Automated planning for robotic guidewire navigation in the coronary arteries

Pierre Schegg, Jérémie Dequidt, Eulalie Coevoet, Edouard Leurent, Rémi Sabatier, Philippe Preux, Christian Duriez

Automated planning for robotic guidewire navigation in the coronary arteries

Pierre Schegg[a,b], Jérémie Dequidt[a], Eulalie Coevoet[a], Edouard Leurent[a], Rémi Sabatier[c], Philippe Preux[a] and Christian Duriez[a]

***Abstract*— Soft continuum robots, and comparable instruments allow to perform some surgical procedures non-invasively. While safer, less morbid and more cost-effective, these medical interventions increase the complexity for the practitioners: the manipulation of anatomical structures is indirect through telescopic and flexible devices and the visual feedback is indirect through monitors. Interventional cardiology is an example of complex procedures where catheters and guidewires are manipulated to reach and treat remote areas of the vascular network. Such interventions may be assisted with a robot that will operate the tools but the planning (choice of tools and trajectories) remains a complex task. In this paper we use a simulation framework for flexible devices inside the vasculature and we propose a method to automatically control these devices to reach specific locations. Experiments performed on 15 patient geometries exhibit good performance. Automatic manipulation reaches the goal in more than 90% of the cases.**

Keywords: Catheter Navigation, Quadratic Programming, Monte Carlo Tree Search.

I. Introduction

Robotic guidewire navigation within arteries is a planning problem of underactuated soft continuum robots. The movements (translation or rotation) are imposed on the proximal base of the guide, outside the patient. This allows to navigate the distal part of the guidewire which is very soft and deforms in the vessels by contacting the walls. The physician (with or without robot) seeks to impose a movement on the distal end and must therefore anticipate the deformation of the guidewire in order to move forward efficiently. In this paper, we propose a new approach for the planning of guidewire navigation in coronary arteries with a robot, in order to offer guidance assistance to physicians.

The targeted cardiovascular diseases are pathologies in which the coronary arteries get narrower (stenosis), thus impairing the blood flow to certain areas of the heart, which may lead to myocardial infarction. They are one of the leading causes of death worldwide, particularly in Europe and the United States [1], [2]. Percutaneous Coronary Interventions (PCI) are minimally invasive surgery procedures used to treat those coronary artery diseases. PCI consists in inserting one or several concentric guidewires and catheters and navigating the arteries in order to inflate a stent inside the lesion, thus restoring a normal blood flow.

The guidewires are pre-curved wires and have an elastic behavior. Contrary to the instruments used in other cardiology procedures (such as electrophysiology), the guidewires used in PCI are not steerable. This means that the shape of the guidewire cannot be actively modified during the procedure. The guidewire can only be either pushed, pulled or rotated along its longitudinal axis. As a consequence, navigating the guidewire in the coronaries relies heavily on the contacts between the instruments and the artery walls, particularly to pass through bifurcations.

Several robotic platforms have been developed for assisting these interventions [3], [4]. Robotization aims to provide the physician with more comfort in performing the procedure, thus reducing the risk for orthopedic injuries [5], and to reduce their exposure to X-rays [6]. Several clinical trials and physicians also report that the use of the robot allows for a greater precision and facilitates some complex procedures [7], [8]. These robots are all used as teleoperation tools and the physician directly controls the movement of the instruments at their base.

In this paper, we focus on the design of a planning module responsible for the generation of feasible trajectories taking into account the highly underactuated aspect of the system and the contacts with the patient's vessels. The long-term goal is to integrate this planner into a larger closed-loop control system to fully automate guidewire navigation. Several semi-automatic and automatic functionalities can be derived from this planning module while developing fully automatic navigation. For instance, the planning module can be used to generate feasible trajectories and can also test reachability with various instruments and select the most appropriate. Both of these information are valuable to less experienced cardiologists. The planning functions could also lead to assistance functions during the procedure, such as compensation for heart movement or safety functions.

Where other recent works propose new steerable [9] or magnetic [10] tools, we designed our algorithm to consider the instruments already used in angioplasty. This should allow to implement them faster in a clinical setting as it doesn't require a change in the tools clinicians use or specific operating rooms.

Several works in the literature have looked at autonomous control of guidewires and catheters. Those include model-based control of steerable catheters in the aorta [11], reinforcement learning based control of guidewires in 2 dimensional vessels [12]–[14], guidewire control based on a Finite State Machine in 2 dimensional vessels [15] and Jacobian based control of ablation catheters [16]. Most of these previous works use steerable catheters in much larger vessels [11], [12], [16] which means the distal part of the instrument can be deformed and steered rather than relying on contacts. Other articles navigate a guidewire in simplified

[a] Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France [b] Robocath, Rouen, France [c] Caen University Hospital, Caen, France pierre.schegg@inria.fr

2 dimensional vessels and train or test their agent on one geometry [13]–[15]. None of these articles have shown the capacity to navigate in several 3-dimensional patient geometries.

Contributions of the work

Our approach is based on a Finite Element Model (FEM) of the guidewire behavior combined with Signorini's law for the contacts with the vessels. The simulator is patient-specific and the scenarios were tested on real 3D patient data. This simulator is coupled with an inverse model obtained by convex optimization extended to complementarity constraints, which often provides good solutions but falls into local minima. Finally, we combine this with tree search techniques to overcome these minima and allow to find a navigation solution even in complex cases. In summary, the contributions are:

1) A framing of a cardiology procedure (PCI) as a sequential decision making problem using a 3-dimensional medical simulator based on FEM.
2) A new method using parallelized optimistic planning to automatically control and steer catheters and guidewires in the coronaries using contacts.
3) A new method for combining direct control and inverse control within optimistic tree search based planning.

This paper focuses on the decision making process which enables the navigation in the coronaries. The physical model of the simulation was validated in previous works [17]–[19].

II. Background

The contributions presented in this paper were made possible by a novel combination of three approaches: the training simulation of interventional cardiology, a contact deformable model inversion technique by optimization, and a tree search approach for planning. Each of these approaches has its own state of the art. We summarize in this section what is needed to understand the remaining of the paper.

A. Physics based training simulation

Many reports [20] have demonstrated the advantages of computer-based simulation over the conventional training method of apprenticeship in medicine. Therefore, new training simulation systems for various surgeries or procedures have been developed. Typically, these simulators offer a combination of simulation software that aims at reproducing some parts of the surgery procedure and hardware devices that allow the practitioners for realistic interaction with the simulation. In the context of interventional cardiology, several virtual-reality simulators have been proposed to train and learn cardio-vascular procedures [21], [22]. The scientific challenge remains in the simulation of the navigation of guidewires and catheters when many computational models exist [23]. The use of physics-based simulation can also applied to robot-assisted cardiovascular interventions [24].

We use the Simulation Open Framework Architecture (SOFA) [25] to simulate the coronary arteries and their interaction with the instruments. SOFA is an open source physics-based simulation framework initially developed for interactive medical simulations, and extended via plugins to a range of applications including soft robotics. In this paper we use two plugins: for the modeling of guidewires and catheters [19] (section III-A) and for inverse model control of the instruments [26] (section IV-A). We also used the standard OpenAI Gym [27] interface and SofaGym [28] to link the planning algorithms to the simulations.

B. Quadratic Programming with Complementarity Constraints for planning

In our modeling, the configuration of the guidewire is represented by a position vector $\mathbf{x} \in \mathbf{SO(3)}$. The quasi-static equilibrium of the guidewire inside the blood vessels is driven by the internal elastic forces $\mathbf{f}(\mathbf{x})$ that are computed using Timoshenko–Ehrenfest beam elements, the constant external forces such as gravity $\mathbf{f}_{\text{ext}}$, the forces exerted by the robot actuators at the base (proximal extremity) of the guidewire $\mathbf{H}_a^\top \boldsymbol{\lambda}_a$ and the contact forces with the arterial walls $\mathbf{H}_c^\top \boldsymbol{\lambda}_c$. For these last two forces, we use Lagrange multipliers because the intensity of the forces is part of the unknowns of the computation (whereas their directions $\mathbf{H}^\top$ can be considered as known on a given configuration $\mathbf{x}$).

$$\mathbf{f}_{\text{ext}} - \mathbf{f}(\mathbf{x}) + \mathbf{H}_a^\top(\mathbf{x})\boldsymbol{\lambda}_a + \mathbf{H}_c^\top(\mathbf{x})\boldsymbol{\lambda}_c = \mathbf{0} \tag{1}$$

When one of these forces is modified, the static equilibrium is altered and new position and new Lagrange multiplier values need to be found to restore the equilibrium. To do this, we linearize around the current position:

$$\underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{x}}}_{\mathbf{K}(\mathbf{x})} d\mathbf{x} = \mathbf{f}_{\text{ext}} - \mathbf{f}(\mathbf{x}) + \mathbf{H}_a^\top(\boldsymbol{\lambda}_a + d\boldsymbol{\lambda}_a) + \mathbf{H}_c^\top(\boldsymbol{\lambda}_c + d\boldsymbol{\lambda}_c) \tag{2}$$

Additionally, we use a specific solver to compute the values of the Lagrange multipliers. This computation is based on a minimization of the distance $\boldsymbol{\delta}_e(\mathbf{x} + d\mathbf{x})$ between the tip position and a desired trajectory.

$$\min_{d\boldsymbol{\lambda}_a, d\boldsymbol{\lambda}_c} \|\boldsymbol{\delta}_e(\mathbf{x} + d\mathbf{x})\|^2 \tag{3}$$

Solving this minimization while relying on the FEM models provides a way to do realistic planning of the guidewire trajectory [29]. Thanks to the linearization in equation (2) and the linearization of this distance $\boldsymbol{\delta}_e(\mathbf{x} + d\mathbf{x}) = \boldsymbol{\delta}_e(\mathbf{x}) + \mathbf{H}_e d\mathbf{x}$, the problem, at each iteration, can be brought to quadratic programming. However, the modeling of contacts is particular and requires the use of complementarity constraints between the contact distance $\boldsymbol{\delta}_c(\mathbf{x} + d\mathbf{x})$ and the contact force $\boldsymbol{\lambda}_c + d\boldsymbol{\lambda}_c$ (Signorini's law):

$$0 \leq \boldsymbol{\delta}_c(\mathbf{x} + d\mathbf{x}) \perp \boldsymbol{\lambda}_c + d\boldsymbol{\lambda}_c \geq 0 \tag{4}$$

Consequently, a non-convex QPCC algorithm (Quadratic Programming with Complementarity Constraints) can be used [29], [30] but as the problem is inherently not convex, the solution found by the algorithm can fall into local minima. In our application, this happens particularly in complex bifurcations where the guidewire becomes "stuck".

In such case, one should first move away from the objective to get out of a contact before trying to minimize the distance to the objective again. Yet, the algorithm QPCC cannot "decide" to temporarily move away from the objective, since it minimizes it at each iteration. Thus we need to rely on other strategies to get out of those local minima.

C. Tree Search based Planning

In the following, we model the problem of navigating in the coronaries as a Markov Decision Process (MDP), a general framework for sequential decision-making in optimal control. An MDP is a tuple $\{S, A, T, R\}$ where:

- S is the state space. The state contains the information that is available about the environment and the agent at a given moment.
- A is the action space which defines the interactions between the agent and the environment.
- $T : S \times A \times S \rightarrow [0, 1]$ is the transition function. $T(s, a, s')$ represents the probability to be in state s' after the agent performed action a in state s.
- $R : S \times A \rightarrow \mathbb{R}$ is the reward function, and indicates whether the agent is making progress towards the goal.

Planning algorithms aim to create a plan, that is to find a sequence of actions, which maximizes the cumulative reward received along the trajectory. Since the plan must be constructed before the sequence of actions is carried out, a model of the environment must be available. In this paper, we use a simulator as a model of reality.

A popular example of tree search based planning algorithm is Monte Carlo Tree Search [31]. Many other tree search algorithms have been developed, often stemming from the Multi-Armed Bandit community. Some examples include Upper Confidence bound for Trees (UCT) [32], Open Loop Optimistic Planning (OLOP) [33], [34] and Optimistic Planning for Deterministic systems (OPD) [35].

In this article we use OPD. The goal of this algorithm is, starting from any state of the system, to select a near-optimal action by planning, while respecting a pre-defined budget. OPD builds a look-ahead tree rooted at the current state s_0. At each node s_t, the branches represent the possible actions a_i that the agent can choose from state s_t and the children nodes are the states s_{t+1,a_i} that the system can reach after performing action a_i in state s_t. The look-ahead tree is constructed by iteratively expanding a leaf node and simulating the corresponding transitions. An example of a tree after two such iterations is shown in Figure 1. At each iteration, OPD selects the most promising leaf to expand by computing an upper-bound on the cumulative reward that can be obtained by following the corresponding trajectory. OPD will grow its look-ahead tree for a predefined number of planning iterations and then return the action with the highest observed cumulative reward.

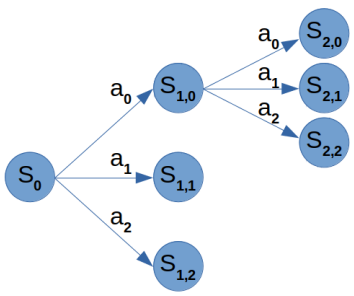

Fig. 1. **Example of a tree which has been expanded twice. In this example there are 3 actions.**

III. Materials and Methods

A. Modeling guidewires and catheters

Guidewires and catheters are wire-like structures. We model them using serially linked elements using Timoshenko–Ehrenfest beam theory which allows to simulate their dynamical behavior [17]–[19]. The guidewires are pre-curved, meaning they are straight and their tip is curved with a specific radius. The curvature is chosen as constant and equal to 2.5mm. Guidewires are made of nitinol and have a Young's modulus of 75GPa. Both of these parameters can easily be changed prior to running the algorithms presented in this article as discussed in section V-A.

B. Patient based coronary arteries

Given the large anatomical variety between patients, we create generic simulations allowing to easily test various vessel geometries. We tested our framework with 15 different patient geometries (6 Right Coronary Arteries and 9 Left Coronary Arteries) from 2 different databases [36], [37]. The dataset is composed of both healthy and anomalous topologies and geometries of coronary arteries, offering a large variety of test cases. There are a total of 70 branches within these 15 patient geometries. This is a preliminary study. In future work, the algorithms presented should be tested on a larger dataset for more thorough validation.

The arteries are represented by rigid surfacic meshes. The vessel deformation due to the heart beat, breathing and interaction with surrounding organs is left as future work. Blood flow is not simulated because realistic and accurate 3D models [38] are computationally intensive.

In order to compare different automatic navigation algorithms, we designed a benchmark comprising 70 different experiments. An experiment corresponds to navigating automatically from the ostium of the coronaries to the farthest point of one branch, as shown in the accompanying video. To simplify the setting we always use the same guidewire shape. This is a limitation as in a real setting a cardiologist would shape the tip of the guidewire differently for each trajectory. Some trajectories are therefore completely infeasible (even manually) with the chosen guidewire, which reduces our final success rate. Overcoming this limitation is discussed in section V-A.

When controlling the guidewire by hand, we can only successfully complete 66 of the 70 trajectories (94% success rate). In the unsuccessful cases the distal tortuosity of some vessels and the large angulation of the bifurcation makes the navigation impossible with the default instrument. In

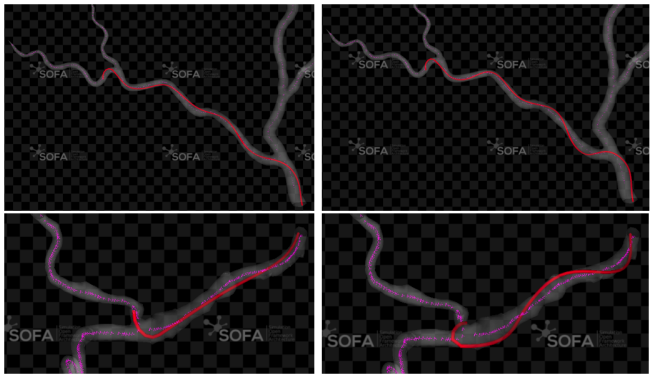

Fig. 2. **Two impossible trajectories. On top the distal tortuosity of the vessel is very high. Trying to push the base of the guidewire (top right) doesn't move the distal part. On the bottom the anglulation of the bifurcation is too large, trying to push (bottom right) or rotate results in the distal part of the guidewire being pulled out of the target branch and pushed into the other.**

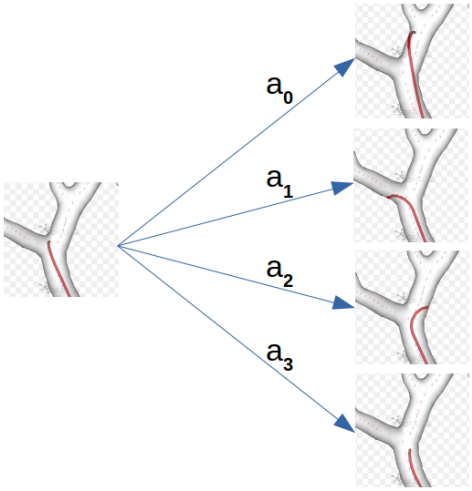

Fig. 3. **Starting from some state of the physical system (left), the agent can choose between 4 discrete actions: move forward (a_0), turn clockwize (a_1), turn counter clockwize (a_2) and move backward (a_3), leading to 4 new states of the system.**

a clinical setting, this would be solved by using either a different guidewire shape or a combination of concentric instruments as discussed in section V-B. In this experiment, this means that the maximum expected success rate for any algorithm is 94%.

C. Center line extraction and reward engineering

Using the SOFA Skeleton Plugin [39] and the CGAL library [40], we extract the center line of the vessels. The center lines form a graph, we can thus use Dijkstra's algorithm [41] to find a sequence of waypoints connecting the starting point (the coronary ostium) and a goal point (the lesion to cure). This list of waypoints is the shortest path within the arteries considered and is used as a trajectory which we aim to follow with Quadratic Programming in section IV-A. It is also used to engineer a reward for the planning algorithms in sections IV-B and IV-C

D. PCI as a sequential decision making problem

We model the problem of navigating guidewires as a Markov Decision Process. Specifically we define:

1) The space S is a vector which contains the position of the goal to reach, of the points of the instruments and of the centerline of the blood vessels.
2) The action space A is discrete and has 4 actions: move forward, move backwards, rotate clockwise and rotate counter-clockwise.
3) The reward function $R(s, a)$ is 1 if the instrument is moving forward along the defined path and 0 otherwise.
4) The Transition function $T : S \times A \times S \rightarrow [0, 1]$ is the simulation of the system when actions are applied

IV. Automatic Guidewire Navigation

A. Using inverse control to navigate

1) Principle: We use the sequence of waypoints defined in section III-C and inverse model based control in the form of quadratic programming with contacts to follow this path. There is one effector which is the tip of the guidewire, and two actuators, controlling the translation and rotation along the guidewire's longitudinal axis. We initialize the target position at the closest way point from the ostium. The QP computes the actuation which allows to minimize the distance between the effector and the target position. Each time this distance is smaller than a predefined threshold we change the target position to the next way point.

2) Results: On our benchmark, this algorithm achieves a success rate of 20% (14 successful trajectories out of 70) and achieves more than 80% of the trajectory in 29% of the test cases (21 out of 70) as shown in Fig. 5.

By examining the videos of the experiments, it is obvious that this algorithm often gets stuck in local minima. In many cases, the guidewire needs to momentarily get farther from the target in order to progress in the longer run and one-step optimization is not able to achieve this behavior. Though the algorithm often gets stuck, most of the time it does make significant progress in the navigation. The main advantage of this algorithm is that it runs very fast. On the benchmark, the longest computation time for a trajectory was 174 seconds and the median was 66 seconds.

B. Using tree search based planning to navigate

1) Principle: Related works often use model-free reinforcement learning to tackle the autonomous navigation problem [13], [14]. On the contrary, we use FEM simulation as a generative model and estimation oracle. Given the access to an oracle and by defining a discrete action space, we can use sampling-based estimation methods such as Monte Carlo Tree Search [31] or other tree search based planning algorithms [32]–[35]. Contrary to Learning approaches, this method does not require any prior training, and will not suffer from generalization problems. It will however require more computation at run time and probably will run slower.

In this type of planning algorithms, a look-ahead tree rooted at the current state is gradually expanded through a fixed number of calls to the generative model. We refer to this predetermined number of calls to the simulator as *planning iterations* and having a fixed number of planning iterations guarantees that the planner converges in finite time. After the tree has been extended within this number of iterations, the planner will recommend a sequence of actions (a plan) which will maximize the reward. The agent can then take one action (receding horizon value of 1) or a sequence of actions (receding horizon value larger than 1). After these steps, either the goal is reached, or a new tree rooted at the current state is created and the planning sequence starts again (we create a new plan, using the same number of planning iterations).

There exists a large variety of tree search based planning algorithms, which differ in the way they expand the look ahead tree. Since our simulator is deterministic we used a parallel version of the OPD algorithm [35] with a receding horizon. OPD was designed for deterministic systems which translates to a faster convergence than algorithms that handle stochastic systems [34].

The efficiency of planning algorithms is often defined in terms of the numbers of calls to the simulator (sample efficiency), which means that nodes have to be expanded *sequentially* to always make the best use of all past information. In contrast, in this work we are interested in time efficiency, we can thus leverage *parallelism* to obtain additional samples in the same compute time, through additional computational power. This is why we select and expand several leaves in parallel. This maintains the optimality of the traditional OPD exploration, but also allows some extra exploration in parallel. For the same reason we use a receding horizon larger than 1, allowing the agent to take several steps without replanning under certain conditions.

2) Receding horizon: In this experiment, the best time performance is achieved when using a receding horizon larger than 1, meaning that for one planning step (one expansion of the tree for the predefined number of planning iterations), the agent is allowed to pick a sequence of actions (instead of picking only one action) before replanning. In our case, as long as the root of the tree has a value larger than 1.01 this means there is at least 2 nodes with non zero rewards in the subtree, therefore we allow the agent to act without replanning.

Without this feature, the agent needs to replan everytime it picks an action. Since planning is the most expensive step in terms of time, reducing the number of times the agent needs to plan divides the median run time of the benchmark by more than 4.

3) Parallelizing the tree expansion: In the standard OPD process, the tree growth is done by choosing one leaf with the highest value and expanding all of its children. While this is optimal in terms of the number of calls to the simulator, it is not necessarily optimal in terms of computation time. Indeed if one has more computation resources, one can choose the n leaves with the highest values and expand all of their children. A lot of this computation will probably not be used for the final trajectory, but since it is executed in parallel, it still allows to converge faster.

In this section we first allowed a very high number of planning iterations to the sequential version of OPD in order to find the maximum success rate. We then decrease the number of iterations to make the algorithm run faster, while maintaining this maximum success rate.

Parallelizing the tree expansion allows to reduce the number of planning iterations needed to find the best trajectory. Indeed, if two leaves are expanded at each planning step for instance, the algorithm only needs half the planning iterations to have the same number of nodes in the tree. Thus, for each number of leaves expanded in parallel, we search for the minimum number of planning iterations that allows to maintain the maximum success rate. As shown in Figure 4, expanding more leaves in parallel and reducing the number of planning iterations allows the algorithm to run faster while maintaining the success rate.

However, planning is still an inherently sequential task and the depth of the tree can never be higher than the number of planning iterations. This is because in order to expand any node, the algorithm needs to have expanded its parent at the previous planning step. This property is shown in Figure 4. Indeed no matter the amount of parallelization, the minimum time required to solve a trajectory is always around 30 seconds. These are the cases when the planner always finds the optimal action immediately, but it still needs to plan the entire trajectory. In those cases, any computation done in parallel is not useful. As a consequence there is an upper limit to the number of leaves we can expand in parallel without hurting the run time performances: when we can no longer reduce the number of planning iterations to compensate the overhead of adding more parallel processes without hurting the success rate of the algorithm.

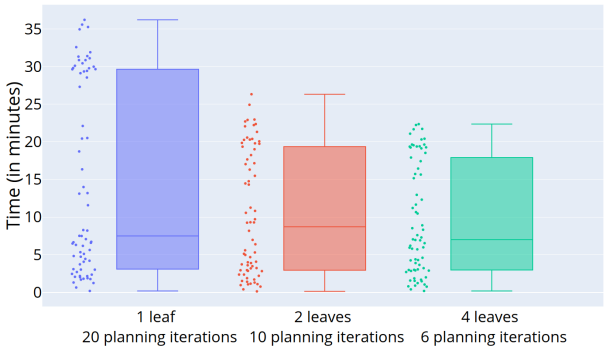

Fig. 4. **Time (in minutes) comparison of running OPD on the benchmark examples using no parallelization (blue), expanding 2 leaves in parallel (red) and expanding 4 leaves in parallel (green). The sample budget (number of planning iterations) is tuned to achieve the same success rate on all three versions.**

4) Results: On the 70 planning tests, the fastest version of OPD, expanding 4 leaves in parallel and with 6 planning iterations, achieves a success rate of 43% (30 successful trajectories out of 70). It also solves more than 80% of the

trajectory on 52% of the experiments as shown in Fig. 5.

The algorithm always runs in less than 23 minutes on a computer with two Intel Xeon E5-2630 v4 CPUs at 2,20 GHz (using 16 of the 20 available cores). The median run time is 7 minutes as shown in Fig. 6. It is interesting to note that the successful trajectories always have a shorter run time than the unsuccessful ones. Indeed, in unsuccessful cases the algorithm will get stuck in an action loop, repeating forward and backward motions without being able to cross the bifurcation. The maximum length of trajectories is 70 actions. Unsuccessful trajectories will run for 70 actions while successful ones will find a shorter action sequence and thus finish the experiment faster.

In the context of generating example trajectories to show the cardiologist feasible ways of navigating to the target location, or in the context of testing several instrument shapes, these computation times are acceptable. Indeed coronary CT scans are usually done at least several hours before the intervention, thus running one or several times an algorithm that runs in less than 30 minutes is acceptable. However, in the context of autonomous navigation, where the planning would be integrated in a larger control scheme, these run times would not be acceptable. Indeed, the planning would need to be recomputed several times online to account for the simulation registration.

C. Coupling tree search planning and inverse control

1) Principle: One of the main limitations of using OPD is that the action space has to be discrete. This implies that the control inputs can not always be as precise as needed, especially on the rotation. Adding more actions to have more choice is not possible without increasing the branching factor and thus, decreasing the efficiency of the tree search algorithm. Reducing the value of the rotation (smaller step size) does not work either as this makes the action sequences much longer. Another possible way to tackle this problem is to use another tree search algorithm which handles continuous action spaces. However those algorithms have a higher complexity which results in longer computation times [42], [43].

Instead we seek to combine the best of the two previous approaches. The QP being based on a convex approach has a deterministic behavior. Using the result of the QP to move forward can therefore be used as a possible discrete meta-action in the MDP. This action will allow for continuous translation and rotation, as long as it does not fall into a local minimum. And precisely, when it does get stuck in such a local minimum, the algorithm will be able to use one of the 4 other actions to get out of it.

More precisely, each step of the MDP, meaning each action taken by the high level controller, corresponds to 50 simulation steps. The four actions we defined in section III-D correspond to a low level controller applying a constant command of $translation/50$ or $rotation/50$ at each simulation step. In this section we introduce a fifth high level action which we name *Apply QP control*. This high level action corresponds on the low level to applying $translation_{qp}(i)$ *and* $rotation_{qp}(i)$, $i \in \{1, ..., 50\}$ at each of the 50 simulation steps. When this fifth high level action is selected, at each simulation step we solve the QPCC problem as in section IV-A and apply this low level command at each of the 50 simulation steps. The new action space of the MDP is thus:

1) Translate forward
2) Rotate clockwize
3) Rotate counter clockwize
4) Translate backward
5) Apply QP control

2) Results: On our benchmark, this algorithm achieves a success rate of 90% (63 successful trajectories out of 70) which is 3 successful trajectories short of the maximum expected success rate. This means the algorithm achieves 95% of successful trajectories obtained in manual mode, our maximum expected success rate (see section III-B).

As shown in Fig. 6, the maximum run time of the fastest version of this algorithm, expanding 4 leaves in parallel and with a budget of 5 planning iterations, is 35 minutes. The median run time is less than 3 minutes. The median computation time is close to what would be acceptable in a clinical context.

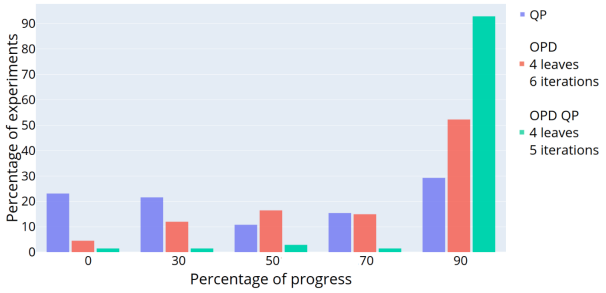

Fig. 5. **Comparison of the progress made by the 3 control strategies. The QP Control achieves more than 80% of the trajectory in 29% of cases, OPD achieves more than 80% of the trajectory in 52%, the combined strategy coupling tree search and inverse control achieves more than 80% of the trajectory in 93%**

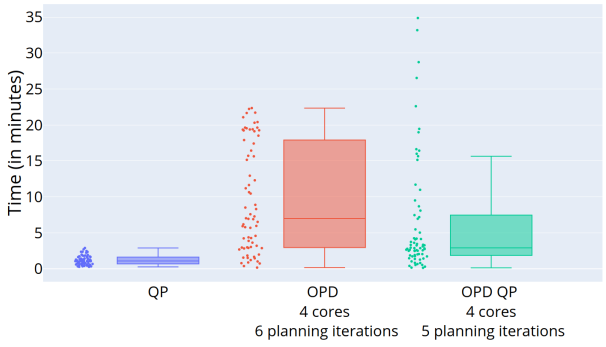

Fig. 6. **Comparison of the run time of the fastest version of the 3 control strategies. The QP is fastest and always runs in less than 3 minutes. The maximum run time of our combined strategy is higher than that of parallelized OPD (35 minutes versus 22) but the median run time of the combined approach is 2.4 times smaller.**

V. Discussion

A. Using planning to optimize the instruments' shape

In a real intervention, cardiologists choose the shape of the guidewire depending on the geometry of the arteries and the location of the lesion to treat by either choosing a pre-curved guidewire, or manually deforming the tip. We have not yet included the guidewire tip curvature radius as a planning parameter but we can run the planning algorithm several times with different radii. The radius which delivers the highest reward is likely to be the most appropriate for the intervention, using this control strategy. We could also generate navigation videos with different instruments to help novice cardiologists choose the most appropriate guidewire.

We show in the accompanying video and Figure 7 the same trajectory with two different instrument radii (on top the curvature of the guidewire has a radius of 2.5mm and on the bottom 3.5mm). The algorithm can't navigate in the branch with the 2.5mm instrument but can with the 3.5mm one. We assume that if the control is easier for the algorithm with a specific instrument shape, it is likely that it would also be easier for human control with this same shape.

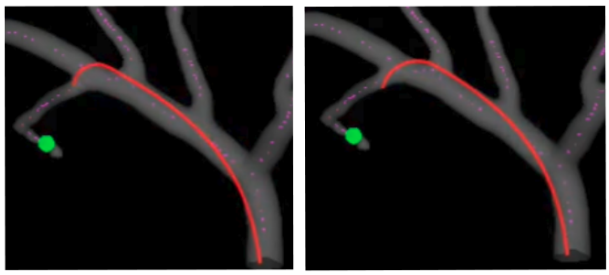

Fig. 7. **Different instruments (left 2.5 mm and right 3.5 mm) attempting the same trajectory.**

B. Manipulating several instruments

Contrary to existing methods in the literature, our method based on tree search based planning and discrete actions easily extends to manipulating several instruments. Indeed, to manipulate both a guidewire and a coaxial catheter for instance, one simply needs to add 2 discrete actions to the 4 listed in section III-D. Those 2 actions are to move the catheter forward and backward.

Preliminary results suggest an improvement in performance when multiple instruments are used. As shown in the video paired with this article, the concentric catheter (green instrument) deforms the guidewire (red instrument) and provides additional rigidity that improves the maneuverability near the branch. This allows in this example to get the instruments into a branch which isn't accessible using only the guidewire of this shape (2.5mm).

Increasing the number of actions from 4 to 6 increases the branching factor of the tree search, which makes the search more complex. However as stated in section IV-B.4, successful trajectories are computed faster than unsuccessful ones. Thus trajectories which are unsuccessful with one instrument tend to be computed faster when using 2 instruments.

C. Preventing hazardous trajectories

One main limitation of our algorithm is that though it can most of the time solve a trajectory, it will sometimes exhibit dangerous behaviours as shown in the video accompanying this article and Figure 8. In a clinical setting, cardiologists rely on visual cues and manual force feedback to estimate the risk of dissection. An extension of our algorithm could be to detect those dangerous situations and forbid those states by making them terminal and with a reward of 0. The tree search planning algorithm would then naturally find another trajectory.

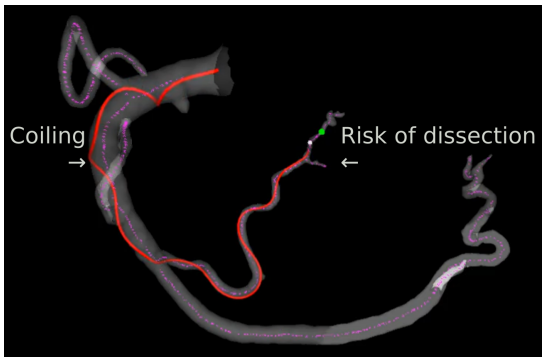

Fig. 8. **In the main branch (top left) the guidewire is coiling on itself, creating a spring. This behavior exerts large forces at the tip of the guidewire, which can lead to a dissection of the artery.**

VI. Conclusion and Future Work

In this work, we introduce a framework for planning the navigation of guidewires within blood vessels, tailored to the geometry of the patient. The method is based on a tree search combined with an inverse model obtained by convex optimization.

In future work, two challenges still need to be overcome for the method to be truly usable in practice. The first is to be able to emulate the influence of the patient's heartbeat movements on the arteries, for which no routine data exist. We are working on integrating this in our simulator. Preliminary results show that the planning algorithm presented in this article should transfer to a model with deforming vessels, but this has to be further verified. The second is the famous sim-to-real challenge and in our case the transfer to a physical robot and physical artery phantom. To compensate for unavoidable modeling errors, this transfer will introduce vision, simulation registration and closed loop control.

Acknowledgements

Authors thank Dr. Giovanni Biglino and Dr. Matthew Lee for their help with the 3D coronary geometries. Authors also thank Dr. Bruno Fournier for his advice on all parts of this project.

References

[1] World Health Organization, "The top 10 causes of death." https://www.who.int/en/news-room/fact-sheets/detail/the-top-10-causes-of-death, 2020.

[2] N. Townsend, L. Wilson, P. Bhatnagar, K. Wickramasinghe, M. Rayner, and M. Nichols, "Cardiovascular disease in Europe: epidemiological update 2016," *Eur Heart J*, vol. 37, pp. 3232–3245, Nov. 2016.

[3] Corindus, "Corindus corpath grx." `https://www.corindus.com/corpath-grx/how-it-works`, 2021.

[4] Robocath, "Robocath r-one." `https://www.robocath.com/product/`, 2021.

[5] L. W. Klein, Y. Tra, K. N. Garratt, W. Powell, G. Lopez-Cruz, C. Chambers, J. A. Goldstein, and On Behalf of the Society for Cardiovascular Angiography and Interventions , "Occupational health hazards of interventional cardiologists in the current decade: Results of the 2014 scai membership survey," *Catheterization and Cardiovascular Interventions*, vol. 86, no. 5, pp. 913–924, 2015.

[6] L. Venneri, F. Rossi, N. Botto, M. G. Andreassi, N. Salcone, A. Emad, M. Lazzeri, C. Gori, E. Vano, and E. Picano, "Cancer risk from professional exposure in staff working in cardiac catheterization laboratory: Insights from the national research council's biological effects of ionizing radiation vii report," *American Heart Journal*, vol. 157, no. 1, pp. 118–124, 2009.

[7] E. Mahmud, J. Naghi, L. Ang, J. Harrison, O. Behnamfar, A. Pourdjabbar, R. Reeves, and M. Patel, "Demonstration of the safety and feasibility of robotically assisted percutaneous coronary intervention in complex coronary lesions: Results of the cora-pci study (complex robotically assisted percutaneous coronary intervention)," *JACC: Cardiovascular Interventions*, vol. 10, no. 13, pp. 1320–1327, 2017.

[8] A. Stevenson, A. Kirresh, M. Ahmad, and L. Candilio, "Robotic-assisted pci: The future of coronary intervention?," *Cardiovascular Revascularization Medicine*, 2021.

[9] S. Jeong, Y. Chitalia, and J. P. Desai, "Design, modeling, and control of a coaxially aligned steerable (coast) guidewire robot," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4947–4954, 2020.

[10] S. Jeon, A. K. Hoshiar, S. Kim, S. Lee, E. Kim, S. Lee, K. Kim, J. Lee, J.-y. Kim, and H. Choi, "Improving guidewire-mediated steerability of a magnetically actuated flexible microrobot," *Micro and Nano Systems Letters*, vol. 6, no. 1, pp. 1–10, 2018.

[11] E. Vander poorten and et al., "Cognitive AutonomouS CAtheters Operating in Dynamic Environments," *Journal of Medical Robotics Research*, vol. 01, no. 03.

[12] A. T. Tibebu, B. Yu, Y. Kassahun, E. Vander Poorten, and P. T. Tran, "Towards autonomous robotic catheter navigation using reinforcement learning," *4th Joint Workshop on New Technologies for Computer/Robot Assisted Surgery*, pp. 163–166, 2014.

[13] T. Behr, T. P. Pusch, M. Siegfarth, D. Hüsener, T. Mörschel, and L. Karstensen, "Deep Reinforcement Learning for the Navigation of Neurovascular Catheters," *Current Directions in Biomedical Engineering*, vol. 5, pp. 5–8, Sept. 2019.

[14] L. Karstensen, T. Behr, T. Pusch, F. Ullrich, and J. Stallkamp, "Autonomous guidewire navigation in a two dimensional vascular phantom," *Current Directions in Biomedical Engineering*, vol. 6, p. 20200007, 09 2020.

[15] Y. Cho, J.-H. Park, J. Choi, and D. E. Chang, "Image processing based autonomous guidewire navigation in percutaneous coronary intervention," in *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pp. 1–6, IEEE, 2021.

[16] M. C. Yip and D. B. Camarillo, "Model-Less Hybrid Position/Force Control: A Minimalist Approach for Continuum Manipulators in Unknown, Constrained Environments," *IEEE Robotics and Automation Letters*, vol. 1, pp. 844–851, July 2016.

[17] J. Lenoir, S. Cotin, C. Duriez, and P. Neumann, "Interactive physically-based simulation of catheter and guidewire," *Computers & Graphics*, vol. 30, no. 3, pp. 416–422, 2006.

[18] J. Dequidt, M. Marchal, C. Duriez, E. Kerien, and S. Cotin, "Interactive simulation of embolization coils: Modeling and experimental validation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 695–702, Springer, 2008.

[19] J. Dequidt, C. Duriez, S. Cotin, and E. Kerrien, "Towards interactive planning of coil embolization in brain aneurysms," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 377–385, Springer, 2009.

[20] L. T. Kohn and J. M. Corrigan, "donaldson, ms, eds. to err is human: Building a safer health system," *Committee on Quality of Health Care in America. Institute of Medicine*, 1999.

[21] P. Korzeniowski, R. J. White, and F. Bello, "Vcsim3: a vr simulator for cardiovascular interventions," *International journal of computer assisted radiology and surgery*, vol. 13, no. 1, pp. 135–149, 2018.

[22] S.-H. Mi, Z.-G. Hou, F. Yang, X.-L. Xie, and G.-B. Bian, "A 3d virtual reality simulator for training of minimally invasive surgery," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 349–352, IEEE, 2014.

[23] H. Sharei, T. Alderliesten, J. J. Van Den Dobbelsteen, and J. Dankelman, "Navigation of guidewires and catheters in the body during intervention procedures: a review of computer-based models," *Journal of Medical Imaging*, vol. 5, no. 1, p. 010902, 2018.

[24] A. Hooshiar, S. Najarian, and J. Dargahi, "Haptic telerobotic cardiovascular intervention: a review of approaches, methods, and future perspectives," *IEEE reviews in biomedical engineering*, vol. 13, pp. 32–50, 2019.

[25] SOFA Consortium, "Simulation open framework architecture." `https://github.com/sofa-framework/sofa`, 2020.

[26] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, and C. Duriez, "Software toolkit for modeling, simulation, and control of soft robots," *Advanced Robotics*, vol. 31, pp. 1208–1224, Nov. 2017.

[27] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[28] P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, P. Preux, and C. Duriez, "Sofagym: An open platform for machine learning based on soft robot simulations." submitted.

[29] Z. Zhang, J. Dequidt, J. Back, H. Liu, and C. Duriez, "Motion control of cable-driven continuum catheter robot through contacts," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1852–1859, 2019.

[30] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1413–1419, 2017.

[31] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*, pp. 72–83, Springer, 2006.

[32] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, pp. 282–293, Springer, 2006.

[33] S. Bubeck and R. Munos, "Open loop optimistic planning," *COLT 2010 - The 23rd Conference on Learning Theory*, pp. 477–489, 01 2010.

[34] E. Leurent and O.-A. Maillard, "Practical Open-Loop Optimistic Planning," *arXiv:1904.04700 [cs, stat]*, Apr. 2019. arXiv: 1904.04700.

[35] J.-F. Hren and R. Munos, "Optimistic planning of deterministic systems," in *European Workshop on Reinforcement Learning*, (France), pp. 151–164, 2008.

[36] P. Medrano-Gracia, J. Ormiston, M. Webster, S. Beier, C. Ellis, C. Wang, A. A. Young, and B. R. Cowan, "Construction of a coronary artery atlas from ct angiography," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014* (P. Golland, N. Hata, C. Barillot, J. Hornegger, and R. Howe, eds.), (Cham), pp. 513–520, Springer International Publishing, 2014.

[37] M. Lee, S. Moharem-Elgamal, R. Beckingham, M. Hamilton, N. Manghat, E. G. Milano, C. Bucciarelli-Ducci, M. Caputo, and G. Biglino, "Evaluating 3D-printed models of coronary anomalies: a survey among clinicians and researchers at a university hospital in the UK," *Open access*, p. 10, Jan. 2019.

[38] Y. Wei, S. Cotin, J. Allard, L. Fang, C. Pan, and S. Ma, "Interactive blood-coil simulation in real-time during aneurysm embolization," *Computers & Graphics*, vol. 35, no. 2, pp. 422–430, 2011.

[39] Nazim Haouchine, "Sofaskeletonplugin." `https://github.com/rouge1616/SofaSkeletonPlugin`, 2020.

[40] The CGAL Project, *CGAL User and Reference Manual*. CGAL Editorial Board, 5.3 ed., 2021.

[41] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, p. 269–271, 1959.

[42] C. Mansley, A. Weinstein, and M. L. Littman, "Sample-based planning for continuous action markov decision processes," in *Proceedings of the Twenty-First International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'11, p. 335–338, AAAI Press, 2011.

[43] A. Weinstein and M. Littman, "Bandit-based planning and learning in continuous-action markov decision processes," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 22, pp. 306–314, May 2012.