are coplanar
- 6: **else** ▷ Four positions determine a unique sphere if and only if they are not coplanar
- 7: $T = \det \begin{vmatrix} (x_1^2 + y_1^2 + z_1^2) & x_1 & y_1 & z_1 \\ (x_2^2 + y_2^2 + z_2^2) & x_2 & y_2 & z_2 \\ (x_3^2 + y_3^2 + z_3^2) & x_3 & y_3 & z_3 \\ (x_4^2 + y_4^2 + z_4^2) & x_4 & y_4 & z_4 \end{vmatrix}$ ▷ D, E, F and G are determined through the Cramer’s rule
- 8: **for** $i = 1$ **in** 4 **do**
- 9: $t_i = -(x_i^2 + y_i^2 + z_i^2)$
- 10: $D = \det \begin{vmatrix} t_1 & y_1 & z_1 & 1 \\ t_2 & y_2 & z_2 & 1 \\ t_3 & y_3 & z_3 & 1 \\ t_4 & y_4 & z_4 & 1 \end{vmatrix}$ $E = \det \begin{vmatrix} x_1 & t_1 & z_1 & 1 \\ x_2 & t_2 & z_2 & 1 \\ x_3 & t_3 & z_3 & 1 \\ x_4 & t_4 & z_4 & 1 \end{vmatrix}$
- 11: $F = \det \begin{vmatrix} x_1 & y_1 & t_1 & 1 \\ x_2 & y_2 & t_2 & 1 \\ x_3 & y_3 & t_3 & 1 \\ x_4 & y_4 & t_4 & 1 \end{vmatrix}$ $G = \det \begin{vmatrix} x_1 & y_1 & z_1 & t_1 \\ x_2 & y_2 & z_2 & t_2 \\ x_3 & y_3 & z_3 & t_3 \\ x_4 & y_4 & z_4 & t_4 \end{vmatrix}$
- 12: $C = -(\frac{D}{2}, \frac{E}{2}, \frac{F}{2})$ ▷ Coordinate of the centre of the sphere
- 13: $r_i = \frac{1}{2} \sqrt{(D^2 + E^2 + F^2 - 4G)}$ ▷ Radius of the sphere

discriminate well the two paths. The inverse happens when D_w approaches $r_{sim} = 0$, i.e. it is able to discriminate well between the two paths, and the reward goes to 0.

The trained IL model generates the paths $\{Q_k^{IL}\} k = 1 : N$.

3.2.4 Deductive reasoning classifier

As already introduced above, the optimal path is selected among the N paths given by IL and the paths produced by the deductive module by means of an ASP-based DR Classifier (Gelfond & Lifschitz, 1991; Brewka et al., 2011; Leone & Ricca, 2015). Among modern declarative formalisms, ASP, originally developed in the field of logic programming and non-monotonic reasoning, has become widely used in AI, and it is recognized as an effective tool for Knowledge Representation and Reasoning (KRR). This is especially due to its high expressiveness and the ability to deal with incomplete knowledge, and also because of the availability of robust and efficient solvers; more details on ASP are given in “Appendix A.2”.

A domain expert provided the knowledge that we encoded into a logic program consisting of rules and constraints and readable by the ASP solver. For each instance of the path planing problem, the ASP program is hence coupled to a proper representation of the pool of paths calculated by the IL model, $\{Q_k^{IL}\}$ (the “input”) and fed to the ASP solver, thus choosing the best one, $\{Q^{ILDR}\}$. In the following, we provide more details about the encoded deductive path selector.

A path Q_x^{ILDR} cannot be selected if the maximum curvature c_{max} (Eq. 7) measured for Q_x^{ILDR} is bigger than the k_{max} , as it can be seen in the following code snippet (In order to fully understand the syntax, we have provided some examples in “Appendix A.2”):

```
:- choose(Q_x^{ILDR}, curvature(c_{max}),
maxCurve(Q_x^{ILDR}, k_{max}), k_{max} < c_{max}.
```

where the atom $\text{maxCurve}(Q_x^{ILDR}, c_{max})$ couples the path Q_x^{ILDR} with the maximum curvature c_{max} .

A path Q_x^{ILDR} must be discarded also if it approaches the sensitive structures at a minimum distance d_{min} (Eq. 4) smaller than the radius ($r = \frac{d_{out}}{2}$) of the moving agent:

```
:- choose(Q_x^{ILDR}, radius(r),
distObst(Q_x^{ILDR}, d_{min}), d_{min} < r.
```

Along with these hard constraints, that make inappropriate paths to be discarded, we identified several criteria for expressing preferences among admissible ones. In particular, starting from the expert’s knowledge, we designed some “soft constraints”, and three corresponding weights ($w_{d_{tot}}$,

$w_{d_{min}}$ and $w_{c_{max}}$, $w \in \mathbb{R} > 0$) used to express preferences towards paths that feature minimum (or maximum) adherence to the criteria.

1. Minimization of the path length (d_{tot}):

$$\# \text{minimize} \{ d_{tot} @ w_{d_{tot}}, Q_x^{ILDR} : \text{choose}(Q_x^{ILDR}), \text{length}(Q_x^{ILDR}, d_{tot}) \}.$$

where d_{tot} is the length of path Q_x^{ILDR} .

2. Maximization of the distance from obstacles:

$$\# \text{maximize} \{ d_{min} @ w_{d_{min}}, Q_x^{ILDR} : \text{choose}(Q_x^{ILDR}), \text{distObst}(Q_x^{ILDR}, d_{min}) \}.$$

3. Minimization of the curvature of the moving agent:

$$\# \text{minimize} \{ c_{max} @ w_{c_{max}}, Q_x^{ILDR} : \text{choose}(Q_x^{ILDR}), \text{maxCurve}(Q_x^{ILDR}, c_{max}) \}.$$

Some additional insights are provided in description of Algorithm 3.

It is worth noting that the purely declarative nature of ASP easily allows fine-tuning the desiderata by combining the constraints. Not only the system can easily be improved if new or more specific knowledge is available from the experts, but the user can change the behavior of the classifier at will when in use. The provided interface gives the user the possibility to compose the desiderata and repeat this step multiple times, after the IL model has generated the output, $\{Q_k^{IL}\}$ (agnostic to the human-chosen desiderata), changing the inputs until she is satisfied with the obtained path. In this case, new weights and more (or less) restrictive constraints (i.e., increasing k_{max} or d_{out}) can be set. Furthermore, the user can decide to take into account all, some, or none of the encoded preferences, depending on the specific application; if she chooses to apply more than one of these rules, then also different priorities can be set, expressed by the weights. Hence, the capability to customize the set of rules and (hard/soft) constraints to use for each case study makes our tool highly flexible and generalizable.

4 Experimental protocol

Criteria for defining the “best” surgical path are several; their importance depends on the application at hand. In our experiments, performed in static simulated environments, we focused on Deep Brain Stimulation (DBS) and Convection Enhanced Delivery (CED), which are two relevant applications of steerable needles in keyhole neurosurgery.

4.1 Neurosurgical environment

3-D brain structures for CED and DBS environment were identified on two datasets: (1) Time-of-Flight (ToF) Magnetic Resonance (MR) for vessels visualisation, (2) T1 for brain cortex, skull surface, arterial blood vessels, ventricles and all the relevant deep grey matter structures visualisation, segmented through FreeSurfer Software (Fischl, 2012) and 3-D Slicer (Pieper et al., 2004).

- As reported in Fig. 4a, in CED the target space is the tumor= C_{target} , surrounded by different essential structures (ventricles, thalamus, pallidum and vessels), that represent the obstacle space, C_{obst} , while gyri represent the free space, C_{free} .
- As reported in Fig. 4b, in DBS the target space is the LSTN= C_{target} , located in the central brain core. The obstacle space, C_{obst} , is represented by relevant structures (ventricles, thalamus, pallidum, vessels and CST), while gyri represent the free space, C_{free} .

4.2 Neurosurgical simulator

A planning tool is implemented in 3D Unity (Goldstone, 2009) and integrated with Machine Learning (ML)-Agents Toolkit (Juliani et al., 2018) that allows to visualize the 3D segmented risk structures of the brain of the patient derived from that data. We designed and developed a neurosurgical simulator, i.e. a “Brain Digital Twin”, described in Fig. 5 and shown in the animation (Online Resource 6.2) to create the environment, collect manual demonstrations by the expert surgeon with a joystick controller (with a combination of the translation along the z axis and the rotation around the x and y axes) and train the moving agent. First of all the surgeon is asked to choose the curvature k_{max} , and diameter d_{out} , and the parameters he/she wants to prioritize in the selection of the best trajectory, $w_{d_{tot}}$, $w_{d_{min}}$ and $w_{c_{max}}$, used either to minimize or to maximize the rule expression. The surgeon can then select a target configuration in the brain (\mathbf{T}_{target}), e.g. on the tumor and N start configurations (\mathbf{T}_{start_k}), on the brain cortex. Once the start and target configurations are defined, it is possible to proceed to either a manual or automatic, pre- or intra-operative procedure. The difference between these last two is the dynamism of the environment in representing the needle-tissue interaction and so the non-holonomic constraints that affect the behaviour and the needle movements. The surgeon can in fact proceed with a static environment or with a dynamic environment whose model is based on a Positions Based Dynamics simulator (Segato et al., 2021b) used to emulate the brain tissues deformations in Key-hole Neurosurgery (KN). The needle model is considered as particle system (O’Brien et

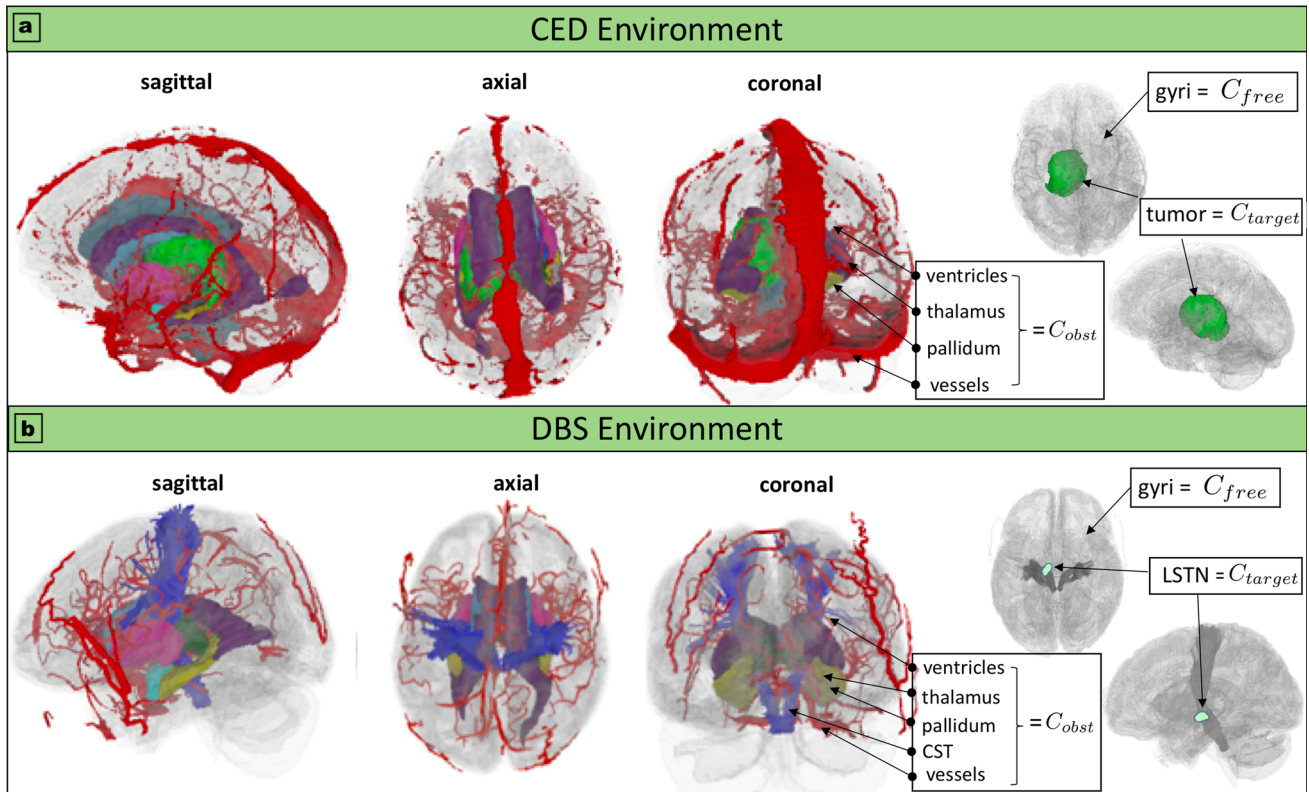


Fig. 4 CED and DBS environments. In **a**, sagittal, axial and coronal view of the CED environment are reported. The obstacle space ($C_{obst} = \{\text{ventricles, thalamus, pallidum and vessels}\}$), the free space ($C_{free} = \text{gyri}$) and the target space ($C_{target} = \text{tumor}$) were defined accordingly. In **b**, sagittal, axial and coronal view of the DBS

environment are reported. Defining accordingly the obstacle space ($C_{obst} = \{\text{ventricles, thalamus, pallidum, vessels and Corticospinal Tract (CST)}\}$), the free space ($C_{free} = \text{gyri}$) and the target space ($C_{target} = \text{LSTN}$)

al., 2001). For the validation a pre-operative procedure with a static environment was considered. It is assumed that the motion of the needle tip fully determines the motion of the needle (“follow-the-leader” deployment) with a combination of the translation along the z axis and the rotation around the x and y axes.

4.3 Experimental validation

The results’ assessment for both scenarios, CED and DBS, is based on the comparison of the proposed method, ILDR, with the Manual and DR approaches. Moreover, in the DBS scenario, ILDR was tested against the Rapidly-exploring Random Trees (RRT)* algorithm.

As shown in Fig. 6, an expert surgeon (age: 37, performed surgical procedures: 2440) was asked to select, for each environment Env_s , 10 desired start configurations, $\mathbf{T}_{start_k}^s$, on the brain cortex, a target configuration, \mathbf{T}_{target}^s , on the target space, C_{target} , and the weights, $w_{c_{max}}^s$, $w_{d_{min}}^s$, $w_{d_{tot}}^s$ for the rules prioritisation, reported for both scenarios in Table 3. j experiments, EXP_j (with $1 \leq j \leq 5$), were conducted for each one of the four approaches: Manual, DR, RRT* (only for DBS Environment) and ILDR.

- Manual approach: For each EXP_j , the surgeon was asked to generate a pool of surgical paths, $\{path_k\}$ (with $1 \leq k \leq 10$), and choose the optimal one, $path_j^{Manual,s}$, based on his expertise.
- DR approach: For each EXP_j was considered the same pool of surgical paths generated in the manual approach, $\{path_k\}$, and the optimal one, $\{path_j^{DR,s}\}$, was selected with the DR classifier, using rules, weights and kinematic constraints given in input by the surgeon.
- RRT* approach: For each EXP_j the pool of paths, $\{path_k\}$, was generated with the RRT* algorithm. The optimal one, $\{path_j^{RRT*,s}\}$, was selected with a Cost Function F_{cost} , to be minimised:

$$F_{cost}(\{path_k\}) = \begin{cases} \infty & \text{if } d_{min} \leq 0 \\ \infty & \text{if } c_{max} > k_{max} \\ w_{d_{min}} \frac{1}{d_{min}} + w_{d_{tot}} \frac{1}{d_{tot}} + w_{c_{max}} \frac{c_{max}}{k_{max}} & \text{otherwise} \end{cases} \quad (13)$$

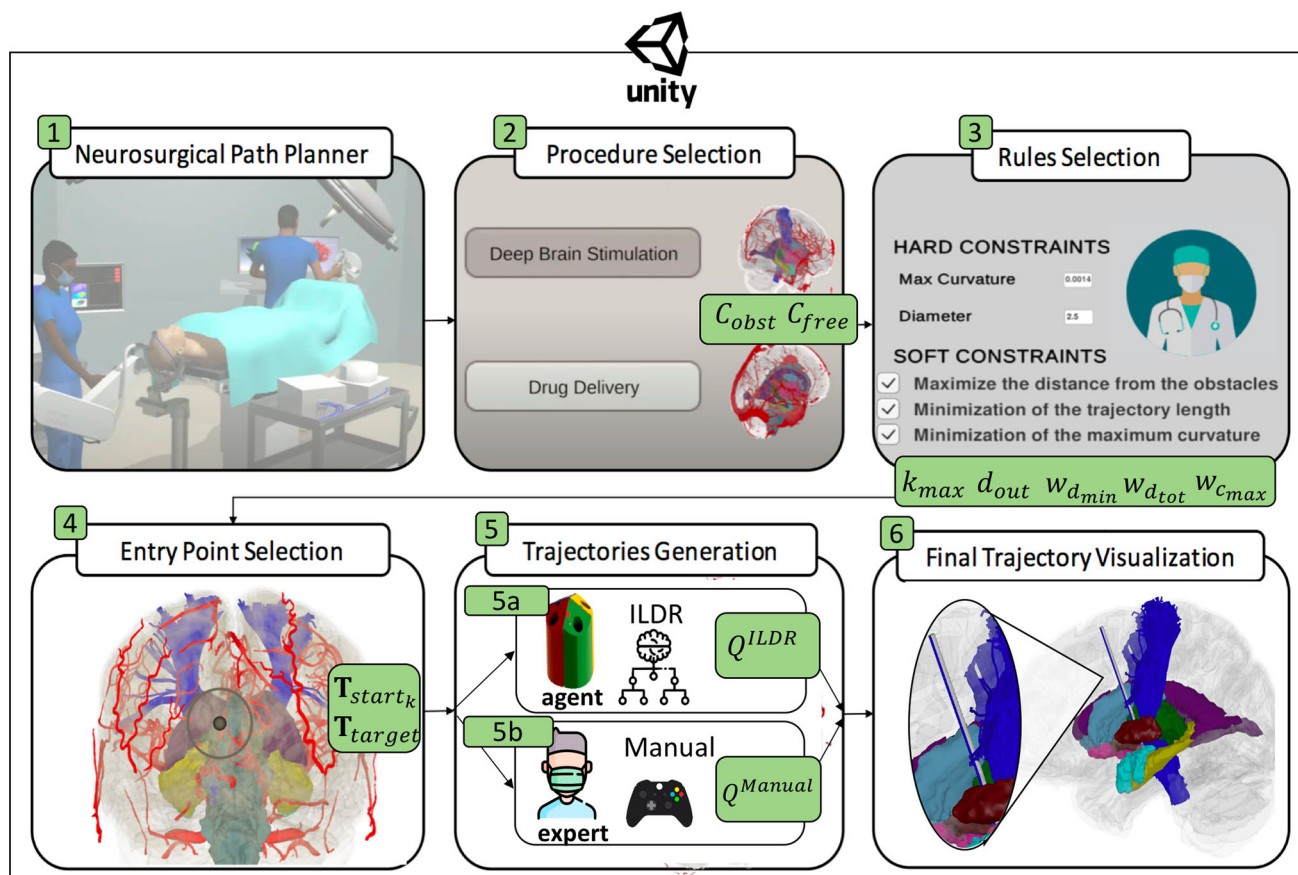


Fig. 5 Neurosurgical simulator. It is possible to: (1) start a simulation, (2) select the procedure between the DBS and CED, (3) input the physical characteristics of the needle, d_{out} and k_{max} , and the parameters to be maximised or minimised in the selection of the best path,

(4) select starting and target configurations, \mathbf{T}_{start_k} and \mathbf{T}_{target} , (5) perform manually the trajectories with a joystick controller or let the agent to perform trajectories autonomously with ILDR and finally (6) visualise all the generated trajectories

Using rules, weights and kinematic constraints given in input by the surgeon. For more information on the implementation of this approach please refer to our previous work Segato et al. (2019).

- ILDR approach: For each EXP_j the pool of paths, $\{path_k\}$, was generated with the IL model. The optimal one, $\{path_j^{ILDR,s}\}$, was selected with the DR classifier, using rules, weights and kinematic constraints given in input by the surgeon.

For each path, $\{path_j^s\}$, we calculated:

- The length (d_{tot}) of the path, as described in Eq. 3;
- The minimum (d_{min}) and the mean (d_{avg}) distances of the path with respect to all the obstacles, as described in Eqs. 4 and 5;
- The maximum curvature (c_{max}) of the path, as described in Eq. 7.

4.4 Hardware specification

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

4.5 IL training strategy

The training phase, for IL models, for each Environment, takes in input w start (\mathbf{T}_{start_w} , with $1 \leq w \leq 20$) and target z (\mathbf{T}_{target_z} , with $1 \leq z \leq 5$) configurations and y expert manual path (Q_y^{manual} , with $1 \leq y \leq 10$) for each start and target. The number of manual demonstrations $\|x = 1000$, (with $1 \leq x \leq \|w \times \|z \times \|y$), is obtained by combining the number of demonstrations ($\|y$) provided by the expert user for each couple of start and target ($\|w \times \|z$). At every episode randomly, a new \mathbf{T}_{start_w} was chosen among the available ones along with its relative \mathbf{q}_{target_z} . Table 4 presents the training parameters values referred to the CED and DBS models.

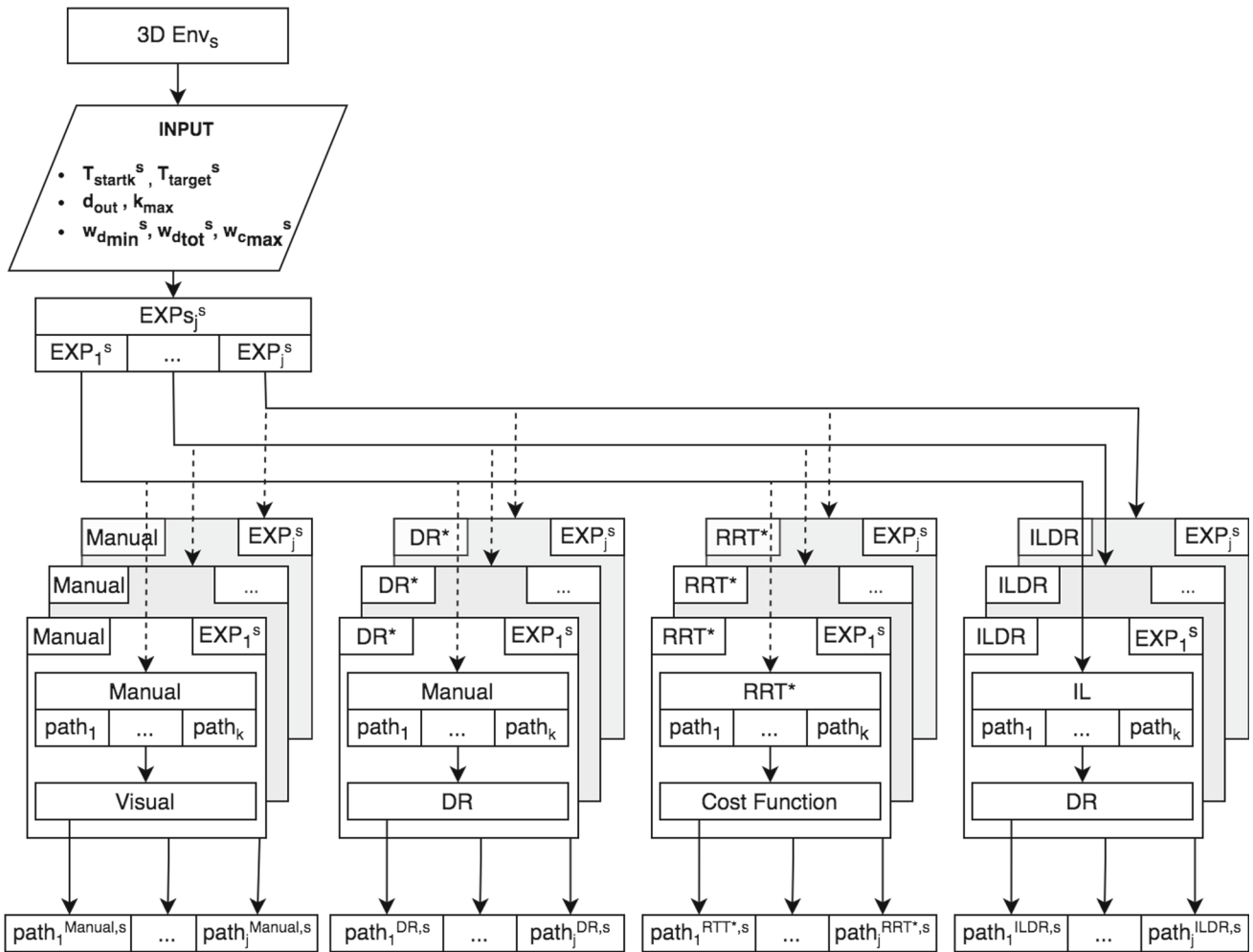


Fig. 6 Schematic representation of the experimental protocol workflow. Each environment Env_s (with $1 \leq s \leq 2$), takes in input the start configurations, T_{start}^s (with $1 \leq k \leq 10$), the target configuration, T_{target}^s , the weights, $w_{cmax}^s, w_{dmin}^s, w_{dtot}^s$ and the kinematic constraints of the

moving agent, d_{out} and k_{max} . j experiments, EXP_j^s (with $1 \leq j \leq 5$), were conducted for each approach: Manual, DR, RRT* (only for DBS Environment) and ILDR. A pool of surgical paths, $\{path_k\}$ is generated, and the optimal one, $path_j^s$, is selected

Table 3 Expert constraints and rules

	$d_{out} (mm)$	$k_{max} (mm^{-1})$	w_{dmin}	w_{dtot}	w_{cmax}
CED	2.5	0.014	9	6	6
DBS	2.5	0.014	1	4	2

Table 4 Training parameters

Parameter	Value	Parameter	Value
Beta	5.0e−4	Max steps	1.0e5
Batch size	64	Buffer size	256

4.6 Statistical analysis

All the performance metrics ($d_{tot}, d_{min}, d_{avg}$ and c_{max}), extracted from the path, were analysed employing Matlab

(The MathWorks, Natick, Massachusetts, R2020a). Lilliefors test has been initially applied for data normality. Due to the non-normality of data distribution, pairwise comparison was performed with the Wilcoxon matched-pairs signed-rank test. Differences were considered statistically significant at p value < 0.05 .

5 Results

5.1 Convection enhanced delivery

Figure 7a shows a comparison between Manual, DR and ILDR approaches in terms of $d_{min}, d_{avg}, c_{max}$ and d_{tot} calculated over the best path of left hemisphere (for each approach, the criteria for the selection of the best path have been described in Sect. 4.3). DR approach keeps a greater

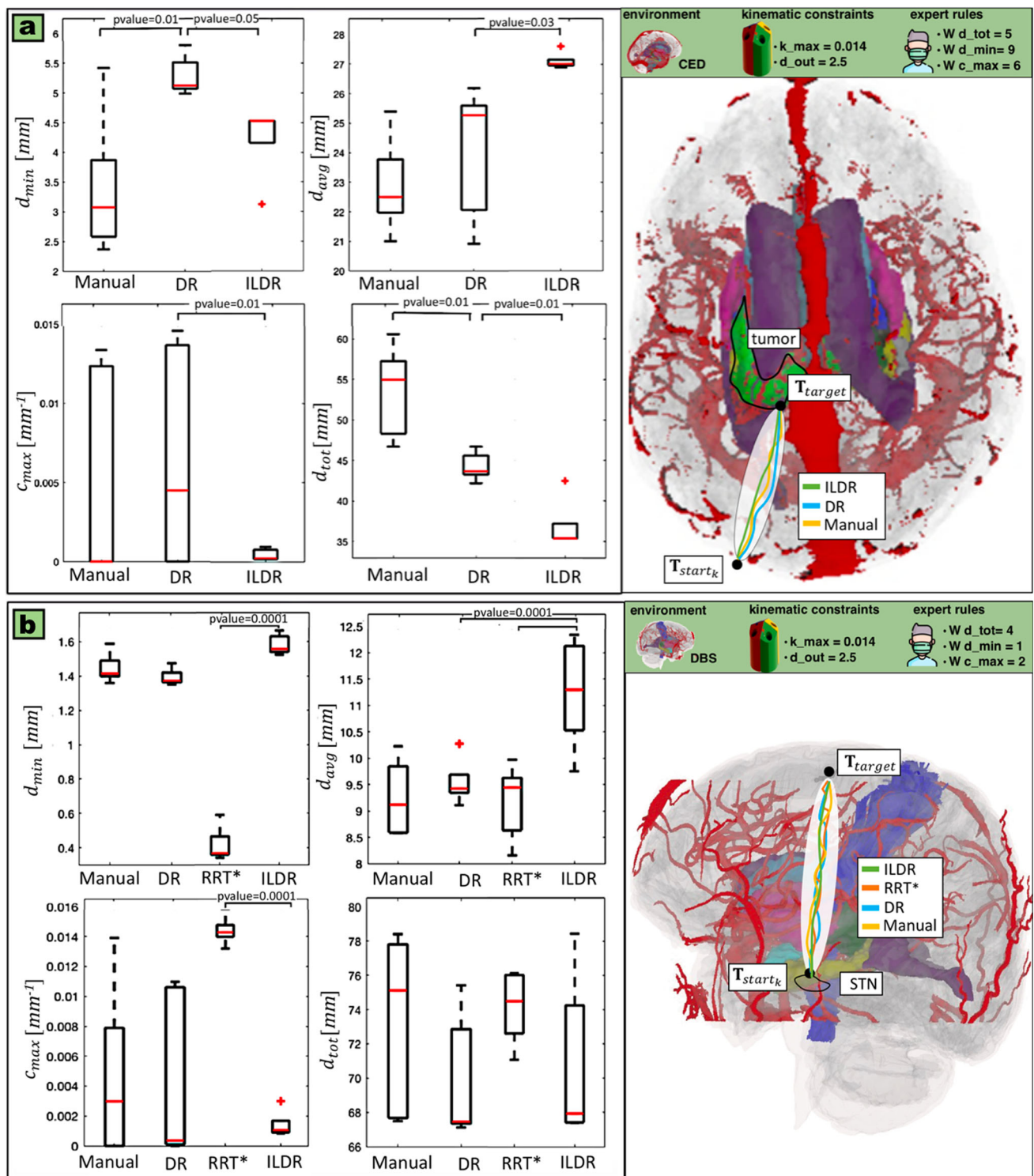


Fig. 7 Comparison between the different approaches. For both CED (a) and DBS (b) environment, expert's rules, and needle kinematic constraints are reported in the upper part. **a** A comparison between Manual, DR, and ILDR approaches in the CED environment. **b** A comparison between Manual, DR, RRT*, and ILDR in the DBS environment. The

results for both considered scenarios and used approaches are reported in terms of the minimum (d_{min}) and the mean (d_{avg}) distance from the critical obstacles, the total path length (d_{tot}) and the curvature (c_{max}) calculated over the five best paths for each approach. P values were calculated using Wilcoxon matched-pairs signed-rank test

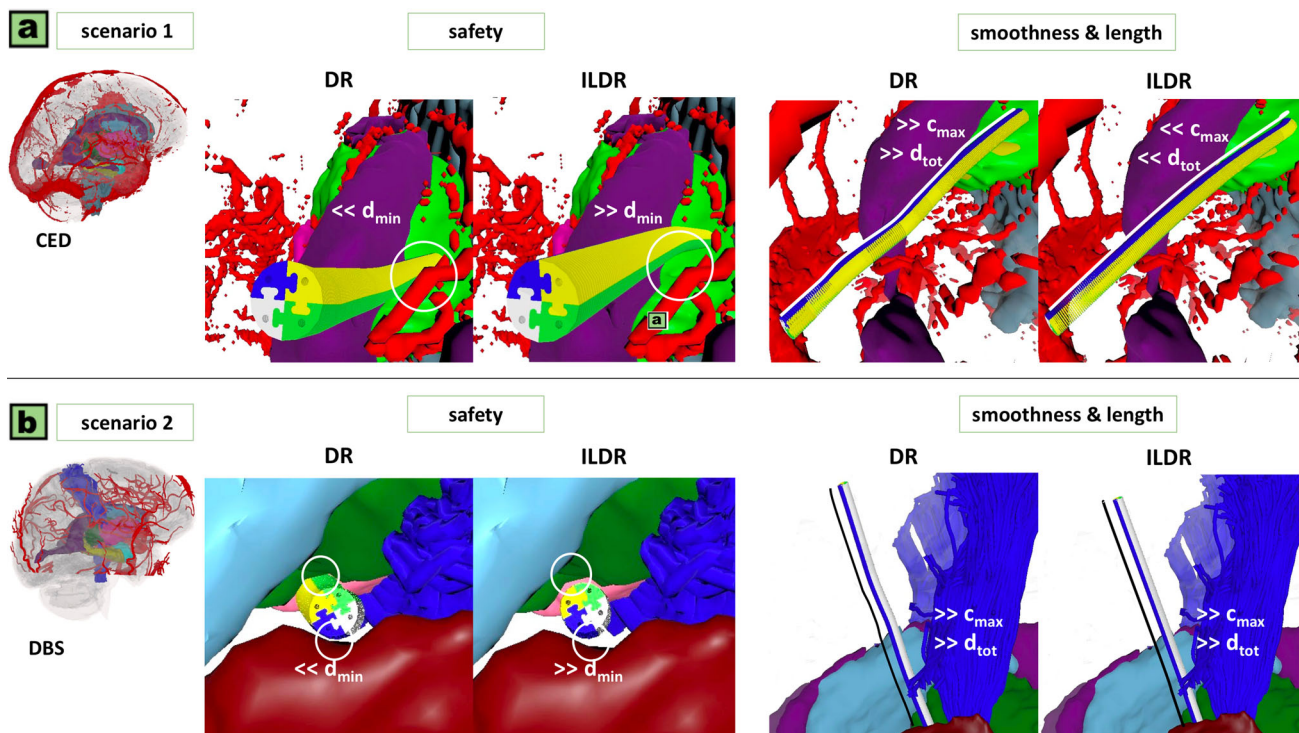


Fig. 8 Comparison between DR and ILDR approaches. For both CED (scenario 1) and DBS (scenario 2) environment, one example of the obtained path is shown. In particular, **a** a comparison between DR and ILDR approaches in the CED environment. **b** A comparison between

DR and ILDR in the DBS environment. The results for both considered scenarios show an increase in safety ($\gg d_{min}$) and smoothness ($\ll c_{max}$) and a reduction in length ($\ll d_{tot}$) for the proposed method (ILDR)

d_{min} and d_{avg} from obstacles and a significantly lower d_{tot} (p value < 0.01) of the path with respect to Manual approach following the rules dictated by the expert. ILDR approach keeps a significantly greater d_{avg} (p value < 0.05) from obstacles and a significantly lower c_{max} (p value < 0.01) and d_{tot} (p value < 0.01) with respect to DR approach and Manual approach.

Figure 8a shows the visual comparison between one resulting path obtained with DR and ILDR approaches considering the previously mentioned optimisation criteria (safety, smoothness and path length), that result better for the ILDR approach even by visualization inspection.

5.2 Deep brain stimulation

Figure 7b shows a comparison between Manual, DR, RRT* and ILDR approaches in terms of d_{min} , d_{avg} , c_{max} and d_{tot} calculated over the best path of left hemisphere (for each approach, the criteria for the selection of the best path have been described in Sect. 4.3). DR approach keeps lower d_{tot} and c_{max} of the path than the Manual approach following the rules dictated by the expert who gives more importance to these two parameters in this case. While ILDR approach keeps a significantly greater d_{min} (p value < 0.01) and

d_{avg} (p value < 0.01) from obstacles and a lower c_{max} and d_{tot} than the DR approach. The comparison between ILDR and RRT* approaches is showed in terms of d_{min} , d_{avg} , c_{max} and d_{tot} calculated over the best path of left hemisphere. ILDR approach keeps a significantly greater d_{min} (p value < 0.0001) and d_{avg} (p value < 0.0001) from obstacles and a lower c_{max} and a significantly lower d_{tot} (p value < 0.01) than the RRT* approach.

Figure 8b shows the visual comparison between one resulting path obtained with DR and ILDR approaches considering the previously mentioned optimisation criteria (safety, smoothness and path length), that result better for the ILDR approach even by visualization inspection.

5.3 Computational time

In Table 5, the computational time T for all the analysed approaches is reported. The ILDR approach is twice as fast compared to the Manual one and keeps much lower computational time than the state-of-the-art sampling-based method RRT*. Although this value is not essential for the proposed pre-operative procedure that is performed off-line, this demonstrates that the proposed method may be poten-

tially applicable to an intra-operative procedure requiring a fast planning time.

6 Discussion and conclusion

The present work proposes a novel automatic path planning approach, called ILDR, for a moving agent in a complex environment. The complexity of the environment represents the worst-case scenario that grants the applicability and reliability of the method. In our experiments, ILDR performed better in terms of obstacle clearance and moving agent kinematic constraints compliance when tested against the Manual approach, the DR classification approach and the RRT* algorithm. By simultaneously optimising paths according to all the requested features, the proposed method outperforms state-of-the-art approaches in terms of path safety, path length, and computational time.

Our method succeeds in obtaining the optimal paths that can be followed to reach a specific target according to rules set by an expert. This approach allows to fully exploit an expert's knowledge: he/she first performs the demonstrations used to train the GAIL model and then selects the constraints and their priorities, which ultimately lead to the choice of the best path with ASP.

It is worth noting that one of the main contributions of the present work consists in the integration of an inductive learning-based approach with a deductive reasoning-based approach. The inductive learning-based method allows the agent to learn the policy by a set of demonstrations provided by an expert, who can introduce in a path planning algorithm all his requirements and knowledge that cannot always be possible in graph- or sampling-based approaches unless additional optimisation steps are applied with additional computational time. Explicit programming cannot fully cover the complexity of the environment (represented by the human brain in this case, due to the presence of delicate and very complicated anatomical structure, narrow passages), the number of parameters and possible complications that have to be considered during the path planning. For this reason we implemented a DR classifier with a user interface, as described in the final part of Sect. 3.2.4, where the experts can express their individual preferences assigning different weights, thus creating a priority list for maintaining different path planning optimisation criteria (i.e., giving more priority to path safety than path length) while visualizing the trajectory and changing the criteria in real time. The DR method is implemented using ASP, that allowed us enjoy several advantages. First of all, even if the whole machinery can be embedded into a graphical user interface for user's convenience, under the hood we are dealing with knowledge explicitly expressed via a declarative formalism: modifying and adapting the criteria for dropping unwanted paths and

Table 5 Results in term of computational time are shown

Method	25th (s)	Median (s)	75th (s)
Manual	15.58	17.94	21.98
DR	11.25	15.00	32.69
RRT*	35.93	61.54	78.87
ILDR	8.02	8.06	8.10

selecting the preferred one(s) is rather easy. Furthermore, as the specification are formally encoded, once the optimization criteria are well-established, we are ensured that the best path is actually chosen, and if more than one are present with the same "score", then picking one or the other is completely indifferent. If, for some reasons, this turns to be not the case, this means that the criteria should be modified, which, as already stated, can be easily done, especially given that the resulting framework allows one to straightforwardly experiments with this respect.

6.1 Clinical translation

As part of the EU's Horizon EDEN2020 project, the current study proposes a novel automatic planner for steerable needles in keyhole-neurosurgery. Given the environment of the brain, a surgeon-defined start, and a target, the proposed method can provide an optimal path, according to predefined features as insertion length, clearance from safety regions, as blood vessels and eloquent morpho-functional landmarks, and compliance to the needle's kinematics limits. It is intended to provide a state-of-the-art combined technology platform for minimally invasive surgery. When tested against the RRT* approach, the proposed method performed better in terms of path smoothness and clearance from safety regions, significantly decreasing the length and with a sensibly lower computational time. Accordingly to the possibility to perform curvilinear path for STN and tumor targeting, the proposed algorithm allows optimising the fundamental aspects of the DBS and CED and to maximising both the effectiveness and safety of the procedure.

6.2 Future directions

The proposed methodology favours high applicability and generalisability, as it could potentially be applied to different path planning problems. Future perspectives may include the exploitation of this automatic path planner method in many applications additional to keyhole neurosurgical procedures. The development of a surgical simulator defines an example of applications based on anatomical topology but not on anatomical dimensions. It is easy to see that different rules and constraints can be defined upon expert suggestion, thus making our methodology highly customisable and paving

the way to extensions to additional 3-D complex environments, beyond brain surgery. Hence, this approach is widely application-independent and can be adapted to other use cases for path planning in a complex environment, where an expert has a crucial role. The application to a totally different context would require a thorough consultation with domain experts and the creation, if missing, of a specific simulation environment. Nonetheless, once the setup is created for a particular domain, as in the current study with brain surgery, it is easily applicable to different problems in such domain, by simply modifying or adding rules.

Supplementary Information Below is the link to the neurosurgical simulator: <https://youtu.be/OM4x4W9lWkE>.

Acknowledgements Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement. We thank Eden2020 project consortium partners for precious advice during the project activities. We thank A. Castellano and A. Falini for providing the dataset for the DBS environment simulation.

Data and materials availability The MRI datasets used for this study have been acquired in the framework of the European Union project EDEN2020 (<https://www.eden2020.eu>, Grant Agreement No. 688279) and will be made publicly available at the end of the project.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

A Appendix

A.1 Computing the radius of curvatures

In this appendix the method used for computing the radius of curvature of the path is explained in algorithm 4. The method is based on the computation of the equation of the sphere that passes through four successive positions. The equation of a generic 3D sphere is determined by the Eq. 14 where the four coefficients D, E, F, G are unknown constant.

$$x^2 + y^2 + z^2 + D(x) + E(y) + F(z) + G = 0 \quad (14)$$

The radius of this sphere is the radius of curvature. First of all, it is important to note that four positions defined a single sphere if they are not coplanar (Purdy & Smith, 2010). In the opposite case there are no spheres that pass through the four positions or there are infinity number of them. For this reason is necessary to verify if the four positions of the trajectory are coplanar. And this is true if and only if the determinant A of the matrix indicated in the algorithm is equal to zero. Since the positions are on the sphere, the substitution of their coordinates in Eq. 14 generates a system of four equations where the unknowns are the coefficients D, E, F, G (Ram, 2009). For resolving this system the Cramer's rule is applied. Then the radius of the sphere can be computed.

A.2 Answer set programming

With ASP, computational problems in a large variety of scenarios can be described by means of simple and elegant logic programs consisting of a set of rules; solutions to a problem instance are then found by computing the semantics of such programs combined with the representation, usually expressed using factual rules, of the instance at hand.

One of the main advantages of ASP consists of its purely declarative nature: rather than focusing on algorithm design and coding, and thus on *how to solve* a computational problem at hand, with ASP one can focus on *how to describe* such computational problem (or how its solutions should look like), completely avoiding the need for explicitly express the steps to be executed. In turn, order of statements in a ASP logic program is immaterial: explicit updates in the problem specification can be more easily incorporated, thus fostering advantages such as fast prototyping, quick error detection and modularity. Besides, a clean model-theoretic semantics grants correctness; intuitively, an ASP program can be seen as a formal yet executable description of the problem. The basic construct of ASP is a rule, that has a general form of *Head* \leftarrow *Body*; the *Body* is a logic conjunction in which nonmonotonic negation may appear, and *Head* can be either an atomic formula or a logic disjunction. Rules are interpreted according to common sense principles: roughly, the intuitive semantics of a rule corresponds to an implication. The answer set semantics associates a problem specification with none, one, or many intended models, called *answer sets*, each one corresponding to a solution; an ASP program that models a computational problem, coupled with a proper representation of an instance of such problem, can be fed to an implementation of ASP, called ASP solver, in order to actually compute all corresponding answer sets. Efficient and reliable ASP solvers exist, such as DLV/DLV2 (Adrian et al., 2018; Leone et al., 2006), WASP for solving the prepositional part of DLV2 (Alviano et al., 2019) and clingo (Gebser et al., 2019).

For a full description of ASP syntax and semantics, along with examples of its applications in academy and industry, we refer the reader to Calimeri et al. (2020) and Lifschitz (2019) and the vast literature. We briefly recall here some very basic preliminaries useful to understand the herein proposed approach.

A variable or a constant is a term. Variables are denoted by strings starting with some uppercase letter, while constants can either be integers, strings starting with some lowercase letter or quoted strings. If t_1, \dots, t_k are terms (either constants or variables) and p is a predicate symbol of arity k , then $p(t_1, \dots, t_k)$ is an atom of arity k .

A literal l is of the form a or $\text{not } a$, where a is an atom; in the former case l is positive, negative otherwise. A rule is of the following form:

$$a_0 | \dots | a_h : -b_1, \dots, b_n \text{ not } b_{n+1}, \dots, b_m. \quad (15)$$

On the left, the symbol “|” connects atoms that are part of a disjunction in the head, whereas comma separated literals in the right side, i.e., the body, are part of a conjunction. An ASP program is a finite set of rules.

A *fact* is a rule with empty body, and represents a piece of information known to be true (typically, facts stand for the knowledge granted before reasoning, or represent the instance of a problem); usually, a fact is immediately followed by the “.” symbol (i.e., the implication symbol “:-” is omitted). A constraint is a rule with empty head; hard (“strong” or “classical”) and soft (“weak”) constraints can be specified in order to cut out undesired models and express preferences, respectively. Weak constraints are expressed with the symbol $:\sim$ instead of $:-$, that is one used for hard constraints. These latter are conditions that must be satisfied, whereas soft constraints represent conditions that should be fulfilled; intuitively, when a solution violates a soft constraint it pays a cost: this induces an ordering among solutions that allows one to express minimization and/or maximization criteria.

ASP enjoys several additional language features for easing knowledge representation; we mention here *choice rules*, that are a compact way for expressing disjunction of atoms that must adhere to some cardinality conditions and *aggregates*, that can be used for compact representations of properties and inductive definitions using sets of propositions (Alviano & Faber, 2018). The scientific community agreed on a standard language (Calimeri et al., 2020); furthermore, in addition to the standard, several flavours of ASP are supported by solvers, featuring additional constructs such as the `#minimize` and `#maximize` statements for expressing preferences in optimization problems similarly to what can be done via weak constraints.

ASP is a very expressive formalism; indeed, in Eiter et al. (1994) it is proved that disjunctive logic programs under

answer set semantics capture the complexity class Σ_2^P (that is, they allow us to express every property which is decidable in non-deterministic polynomial time with an oracle in NP), and weak constraints make ASP well-suited to represent a wide class of problems (including, e.g., NP optimization problems) in a very natural and compact way (Buccafurri et al., 1997).

The following example briefly illustrates how the common “Guess&Check” paradigm is used for modelling problem with ASP.

Example 1 Let us consider the well-known problem of 3-colorability, which consists of the assignment of three colors to the nodes of a graph in such a way that adjacent nodes always have different colors; this problem is known to be NP-complete (Brandstädt et al., 1998). Suppose that the nodes and the arcs are represented by a set \mathcal{F} of facts with predicates *node* (unary) and *arc* (binary), respectively. Then, the following ASP program allows us to determine the admissible ways of coloring the given graph with the three given colors.

```
r1: color(X, red) | color(X, green) |
    color(X, blue) :- node(X) .
r2: :- arc(X, Y), color(X, C), color(Y, C) .
```

Rule r_1 (*guess*) above states that every node of the graph must be colored as red or green or blue; r_2 (*check*) forbids the assignment of the same color to any couple of adjacent nodes. The minimality of answer sets semantics guarantees that every node is assigned only one color. Thus, there is a one-one correspondence between the solutions of the 3-coloring problem for the instance at hand and the answer sets of $\mathcal{F} \cup \{r_1, r_2\}$: the graph represented by \mathcal{F} is 3-colorable if and only if $\mathcal{F} \cup \{r_1, r_2\}$ has some answer set.

The following example illustrates the use of weak constraints for expressing preferences while dealing with optimization problems.

Example 2 Let us consider the same 3-colorability problem of Example 1. Imagine that we know that some of our graphs are not 3-colorable; as already stated, for such graphs our programs would have no answer sets (meaning that there is no admissible solution). Nevertheless, for such cases we would like to have admissible colorings, even if not complete. The following program, that is slightly different from the previous, allows us to determine all partial colorings.

```
r3: color(X, red) | color(X, green)
    | color(X, blue) | noColor(X) :- node(X) .
r2: :- arc(X, Y), color(X, C), color(Y, C) .
```

Here, rule r_1 of Example 1 is replaced with rule r_2 ; note that now, for the program $\mathcal{F} \cup \{r_3, r_2\}$, admissible answer sets exist with some nodes with no color assigned. These contains answer sets with no node colored at all, others with some colored, and up to others with all node colored, in case the

graph is 3-colorable. We can state that we prefer a solution over another one if it features a higher number of colored nodes by means of the following weak constraint:

$$r_4: \sim \text{node}(X), \text{noColor}(X). [1@1, X]$$

Now, for a solution of $\mathcal{F} \cup \{r_3, r_2, r_4\}$, while if it violates r_2 is inadmissible, and thus discarded, if it violates r_4 then it is assigned a cost as specified in the square brackets. In this latter case, for each node X that has no color, the solution costs 1 at level 1. Weight and level can be constant values or variables appearing in the body of the constraint; furthermore, weights are additive, grouped by levels, among all constraints in the program: in this simple example, just a weak constraint is present, but one can make use of more in order to express complex sets of desiderata. Higher levels correspond to more important desiderata.

References

- Adhami, L., & Coste-Manière, È. (2003). Optimal planning for minimally invasive surgical robots. *IEEE Transactions on Robotics and Automation*, 19(5), 854–863. <https://doi.org/10.1109/TRA.2003.817061>
- Adrian, W. T., Alviano, M., Calimeri, F., Cuteri, B., Dodaro, C., Faber, W., et al. (2018). The ASP system DLV: Advancements and applications. *KI-Künstliche Intelligenz*, 32(2–3), 177–179. <https://doi.org/10.1007/s13218-018-0533-0>
- Al-Khawaldeh, M., Al-Naimi, I., Chen, X., & Moore, P. (2016). Ubiquitous robotics for knowledge-based auto-configuration system within smart home environment. In *2016 7th international conference on information and communication systems (ICICS)* (pp. 139–144). IEEE. <https://doi.org/10.1109/IACS.2016.7476100>
- Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., & Ricca, F. (2019). Evaluation of disjunctive programs in WASP. In *International conference on logic programming and nonmonotonic reasoning (LPNMR)* (pp. 241–255). Springer. https://doi.org/10.1007/978-3-030-20528-7_18
- Alviano, M., & Faber, W. (2018). Aggregates in answer set programming. *KI-Künstliche Intelligenz*, 32(2–3), 119–124. <https://doi.org/10.1007/s13218-018-0545-9>
- Audette, M. A., Bordas, S. P., & Blatt, J. E. (2020). Robotically steered needles: A survey of neurosurgical applications and technical innovations. *Robotic Surgery: Research and Reviews*, 7, 1.
- Bellman, R. (1966). Dynamic programming. *Science*, 153(3731), 34–37. <https://doi.org/10.1126/science.153.3731.34>
- Besada-Portas, E., de la Torre, L., Jesus, M., & de Andrés-Toro, B. (2010). Evolutionary trajectory planner for multiple UAVs in realistic scenarios. *IEEE Transactions on Robotics*, 26(4), 619–634. <https://doi.org/10.1109/TRO.2010.2048610>
- Brandstädt, A., Le, V. B., & Szymczak, T. (1998). The complexity of some problems related to graph 3-colorability. *Discrete Applied Mathematics*, 89(1–3), 59–73. [https://doi.org/10.1016/S0166-218X\(98\)00116-4](https://doi.org/10.1016/S0166-218X(98)00116-4)
- Brewka, G., Eiter, T., & Truszczyński, M. (2011). Answer set programming at a glance. *Communications of the ACM*, 54(12), 92–103. <https://doi.org/10.1145/2043174.2043195>
- Buccafurri, F., Leone, N., & Rullo, P. (1997). Strong and weak constraints in disjunctive datalog. In Dix, J., Furbach, U., & Nerode, A. (Eds.), *Logic Programming and nonmonotonic reasoning, 4th international conference, LPNMR'97, Dagstuhl Castle, Germany, July 28–31, 1997, proceedings. Lecture notes in computer science* (Vol. 1265, pp. 2–17). Springer. https://doi.org/10.1007/3-540-63255-7_2
- Burrows, C., Secoli, R., & Baena, F. R. (2013). Experimental characterisation of a biologically inspired 3D steering needle. In *2013 13th international conference on control, automation and systems (ICCAS 2013)* (pp. 1252–1257). IEEE. <https://doi.org/10.1109/ICCAS.2013.6704141>
- Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., et al. (2020). Asp-core-2 input language format. *Theory and Practice of Logic Programming*, 20(2), 294–309. <https://doi.org/10.1017/S1471068419000450>
- Chen, C. H., Liu, T. K., & Chou, J. H. (2014). A novel crowding genetic algorithm and its applications to manufacturing robots. *IEEE Transactions on Industrial Informatics*, 10(3), 1705–1716. <https://doi.org/10.1109/TII.2014.2316638>
- Crooks, W., Vukasin, G., O'Sullivan, M., Messner, W., & Rogers, C. (2016). Fin ray® effect inspired soft robotic gripper: From the robosoft grand challenge toward optimization. *Frontiers in Robotics and AI*, 3, 70.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271. <https://doi.org/10.1007/BF01386390>
- Eiter, T., Gottlob, G., & Mannila, H. (1994). Adding disjunction to datalog. In Vianu, V. (Ed.), *Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, May 24–26, 1994, Minneapolis, Minnesota, USA* (pp. 267–278). ACM Press. <https://doi.org/10.1145/182591.182639>
- Erdem, E., Aker, E., & Patoglu, V. (2012). Answer set programming for collaborative housekeeping robotics: Representation, reasoning, and execution. *Intelligent Service Robotics*, 5(4), 275–291. <https://doi.org/10.1007/s11370-012-0119-x>
- Erdem, E., Patoglu, V., & Saribatur, Z. G. (2015). Integrating hybrid diagnostic reasoning in plan execution monitoring for cognitive factories with multiple robots. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 2007–2013). IEEE.
- Erdos, D., Erdos, A., & Watkins, S. E. (2013). An experimental UAV system for search and rescue challenge. *IEEE Aerospace and Electronic Systems Magazine*, 28(5), 32–37. <https://doi.org/10.1109/MAES.2013.6516147>
- Favaro, A., Cerri, L., Galvan, S., Baena, F. R. Y., & De Momi, E. (2018). Automatic optimized 3d path planner for steerable catheters with heuristic search and uncertainty tolerance. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 9–16). IEEE. <https://doi.org/10.1109/ICRA.2018.8461262>
- Favaro, A., Secoli, R., Baena, F. R., & De Momi, E. (2020). Model-based robot pose estimation for a multi-segment, programmable bevel-tip steerable needle. *IEEE Robotics and Automation Letters*, 5(4), 6780–6787. <https://doi.org/10.1109/LRA.2020.3018406>
- Favaro, A., Segato, A., Muretti, F., & De Momi, E. (2021). An evolutionary-optimized surgical path planner for a programmable bevel-tip needle. *IEEE Transactions on Robotics*. <https://doi.org/10.1109/TRO.2020.3043692>
- Fischl, B. (2012). *Freesurfer*. *Neuroimage*, 62(2), 774–781. <https://doi.org/10.1016/j.neuroimage.2012.01.021>
- Fu, M., Salzman, O., & Alterovitz, R. (2021). Toward certifiable motion planning for medical steerable needles. arXiv preprint [arXiv:2107.04939](https://doi.org/10.15607/RSS.2021.XVII.081). <https://doi.org/10.15607/RSS.2021.XVII.081>
- Garrido, S., Moreno, L., Abderrahim, M., & Martin, F. (2006). Path planning for mobile robot navigation using Voronoi diagram and fast marching. In *2006 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2376–2381). IEEE. <https://doi.org/10.1109/IROS.2006.282649>
- Gebser, M., Kaminski, R., Kaufmann, B., & Schaub, T. (2019). Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, 19(1), 27–82. <https://doi.org/10.1017/S1471068418000054>

- Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3–4), 365–385. <https://doi.org/10.1007/BF03037169>
- Goldstone, W. (2009). *Unity game development essentials*. Packt Publishing Ltd.
- Gómez, R. N., Hernández, C., & Baier, J. A. (2021). A compact answer set programming encoding of multi-agent pathfinding. *IEEE Access*, 9, 26886–26901. <https://doi.org/10.1109/ACCESS.2021.3053547>
- Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107. <https://doi.org/10.1109/TSSC.1968.300136>
- Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in neural information processing systems* (pp. 4565–4573).
- Hoy, M., Matveev, A. S., & Savkin, A. V. (2015). Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey. *Robotica*, 33(3), 463–497. <https://doi.org/10.1017/S0263574714000289>
- Huang, H., Hoffmann, G. M., Waslander, S. L., & Tomlin, C. J. (2009). Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *2009 IEEE international conference on robotics and automation (ICRA)* (pp. 3277–3282). IEEE. <https://doi.org/10.1109/ROBOT.2009.5152561>
- Jordan, M., & Perez, A. (2013). Optimal bidirectional rapidly-exploring random trees. In *Computer science and artificial intelligence laboratory technical report*.
- Juliani, A., Berges, V.P., Vckay, E., Gao, Y., Henry, H., Mattar, M., & Lange, D. (2018). Unity: A general platform for intelligent agents. arXiv preprint [arXiv:1809.02627](https://arxiv.org/abs/1809.02627)
- Kallem, V., Komoroski, A. T., & Kumar, V. (2011). Sequential composition for navigating a nonholonomic cart in the presence of obstacles. *IEEE Transactions on Robotics*, 27(6), 1152–1159. <https://doi.org/10.1109/TRO.2011.2161159>
- Kanda, T., Shiomi, M., Miyashita, Z., Ishiguro, H., & Hagita, N. (2009). An affective guide robot in a shopping mall. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction (HRI)* (pp. 173–180).
- Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846–894. <https://doi.org/10.1177/0278364911406761>
- Lau, B., Sprunk, C., & Burgard, W. (2009). Kinodynamic motion planning for mobile robots using splines. In *2009 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2427–2433). IEEE. <https://doi.org/10.1109/IROS.2009.5354805>
- LaValle, S. M., & Kuffner Jr, J. J. (2000). *Rapidly-exploring random trees: Progress and prospects*. Citeseer
- Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., & Scarcello, F. (2006). The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3), 499–562. <https://doi.org/10.1145/1149114.1149117>
- Leone, N., & Ricca, F. (2015). Answer set programming: A tour from the basics to advanced development tools and industrial applications. In Faber, W., & Paschke, A. (Eds.), *Reasoning web. Web logic rules: 11th international summer school 2015, Berlin, Germany, July 31–August 4, 2015, tutorial lectures. Lecture notes in computer science* (Vol. 9203, pp. 308–326). Springer. https://doi.org/10.1007/978-3-319-21768-0_10
- Lifschitz, V. (2019). Answer set programming. Springer. <https://doi.org/10.1007/978-3-030-24658-7>
- Lifschitz, V. (2002). Answer set programming and plan generation. *Artificial Intelligence*, 138(1–2), 39–54. [https://doi.org/10.1016/S0004-3702\(02\)00186-8](https://doi.org/10.1016/S0004-3702(02)00186-8)
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint [arXiv:1509.02971](https://arxiv.org/abs/1509.02971)
- Michalski, R. S. (1983). A theory and methodology of inductive learning. In *Machine learning* (pp. 83–134). Elsevier. <https://doi.org/10.1016/B978-0-08-051054-5.50008-X>
- Mirowski, P., Grimes, M., Malinowski, M., Hermann, K. M., Anderson, K., Teplyashin, D., Simonyan, K., Zisserman, A., Hadsell, R. (2018) Learning to navigate in cities without a map. In *Advances in neural information processing systems* (pp. 2419–2430).
- Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., & Kumaran, D. (2016). Learning to navigate in complex environments. arXiv preprint [arXiv:1611.03673](https://arxiv.org/abs/1611.03673)
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Belle-mare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529. <https://doi.org/10.1038/nature14236>
- Nguyen, D. T., Kumar, A., & Lau, H. C. (2017). Policy gradient with value function approximation for collective multiagent planning. In *Advances in neural information processing systems* (pp. 4319–4329).
- O'Brien, D., Fisher, S., & Lin, M. C. (2001). Automatic simplification of particle system dynamics. In *Proceedings computer animation 2001. Fourteenth conference on computer animation (Cat. No. 01TH8596)* (pp. 210–257). IEEE. <https://doi.org/10.1109/CA.2001.982395>
- Obitko, M. (1998). *Genetic algorithms*. Internet publication
- Panov, A. I., Yakovlev, K. S., & Suvorov, R. (2018). Grid path planning with deep reinforcement learning: Preliminary results. *Procedia Computer Science*, 123, 347–353. <https://doi.org/10.1016/j.procs.2018.01.054>
- Pieper, S., Halle, M., & Kikinis, R. (2004). 3D slicer. In *2004 2nd IEEE international symposium on biomedical imaging: Nano to macro (ISBI) (IEEE Cat No. 04EX821)* (pp. 632–635). IEEE. <https://doi.org/10.1109/ISBI.2004.1398617>
- Portillo, J. J., Garcia-Mata, C. L., Márquez-Gutiérrez, P. R., & Baray-Arana, R. (2011). Robot platform motion planning using answer set programming. In *LA-NMR* (pp. 35–44).
- Purdy, G. B., & Smith, J. W. (2010). Lines, circles, planes and spheres. *Discrete & Computational Geometry*, 44(4), 860–882. <https://doi.org/10.1007/s00454-010-9270-3>
- Qu, Z., Wang, J., & Plaisted, C. E. (2004). A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles. *IEEE Transactions on Robotics*, 20(6), 978–993. <https://doi.org/10.1109/TRO.2004.829461>
- Raja, P., & Pugazhenth, S. (2012). Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences*, 7(9), 1314–1320.
- Ram, B. (2009). *Engineering mathematics*. Pearson Education India
- Robert, A., Vijay, K., Brian, W., Jun-ku, Y., David, L., Sanderson, A. C., Zheng, Y. F., & Bekey, G. A. (2008). *Robotics: State of the art and future challenges*. World Scientific
- Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning* (pp. 1889–1897).
- Segato, A., Di Marzo, M., Zucchelli, S., Galvan, S., Secoli, R., & De Momi, E. (2021). Inverse reinforcement learning intra-operative path planning for steerable needle. *IEEE Transactions on Bio-medical Engineering*. <https://doi.org/10.1109/TBME.2021.3133075>
- Segato, A., Di Vece, C., Zucchelli, S., Di Marzo, M., Wendler, T., Azampour, M. F., et al. (2021). Position-based dynamics simulator of brain deformations for path planning and intra-operative control in keyhole neurosurgery. *IEEE Robotics and Automation Letters*, 6(3), 6061–6067.

- Segato, A., Sestini, L., Castellano, A., Baena, & De Momi, E. (2020). GA3C reinforcement learning for surgical steerable catheter path planning. In *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE.
- Segato, A., Valentina, P., Favaro, A., Marco, R., Andrea, F., De Momi, E., & Antonella, C. (2019). Automated steerable path planning for deep brain stimulation safeguarding fiber tracts and deep grey matter nuclei. *Frontiers in Robotics and AI*. <https://doi.org/10.3389/frobt.2019.00070>
- Shi, C., Luo, X., Qi, P., Li, T., Song, S., Najdovski, Z., et al. (2016). Shape sensing techniques for continuum robots in minimally invasive surgery: A survey. *IEEE Transactions on Biomedical Engineering*, *64*(8), 1665–1678. <https://doi.org/10.1109/TBME.2016.2622361>
- Sniedovich, M. (2006). Dijkstra's algorithm revisited: The dynamic programming connexion. *Control and Cybernetics*, *35*, 599–620.
- Tai, L., Paolo, G., & Liu, M. (2017). 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)* (pp. 31–36). IEEE. <https://doi.org/10.1109/IROS.2017.8202134>
- Tovar, B., Murrieta-Cid, R., & LaValle, S. M. (2007). Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, *23*(3), 506–518. <https://doi.org/10.1109/TRO.2007.898962>
- Wang, J., Ohya, T., Liao, H., Sakuma, I., Wang, T., Tohnai, I., & Iwai, T. (2011). Intravascular catheter navigation using path planning and virtual visual feedback for oral cancer treatment. *The International Journal of Medical Robotics and Computer Assisted Surgery*, *7*(2), 214–224. <https://doi.org/10.1002/rcs.392>
- Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, *4*(2), 65–85. <https://doi.org/10.1007/BF00175354>
- Wulfmeier, M., Wang, D. Z., & Posner, I. (2016). Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2089–2095). IEEE. <https://doi.org/10.1109/IROS.2016.7759328>
- Xu, J., Duindam, V., Alterovitz, R., & Goldberg, K. (2008). Motion planning for steerable needles in 3D environments with obstacles using rapidly-exploring random trees and backchaining. In *2008 IEEE international conference on automation science and engineering* (pp. 41–46). IEEE.
- Zhou, J. H., Zhou, J. Q., Zheng, Y. S., & Kong, B. (2016). Research on path planning algorithm of intelligent mowing robot used in large airport lawn. In *2016 international conference on information system and artificial intelligence (ISAI)* (pp. 375–379). IEEE. <https://doi.org/10.1109/ISAI.2016.0086>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Alice Segato was born in 1994, in Monza (MB), Italy. She obtained the Bachelor and Master of Science in Computer Science and Engineering at Politecnico di Milano. Since November 2018, she is a Ph.D. student at the Department of Electronics, Information and Bioengineering (DEIB) of the Politecnico di Milano. Her main research interests include Robotics, Artificial Intelligence and Machine Learning, in the field of Minimal Invasive Neurosurgery.



Francesco Calimeri currently serves as Associate Professor at the Department of Mathematics and Computer Science, University of Calabria, Italy. He received his “Laurea” degree (pretty equivalent to an M.Sc. degree) in “Ingegneria Informatica” (Computer Science Engineering) in 2001, from the University of Calabria, Italy, where he also got a Ph.D. in Mathematics and Computer Science in 2006, after a series of on-contract scientific collaborations with private and public institutions, such as CIES (Centro di Ingegneria Economica e Sociale, Italy) and the Institute of Information Systems (Vienna University of Technology, Vienna, Austria).



Irene Testa was born on february 3rd, 1996, in Milan (MI), Italy. In July 2018 she obtained the Bachelor of Science in Biomedical Engineering at Politecnico di Milano. In December 2020, she obtained the Master of Science in Biomedical Engineering, Technologies for Electronics (BTE) defending the thesis “Automatic Neurosurgical 3D Path Planner Based on Inductive Learning Approach for Robotic Steerable Needles”.



Valentina Corbetta was born in 1996 in Milan (MI), Italy. She got her Bachelor's degree in Biomedical Engineering at Politecnico di Milano in 2018. She is currently attending the last year of her Master's degree in Information Bioengineering at Politecnico di Milano. Her main research interests consist in Artificial Intelligence and Deep Learning applied to computer vision for surgical planning.



Marco Riva is currently a neurosurgeon at IRCCS Istituto Galeazzi. He graduated cum laude in Medicine and Surgery at Università Vita-Salute San Raffaele in 2008 and specialised in Neurosurgery at Università degli Studi di Milano in 2015. He has dedicated over the years to different research lines to study the pathologies of the central nervous system and how to treat them. As a researcher, he is involved in the didactic activity of the School of Specialisation in Neurosurgery at

Università degli Studi di Milano.



Elena De Momi was born in Legnago, Verona, Italy, in 1978. MSc in Biomedical Engineering in 2002, Ph.D. in Bioengineering in 2006, currently Assistant Professor in Electronic Information and Bioengineering Department (DEIB) of Politecnico di Milano. IEEE Senior Member, from 2016 she has been an Associated Editor of IEEE ICRA, IROS and BioRob and she is currently Publication Co-Chair of ICRA 2019. She is currently PI for POLIMI of the EDEN2020 project and of ATLAS MSCA-ITN-2018-EJD. She has been evaluator and reviewer for the European Commission in FP6, FP7 and H2020