Inductive and Deductive Reasoning for Robotic Steerable Needle in Neurosurgery

Alice Segato[1], Valentina Corbetta [1], Francesco Calimeri[2] and Elena De Momi[1]

I. INTRODUCTION

Keyhole approaches in brain surgery require the surgeon to reach targets located in deep brain region. Endpoints can be difficult to reach by traditional rigid needles, without damaging the adjacent tissues. Steerable needles can represent a breakthrough in neurosurgery, as they grant access to sensitive destinations [1]. Computing the optimal trajectory in the preoperative phase consists in a path planning problem, which can be tackled exploiting classical methods, such as graph-based, search-based and learning-based approaches. However these techniques present some limitations: the first two require a trade-off between completeness and efficiency, while the latter needs large datasets to successfully train models. To overcome these drawbacks, we propose to model the path planning problem for steerable needle in neurosurgery combining deductive and inductive reasoning. In particular our system, depicted in figure 1, exploits Answer Set Programming (ASP) semantics to model the brain environment and thus implement an artificial intelligent agent able to move within it, satisfying requirements, which can be customized depending on the specific application and based on the preferences expressed by domain experts, as surgeons and clinicians.

II. MATERIALS AND METHODS

A. Answer Set Programming

Efficient and reliable ASP solver exist, such as DLV/DLV2 [2], [3] and clingo [4] supporting a standard language. If `1,...,tk` are terms (either constant or variable), and `p` is a predicate symbol of arity `k`, then `p(t1,...,tk)` is an atom. A literal `l` is of the form `a` or `not a`, where `a` is an atom; in the former case `l` is positive, otherwise negative. A rule is of the following form:

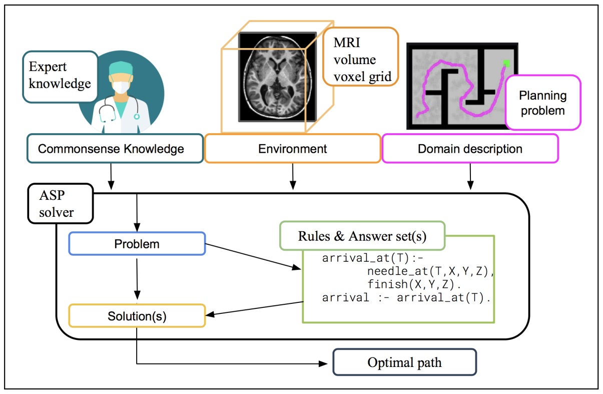
```
a0|...|ak:-b1,...,bn not bn+1,...,bm   (1)
```

The left and right side of the clause are called head and body, respectively; the head is a disjunction of atoms, while the body consists of a conjunction of literals, that can be either positive or negative. A constraint is a rule with empty head; hard ("strong") and soft ("weak") constraints can be specified in order to cut out undesired models and express preferences in case of optimization problems, respectively.

*This project has received funding from the European Union Horizon 2020 research and innovation program under grant agreement No 688279.

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

[2]Department of Mathematics and Computer Science, University of Calabria, Italy



```
arrival_at(T):-
      needle_at(T,X,Y,Z),
      finish(X,Y,Z).
arrival :- arrival_at(T).
```

Fig. 1. Neurosurgical learning system using ASP for Robotic steerable catheter.

Soft constraints are expressed with the symbol $:\sim$ instead of $:-$ as it is the case for hard constraints.

B. The Agent

In our system, the agent is the probe moving in the brain environment. The herein proposed system, depicted in figure 1, makes use of ASP to model the brain environment and implement an artificial intelligent agent able to move within it while meeting some given requirements.

C. Modelling the environment

The environment in which the agent moves is obtained from a 3D MRI image. As first step, we resized it from $256 \times 256 \times 256$ to $64 \times 64 \times 64$; then we transformed it from a gray-scale image into a binary voxel grid, in which obstacles have value 1, while portions of the brain in which the needle can pass through have value 0. Each voxel is described by its X, Y and Z coordinates and can be linked to one or more attributes to incorporate semantics [5]. In the ASP program, each voxel can be represented by the atom `voxel(P,X,Y,Z)`, where P is the value of the voxel and X, Y and Z are the voxel coordinates. The starting and finish point are defined as `start(X1,Y1,Z1)` and `finish(X2,Y2,Z2)`.

D. Moving in the environment

In our first implementation, we chose as "hard constraint" the impossibility of passing through an obstacle, which is intrinsic in the definition of adjacent voxels. The concept of adjacency allowed us also to delineate the possible directions in space of the agent. Therefore, we considered two voxels as adjacent only if they both have value 0, thus only if they are not an obstacle. In Algorithm 1, we express with pseudocode the clause which grants the agent the ability to move forward:

Algorithm 1 Condition for moving forward - pseudocode

```
1: if $(X2 = X1 \ \& \ Y2 = Y1 + 1 \ \& \ Z2 = Z1) \ \& \ (P1 = 0 \ \& \ P2 = 0)$ then
2:     $voxel1 \ adjacent \ to \ voxel2$
3: end if
```

We use a disjunctive rule (representing the so-called "guess" part) for defining the search space, and an integrity constraint (representing the so-called "check" part) for stating that the needle must arrive to the target for the problem to be satisfiable.

```
needle_at(T,X2,Y2,Z2)|
n_needle_at(T,X2,Y2,Z2) :- ...        (2)
```

```
:- not arrival.        (3)
```

Weak constraints are used to express conditions (representing the so-called "optimize" part) that should be satisfied, but can be violated. Optimum solution(s), if exist, are identified, in which weak constraints are satisfied as much as possible. In our case, we used a weak constraint to minimize the number of steps spent by the needle while reaching the target from the start:

```
:~ step(T), needle_at(T,X,Y,Z).        (4)
```

The pseudocode reported in algorithm 2 expresses (line 4) the basic concept of guessing the "next" step: any non-visited voxel that is adjacent to another that was visited at time "T-1" can be eligible for the visit at time "T". In addition, line 7 expresses the success criterion.

Algorithm 2 Arrival - pseudocode

```
1: if $start(X, Y, Z)$ then
2:     $needle\_at(0, start)$
3: end if
4: if $step(T) > 0 \ \& \ needle\_at(T-1, voxel1) \ \& \ adjacent(voxel1, voxel2) \ \& \ not\_visited(voxel2)$ then
5:     $needle\_at(T, voxel2)$ or $not\_needle\_at(T, voxel2)$
6: end if
7: if $needle\_at(T, finish) \ \& \ finish(X, Y, Z)$ then
8:     $arrival$
9: end if
```

E. Classification of trajectories

ASP provides a set of possible solutions classified and evaluated on the basis of different parameters, which can be customized by surgeons on the basis of each specific clinical case, providing a high flexibility.

The ASP solver gives back as output, besides the optimal path in terms of step minimization, other possible paths. Once these paths are obtained, it is possible to classify them on the basis of other parameters, using weak constraints.

F. Experimental Protocol

To evaluate the optimal path found by the ASP planner, we compared it with an implementation of the A* search algorithm, which finds the shortest path possible. We ran both programs on ten different randomly-selected couples of possible start and finish points.

III. Results

The average number of steps of the paths found by A* and ASP are equal (7,4). The computational times required by the ASP solver (2043,01s) are higher than the ones for A*(0.002s); however, this parameter is not critical in the pre-operative phase. Table I displays the minimum distance from the obstacles, the average distance from the obstacles and the curvature chosen by the classification for the answer sets found by ASP for one of the ten evaluated couples of points.

TABLE I
Best trajectory parameters classified by ASP

Min Dist [voxel]	Avg Dist [voxel]	Curvature [mm^{-1}]
1	21	0.36
trajectory n. 1	trajectory n. 1	trajectory n. 49

IV. Discussion and Conclusion

In this work we presented a novel approach to model the problem of path planning in the context of keyhole neurosurgery using inductive and deductive reasoning. Answer Set Programming allows us to model the environment and the steerable needle which moves in it using a set of rules, creating a flexible semantics, which can be customized by surgeons depending on the single patient. Results show the ability of the ASP program to find the minimum number of steps to reach the target and also the possibility to obtain different trajectories and thus optimizing distinct characteristics. As future developments, we would like to integrate our ASP program with a surgical simulator and introduce more specific rules for tumor and deep brain stimulation treatments.

References

[1] A. Segato, V. Pieri, A. Favaro, M. Riva, A. Falini, E. D. Momi, and A. Castellano, "Automated steerable path planning for deep brain stimulation safeguarding fiber tracts and deep gray matter nuclei," *Front. Robotics and AI*, vol. 2019, 2019.

[2] W. T. Adrian, M. Alviano, F. Calimeri, B. Cuteri, C. Dodaro, W. Faber, D. Fuscà, N. Leone, M. Manna, S. Perri *et al.*, "The asp system dlv: advancements and applications," *KI-Künstliche Intelligenz*, vol. 32, no. 2-3, pp. 177–179, 2018.

[3] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello, "The dlv system for knowledge representation and reasoning," *ACM Transactions on Computational Logic (TOCL)*, vol. 7, no. 3, pp. 499–562, 2006.

[4] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub, "Multi-shot asp solving with clingo," *Theory and Practice of Logic Programming*, vol. 19, no. 1, pp. 27–82, 2019.

[5] M. Koopman, "3d path-finding in a voxelised model of an indoor environment," *f*, 2016.