Towards solving path planning in keyhole neurosurgery with Answer Set Programming

Valentina Corbetta
Department of Electronics, Information and Bioengineering
Politecnico di Milano, Italy*
valentina.corbetta@polimi.it

Alice Segato
Department of Electronics, Information and Bioengineering
Politecnico di Milano, Italy
alice.segato@polimi.it

Jessica Zangari
Department of Mathematics and Computer Science
University of Calabria, Italy
zangari@mat.unical.it

Simona Perri
Department of Mathematics and Computer Science
University of Calabria, Italy
perri@mat.unical.it

Francesco Calimeri
Department of Mathematics and Computer Science
University of Calabria, Italy
francesco.calimeri@unical.it

Elena De Momi
Department of Electronics, Information and Bioengineering
Politecnico di Milano, Italy
elena.demomi@polimi.it

Among all procedures, keyhole neurosurgery is one of the most hazardous ones, due to the complexity of the brain environment and the high density of vital structures. Respect of the kinematic constraints of the probe selected to perform the procedure and of the preferences dictated by the surgeon's experience are essential to find the safest path to the surgical target. This work presents and optimisation and classification strategy for neurosurgical interventions. The framework relies on Answer Set Programming to translate the requirements and the expert's knowledge into the objectives of the optimisation procedure. The semantics of Answer Set Programming grants extensive flexibility, as it allows to easily change the requirements to optimise and their priority based on the specific needs of the clinical case.

1 Introduction

Keyhole Neurosurgery (KN) has become the gold standard to perform brain procedures, since it allows access to the brain via a small hole on the skull, minimising trauma to the patient [6]. Paramount to the success of this kind of procedures is accurate and precise planning of the trajectory the probe has to follow to reach the Target Structure (TS) deep inside the brain.

To this end, it is important to identify and comply to precise requirements; some of them are strictly mandatory, while others can be defined as soft constraints. The first ones must be fulfilled and are strictly related to the respect of the kinematic constraints of the selected probe: for the trajectory to be feasible, it should not surpass the maximum curvature K_{max} the needle can perform; moreover, the path cannot be

*We thank Eden2020 project consortium partners for precious advice during the project activities. We thank M. Riva for providing the dataset and the manual trajectories for the CED environment simulation. This work has been published at ICRA 2021.

Submitted to:
ICLP 2021

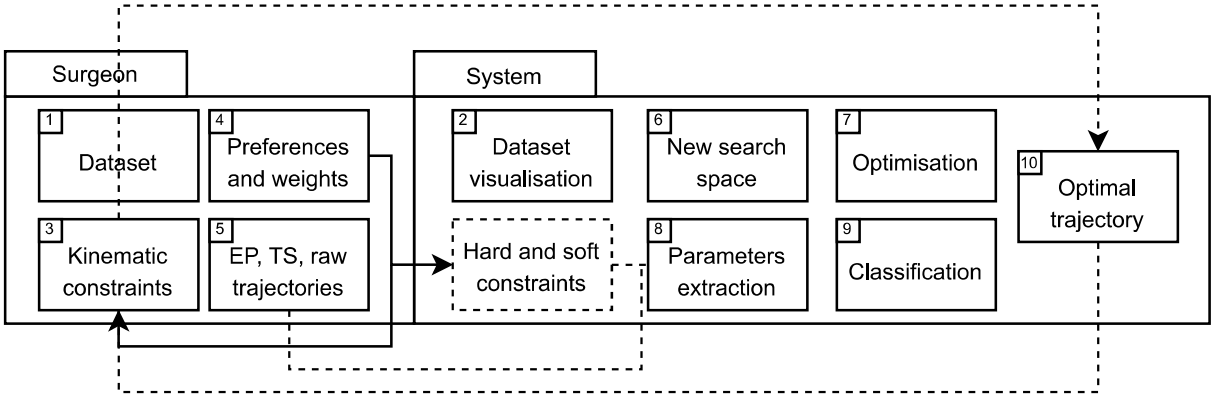

Figure 1: Architecture of the proposed system, displaying the basic workflow and the user interactions.

chosen if it approaches the delicate structures at a minimum distance that is smaller than the radius r of the needle.

Soft constraints are instead dictated by implicit and explicit desiderata that guide the decision-making process of the neurosurgeon. These are used to express preferences, that can vary from procedure to procedure, and even from patient to patient [2]. Generally, they are the following:

- Minimisation of the path length d_{tot} to minimise risks in the motion of the needle.
- Maximisation of the distance from obstacle d_{min} to mitigate the damage of possible mistakes in the motion of the needle.
- Minimisation of the curvature c_{max} performed by the needle to have a smoother path that facilitates the control of the probe.

Given these premises, the attempt of fulfilling these requirements can be seen as an optimisation problem, where soft constraints are the objectives. Classical methods, like graph- and sampling-based techniques [5, 7] or learning-based approaches [9], generate a raw path that is then input of the optimisation phase. The desiderata are translated into numerical constraints via cost functions, that are generally formulated as follows:

$$F(x) = \sum_{i=1}^{N} k_i * f_i(x) \tag{1}$$

where f_i represents the soft constraint weighted of a factor k_i, that can be chosen empirically or suggested by the surgeon.

This standard optimisation approach presents several limitations when applied to KN. Indeed, the transposition of the preferences dictated by the clinician into ad hoc cost functions requires specific mathematical and programming skills together with several hours of coding to implement them. Therefore, it is not possible to immediately implement new requirements. This lack of flexibility and the impossibility of changing them and adapting them real-time during the pre-operative phase hinders their validity as optimisation techniques for KN.

Interestingly, deductive reasoning, and in particular Answer Set Programming (ASP), can provide a valid alternative to overcome these drawbacks. ASP is a declerative logic formalism that encodes computational problems via logic programs [1, 3]. Indeed, its semantics allows to explicitly represent domain

knowledge that can be used to intuitively implement the constraints of the complex brain environment and the surgeon's preferences. Therefore, we herein present an optimisation and classification strategy that exploits the flexible semantics of ASP to find the optimal curvilinear trajectory in the pre-operative phase. The strategy takes in input not only the raw trajectory, but also the priority associated to each optimisation objective that can differ from case to case. Moreover, we show how the optimisation and classification techniques have been integrated in a surgical simulator developed in Unity [4] to intuitively guide the neurosurgeon through all the steps of the decision making process and easily visualise the result of the planning.

2 Materials and methods

2.1 Definition of manual trajectories

To test our methodology, we have decided to use as raw trajectories in input manual trajectories traced by the neurosurgeon. As first step, the neurosurgeon inputs the kinematic constraints of the steerable needle in use, K_{max} and OD, and the parameters he/she intends to prioritise in the selection of the best trajectory, d_{tot}, d_{min} and c_{max} with their respective weights, w_{dtot}, w_{dmin} and w_{cmax}; the higher the weight, the higher the priority. The surgeon then proceeds with the choice of the TS and Entry Point (EP) located on the brain cortex.

Then, starting from EP, the surgeon manually draws a trajectory using a joystick, defined as follows:

$$T(T = t_0, t_1, t_{n-1}) \tag{2}$$

where $t_0 = EP$ and $t_{n-1} = TS$. Each of these trajectories is the input of the optimisation phase, which is followed by the classification step. The final output is the optimal trajectory T^{Opt}.

2.2 Search space definition

Taken as input the manual trajectory T, the planner creates a new search space in which the ASP program can search for new, alternative trajectories. The search space is defined as follows: around each point $t_j \in T$, excluding EP and TS, a set of concentric circles of radius $0 \leq r \leq 5mm$ is defined, to be compliant with the kinematic constraints of the specific case; the range of the radius is specified by the surgeon, depending on the characteristics and design of the specific needle in use.

Then, for each circle, $\overline{t_j}$ points are sampled, equally spaced on the circumference by applying the formulas in Equation 3. Again, the number of $\overline{t_j}$ to be sampled is chosen by the operator, based on the needed resolution and also on the computational power available.

Given the coordinates x_{t_j}, y_{t_j} and z_{t_j} of the centre of the circle, given $\overline{\theta_{t_j}}$ values of angles uniformly distributed between 0° and 360° and the value of the radius r of the desired circle, the formulas in Equation 3 are applied to convert the coordinates from polar to cartesian convention. Thus, x, y and z coordinates for each point $\overline{t_j}$ on the circumference of the circle are obtained.

$$\begin{aligned} \overline{x_{t_j}} &= r \times cos(\overline{\theta_{t_j}}) + x_{t_j} \\ \overline{y_{t_j}} &= r \times sin(\overline{\theta_{t_j}}) + y_{t_j} \\ \overline{z_{t_j}} &= z_{t_j} \end{aligned} \tag{3}$$

From the newly generated set of available points $\overline{t_j}$ a search graph is obtained, where each point is a node of the graph. Considering point t_j in the input path T, each point $\overline{t_j}$, in the concentric circles around it, is connected by an edge to the corresponding point $\overline{t_{j+1}}$, on the concentric circle having the same radius, built around point t_{j+1}; $\overline{t_j}$ is also connected by an edge with the immediately adjacent points to $\overline{t_{j+1}}$. The EP $\overline{t_1}$ and the TS $\overline{t_{n-1}}$ are the same of the original trajectory T.

2.3 ASP trajectory optimisation

The search graph is encoded by facts of form `edge(X,Y)` expressing that there is an edge between points `X` and `Y`. Moreover, facts `start(X)` and `finish(X)` encode EP and TS, respectively. Eventually, possible steps are represented as facts of the form `step(X)`. These facts are the input to an ASP program constituted by the logic rules described next. First, it is stated that at the step 0 the needle must be placed in the start position EP. Then, the program guesses for each step $S > 0$ different from TS the points the needle should visit in order to obtain a possible path. More in detail, the needle can move from point `P1` at step `S-1` to point `P2` at step `S` only if the needle visited `P1` at step `S-1`, `P1` and `P2` are connected by an edge and the target structure TS has not been reached yet. Moreover, the program ensures that the needle cannot visit two or more points at the same time and that it cannot move to positions already visited. This is expressed by the following choice rule:

```
0 <= {needle_at(S, P2) :  edge(P1, P2),needle_at(S-1, P1),
not finish(P2)} <= 1 :- step(S).
```
(4)

Lastly, ASP checks that guessed paths reach the target structure TS and discards paths that do not satisfy such condition via an integrity constraint:

```
finished :- needle_at(S, P),step(S), finish(P).
```
(5)

```
:- not finished.
```
(6)

2.4 Parameters extraction

For each path $\overline{T}$ found by the optimisation in the optimisation phase and for each of the manual trajectories the parameters d_{tot}, d_{min} and c_{max} are computed. These parameters are paramount for the final step of the presented workflow, that is the classification of the trajectories to select the optimal one.

2.5 ASP trajectory classification

In the classification step of this methodology, the best trajectory among the ones provided by the clinician and the ones generated in the optimisation step is chosen.

The ASP program guesses a path to choose among all trajectories, that are encoded by facts of form `trajectory(X)`. The guessed path is then checked against the requirements introduced in Section 1.

We now show how these requirements are translated in hard and soft constraints.

First we report the hard constraints:

```
:- choose(X), curvature(c_max),
maxCurve(X, K_max), K_max < c_max.
```
(7)

$$\texttt{:- choose(X), radius(}r\texttt{),}\\ \texttt{distObst(X, }d_{min}\texttt{), }d_{min}\texttt{ < }r\texttt{.} \tag{8}$$

Essentially, the facts maxCurve(X, c_{max}) and distObst(X, d_{min}) respectively couple c_{max} and d_{min} to the corresponding trajectory X.

Soft constraints have been encoded as follows:

$$\texttt{\#min\{}d_{tot}\texttt{@}w_{d_{tot}}\texttt{,X: choose(X),length(X,}d_{tot}\texttt{)\}.} \tag{9}$$

$$\texttt{\#max\{}d_{min}\texttt{@}w_{d_{min}}\texttt{,X: choose(X),}\\ \texttt{distObst(X,}d_{min}\texttt{)\}.} \tag{10}$$

$$\texttt{\#min\{}c_{max}\texttt{@}w_{c_{max}}\texttt{,X: choose(X),}\\ \texttt{maxCurve(X, }c_{max}\texttt{)\}.} \tag{11}$$

where the facts length(X, d_{tot}), distObst(X, d_{min}) and maxCurve(X, c_{max}) respectively couple d_{tot}, d_{min} and c_{max} to the corresponding trajectory X.

2.6 Optimal trajectory evaluation

The surgeon can now visualise the optimal trajectory T^{Opt} in the simulator. Therefore, he/she can visually evaluate the result. If not completely satisfied, he/she can easily go back and select new optimisation parameters and/or change their respective weigths and obtain immediately a new trajectory.

3 Experimental evaluation and conclusions

The surgical procedure considered as case study is Convection Enhanced Delivery (CED), in which the target is the tumor that is surrounded by any essential structures, mainly vessels. 3-D brain structures were identified on two datasets: 1) Time-of-Flight (ToF) Magnetic Resonance Imaging (MRI) 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 and 3D Slicer. The selected probe is the bio-inspired, multi-segment programmable bevel-tip needle developed in the EDEN2020 project [8].

3.1 Experimental protocol

In order to evaluate the optimal path obtained with our framework, we compared it with the manual trajectories computed by the expert surgeon. To evaluate the optimal path found by the ASP planner, we compared it with the manual trajectories computed by the expert surgeon. The expert surgeon (age: 37, performed surgical procedures: 2440) was asked to select 10 entry points, EP_i, on the brain cortex, a target structure TS and the weights, $w_{c_{max}}, w_{d_{min}}, w_{d_{tot}}$ for the rules prioritization. k experiments, EXP_k (with

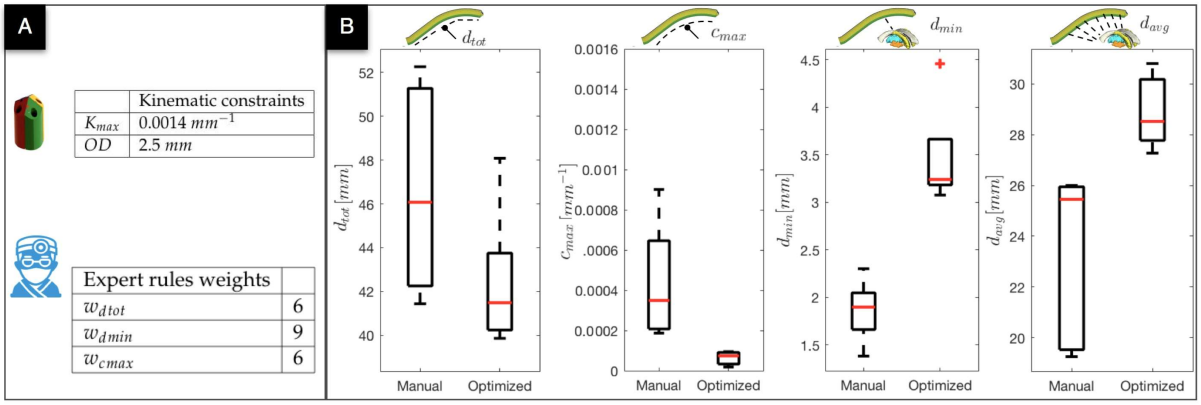

Figure 2: **A** Shows expert's rules, and catheter kinematic constraints. **B** Shows a comparison between Manual and Optimized approach for the CED procedure. The results for both considered scenarios and used approaches are reported in terms of the minimum (d_{min}) and the mean (d_{avg}) distance 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.

$1 \leq k \leq 5$), were conducted. For each EXP_k, the surgeon was asked to generate a pool of trajectories, $\{T_j\}$ (with $1 \leq j \leq 10$), and choose the optimal one, T^{Manual}, based on his expertise. The same pool of surgical paths generated in the manual approach,$\{T_j\}$, was used for the optimization procedure and the optimal one, $\{T^{Opt}\}$, was selected, using rules, weights and kinematic constraints given in input by the surgeon as reported in Figure 2A.

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. The performance metrics of the paths were analysed using Matlab (The MathWorks, Natick, Massachussetts, R2020a). To assess data normality, Lilliefors test is applied. Since the data distribution resulted to be non-normal, we performed pairwise comparison with the Wilcoxon matched-pairs signed-rank test. Differences were considered statistically significant at p-value < 0.05.

3.2 Results

Figure 2B shows a comparison between the Manual and Optimized approaches in terms of d_{min}, d_{avg}, c_{max} and d_{tot} calculated over the best path of left hemisphere. The Optimized approach keeps lower d_{tot} and significantly lower c_{max} (p-value < 0.01) of the path than the Manual approach. While it keeps a significantly greater d_{min} (p-value < 0.01) and d_{avg} (p-value < 0.01) following the rules dictated by the expert who gives more importance to these two parameters in this case.

3.3 Conclusion and future works

The herein proposed methodology consists of the application of deductive reasoning via ASP to support the neurosurgeon in the development of the surgical workflow to increase the accuracy and the outcome of the operation. This methodology allows one to describe the complex environment of the brain and

the steerable needle that moves in it in a purely declarative fashion; using a set of logic rules, a flexible approach is implemented that can be customised by the clinician depending on the selected application and on the specific patient. This logic-based tool optimises the trajectories traced by the neurosurgeon and then selects the best one.

Using ASP for modelling the expert's knowledge and his/her desiderata can be a significant asset for many reasons. First of all, even if the whole path planning tool is implemented in Unity, this is just to provide a user-friendly graphical interface, but under the hood, the knowledge is expressed through a declarative formalism: modifying and adapting the criteria for optimising path and rejecting unwanted ones would be rather easy. Moreover, the specifications are formally encoded, yet executable; and once the optimisation rules have been properly selected, ASP guarantees that the best trajectory is chosen. If two or more paths have the same "score", selecting one or the other would make no difference. Indeed, ASP guarantees the flexibility necessary in KN: the neurosurgeon can easily and in real time change the optimisation objectives and their priority, so that the planning is easily customisable depending on the procedure and on the specific patient.

Results show the effectiveness and robustness of our approach. As future works are concerned, we would like to test our methodology in the Eden2020 framework and also try to apply our optimisation and classification strategy to raw trajectories generated by other path planning approaches, like learning-based methods, to prove the generalisability of our methodology.

References

[1] Gerhard Brewka, Thomas Eiter & Mirosław Truszczyński (2011): *Answer set programming at a glance*. *Communications of the ACM* 54(12), pp. 92–103, doi:10.1145/2043174.2043195.

[2] Caroline Essert, Claire Haegelen, Florent Lalys, Alexandre Abadie & Pierre Jannin (2012): *Automatic computation of electrode trajectories for deep brain stimulation: a hybrid symbolic and numerical approach*. *International journal of computer assisted radiology and surgery* 7(4), pp. 517–532, doi:10.1007/s11548-011-0651-8.

[3] Michael Gelfond & Vladimir Lifschitz (1991): *Classical negation in logic programs and disjunctive databases*. *New generation computing* 9(3-4), pp. 365–385, doi:10.1007/BF03037169.

[4] Will Goldstone (2009): *Unity game development essentials*. Packt Publishing Ltd.

[5] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz & Sebastian Thrun (2005): *Anytime Dynamic A*: An Anytime, Replanning Algorithm*. In: *ICAPS*, pp. 262–271. Available at `https://www.aaai.org/Papers/ICAPS/2005/ICAPS05-027.pdf`.

[6] Tao Liu, Yonghang Tai, Chengming Zhao, Lei Wei, Jun Zhang, Junjun Pan & Junsheng Shi (2020): *Augmented reality in neurosurgical navigation: A survey*. *The International Journal of Medical Robotics and Computer Assisted Surgery*, p. e2160, doi:10.1002/rcs.2160.

[7] Sachin Patil & Ron Alterovitz (2010): *Interactive motion planning for steerable needles in 3D environments with obstacles*. *2010 3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2010*, pp. 893–899, doi:10.1109/BIOROB.2010.5625965.

[8] Riccardo Secoli & Ferdinando Rodriguez y Baena (2016): *Adaptive path-following control for bio-inspired steerable needles*. In: *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, IEEE, pp. 87–93, doi:10.1109/BIOROB.2016.7523603. Available at `http://ieeexplore.ieee.org/document/7523603/`.

[9] Alice Segato, Luca Sestini, Antonella Castellano & Elena De Momi (2020): *GA3C Reinforcement Learning for Surgical Steerable Catheter Path Planning*. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 2429–2435, doi:10.1109/ICRA40945.2020.9196954.