2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Global/local motion planning based on Dynamic Trajectory Reconfiguration and Dynamical Systems for Autonomous Surgical Robots

Narcís Sayols[1], Alessio Sozzi[2], Nicola Piccinelli[3], Albert Hernansanz[1], Alicia Casals[1], Marcello Bonfè[2] and Riccardo Muradore[3]

Abstract— **This paper addresses the generation of collision-free trajectories for the autonomous execution of assistive tasks in Robotic Minimally Invasive Surgery (R-MIS). The proposed approach takes into account geometric constraints related to the desired task, like for example the direction to approach the final target and the presence of moving obstacles. The developed motion planner is structured as a two-layer architecture: a global level computes smooth spline-based trajectories that are continuously updated using virtual potential fields; a local level, exploiting Dynamical Systems based obstacle avoidance, ensures collision free connections among the spline control points. The proposed architecture is validated in a realistic surgical scenario.**

I. Introduction

In the longstanding debate on the features of robots against those of humans, the capability to move in unknown or dynamic environments is a central aspect. This feature is normal for humans, that typically live and work in such environments and excel in interacting with moving objects, while its achievement by robots is quite challenging and require advanced sensing technologies (i.e. to detect objects and estimate/predict their motion) and fast reasoning, planning and control systems. The presence of humans in the operational space of a robot further complicates its dynamic motion adaptation. Indeed, their behaviour may be difficult to predict precisely even if they are supposed to execute a known task. On the other hand, the knowledge about the task to be performed by both humans and robots may allow to introduce simplifying assumptions (e.g. a phase of the task may require the presence of a human in a subset of the available workspace) and ease the design of dynamic and reconfigurable robot motion planners.

For the latter reason, the manufacturing industry is the domain that has promoted more, so far, the raise of human-robot collaborations (i.e. the so-called *4th Industrial Revolution*) in terms of workspace sharing or even physical interaction [1], with a strong emphasis on collision avoidance for human safety [2]. Another domain in which human safety is the primary concern and robots are more and more employed is medical surgery. Indeed, the use of teleoperated robotic systems is a *gold standard* for Minimally Invasive Surgery (MIS) since the commercial release of the well-known daVinci® system (Intuitive Surgical, Inc.) almost 20 years ago. Nowadays, the most appealing and debated challenge in robotic surgery is the introduction of certain levels of autonomy in robot behaviour [3], [4], implying technical advances in collision-free motion planning and physical environment interaction. Indeed, in the surgery domain, such tasks are further complicated, especially in case of operations involving soft tissues, because the dynamics of non-rigid anatomical environments is much more difficult to predict than in manufacturing plants and because contacts with objects with unknown/variable viscoelastic properties are more difficult to control.

On the other hand, the growth of investments in research and development projects for autonomous surgical robotics demonstrates the confidence and the expectations of the medical community on the benefits of such technologies. Examples of such projects are given by EU funded I-SUR (*Intelligent Surgical Robotics*, see [5]), MURAB (*MRI and Ultrasound Robotic Assisted Biopsy*, see [6]) and SARAS (*Smart Autonomous Robotic Assistant Surgeon*, see `https://saras-project.eu`). In particular, the objective of the latter is to develop assistive surgical robots, autonomously operating in the same workspace of either a teleoperated surgical robot or a manually driven surgical tool.

The application context of SARAS is laparoscopic MIS, a kind of surgery in which tools are inserted into the abdomen of a patient through so-called *trocars*. In this scenario, motion planning of the assistive robots must cope with the dynamics of human driven tools and of patient's organs, which are predictable only within a short time horizon. Therefore, the robot trajectories must be generated with an adaptive and reactive strategy. Planning methods based on random sampling may not fulfil the timing constraints, even though several solutions are proposed in the literature to speed up replanning and online path adaptation [7]. Moreover, they inherently produce non deterministic behaviours, not compatible with such a safety critical application. Conversely, purely reactive solutions (i.e. based on the instantaneous computation of the desired motion towards a target or avoiding an obstacle,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 779813 (SARAS).

[1]N. Sayols, A. Hernansanz and A. Casals are with the Universitat Politècnica de Catalunya, Barcelona, Spain `{narcis.sayols,albert.hernansanz,alicia.casals}@upc.edu`

[2]A. Sozzi and M. Bonfè are with the Department of Engineering, University of Ferrara, Italy `{alessio.sozzi,marcello.bonfe}@unife.it`

[3]N. Piccinelli and R. Muradore are with the Department of Computer Science, University of Verona, Italy `{nicola.piccinelli,riccardo.muradore}@univr.it`

as in [8] or [9]) may not allow to fully control the shape of generated paths or the way in which the final target is approached, which is instead relevant for the proper execution of some surgical tasks (e.g. suturing, resecting).

In this paper, we aim to address the generation of collision-free trajectories for surgical robots in laparoscopic MIS assistive tasks, using a two-level approach merging the benefits of an offline geometric path construction method with those of online trajectory reconfiguration and reactive adaptation. At a global level, the path is built according to the initial knowledge on the operating scene and the requirements of the surgical tasks. Then, the path defined in terms of a set of control points and an interpolating Catmull-Rom spline [10]) is reconfigured with respect to the dynamic environment with an approach based on artificial potential fields, inspired by the seminal work on Elastic Strips [11]. Finally, a local level computes the robot trajectory, preserving its collision-free property even in presence of obstacles with small diameter (i.e. the manually driver surgical instruments), by enforcing a velocity modulation technique derived from the Dynamical Systems (DS) based approach of [9].

It is important to highlight that all of these planning methods are specifically reworked to take into account the peculiar kinematic constraints of laparoscopic surgical robots. Indeed, the previously mentioned trocar imposes a constraint on the degrees of freedom (DOFs) of the surgical tool, so that the latter can rotate around the insertion point (also called Remote Center of Motion, RCM), but can only translate along the direction connecting its tip and the insertion point itself. The main contributions of the paper are the following:

- the usage of depth maps to ensure the reachability of all control points during the trajectory taking into account the tool geometry in real-time,
- the two-level approach to motion planning: a global dynamic force field trajectory reconfiguration to compute control points (**CP**s) and a local planner to ensure the obstacle avoidance between two consecutive **CP**s,
- an extension of the DS-based approach to obstacle avoidance for a robot geometrically modelled as a rod constrained by an RCM, which is suitable for surgical laparoscopic MIS.

II. Dynamic Trajectory Planner

The SARAS control system is subdivided into several modules. As shown in Figure 1 given the desired task, the workspace geometrical representation and current pose of the robot the system is able to plan and control the task execution.

The state machine controller determines the high level control actions to be executed by the autonomous robots, the robot trajectories are defined by a sequence of two modules. The first, called Dynamic trajectory reconfigurator, generates a trajectory defined by a set of free of collision **CP**s using the current and goal points as well as collision risk information provided by collision risk estimator module. The second one, called local modulation planner, computes

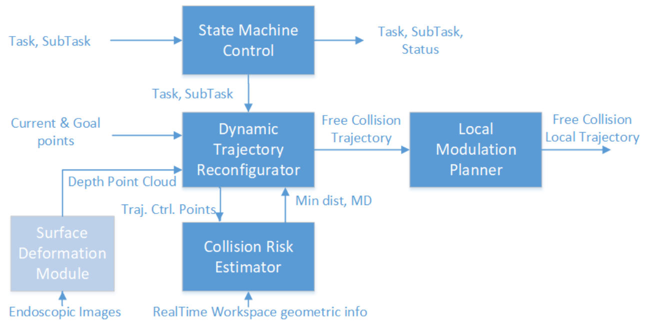


Fig. 1. Block diagram of all control modules of SARAS control system.

free collision trajectories between each consecutive pair of **CP**s.

A. Dynamic Trajectory Reconfiguration

The dynamic trajectory reconfiguration (DTR) module starts with the computation of an initial smooth trajectory (C^1 third order polynomial) from the current tool position to the destination pose (which can be either fixed or dynamically updated). When the destination pose is over a surface, the trajectory reaches the point following the normal to the surface (in the destination pose) and imposing the second derivative of the trajectory in the goal pose equal zero. The output of the module is a set of control points equally spaced all along the initial trajectory. The number of control points depends on the length of the initial trajectory.

DTR module is also in charge to eventually recompute at each control step the initial **CP**s. This continuous trajectory reconfiguration is necessary to ensure collision free trajectories in a dynamic workspace (deformable tissues and other laparoscopic tools which moves in the workspace). The reconfiguration of the control points $\{\mathbf{CP}_0, \ldots, \mathbf{CP}_n\}$ is done using artificial potential fields [12], [13].

Each **CP** is described as a mass affected by attraction and repulsion forces generated by the different elements of the workspace (repulsion forces following the minimum distance vectors between SARAS tool and each obstacle) and the original trajectory **CP**s (attraction forces from modified to original **CP**s positions). The result is a smooth locally-modified trajectory.

At every control step the collision risk at each **CP** is computed and the corresponding forces are applied following the rules below:

- repulsion forces created by the obstacles in the plane perpendicular to the trajectory gradient,
- an attraction force to the initial position of the **CP** if the obstacle force is zero,
- an attraction force to the farthest among the direct neighbour **CP**s to ensure the coherence in the movement of the **CP**s.

In order to ensure the collision avoidance for all the geometry of the tool (not only the tip) a Depth Map 2D matrix of the workspace around tool direction in each **CP** is computed: DM^k for the k-th **CP**. The tool movement restrictions imposed by RCM and the geometry of the tool allow to transform the tridimensional workspace representation into spherical coordinates (r, θ, ϕ) with origin in the tool RCM. The matrix element $DM^k_{i,j}$ is the maximum free collision depth reachable by the tool in θ_i, ϕ_j direction in the $\mathbf{CP}_k$. Each $\mathbf{CP}_k$ is also transformed into its spherical representation to be compared with its corresponding DM^k and detect if a collision occurs. If the comparison indicates a collision, the corresponding **CP** receives an attraction force in the direction of RCM.

Once the new collision free **CP**s are computed, DTR generates an interpolation spline inspired by the Catmull-Rom Spline, $\mathbf{p}(t)$ that goes through all n **CP**s

$$\mathbf{p}(t) = \sum_{i=1}^{n} \mathbf{p_i}(t)\mathbf{1}_{[t_{i-1},t_i]}(t) \tag{1}$$

where

$$\mathbf{1}_{[t_{i-1},t_i]}(t) = \begin{cases} [1,1,1] & \text{if } t \in [t_{i-1}, t_i] \\ [0,0,0] & \text{if } t \notin [t_{i-1}, t_i] \end{cases} \tag{2}$$

and $\mathbf{p}_i(t)$ satisfies the following conditions

$$\begin{aligned} \mathbf{p}_i(t_i) &= \mathbf{CP}_i \text{ for } i = 1 \dots n \qquad (3)\\ \mathbf{p}_i(t_{i-1}) &= \mathbf{CP}_{i-1} \text{ for } i = 1 \dots n \\ \dot{\mathbf{p}}_i(t_i) &= \alpha(\mathbf{CP}_{i+1} - \mathbf{CP}_{i-1}) \text{ for } i = 1 \dots n-1 \\ \dot{\mathbf{p}}_{i+1}(t_i) &= \alpha(\mathbf{CP}_{i+1} - \mathbf{CP}_{i-1}) \text{ for } i = 1 \dots n-1 \\ \dot{\mathbf{p}}_1(t_0) &= \mathbf{g}_0 \\ \dot{\mathbf{p}}_n(t_n) &= \mathbf{g}_n \end{aligned}$$

where $\mathbf{g}_0$ is the gradient of the trajectory at the current point and $\mathbf{g}_n$ the desired gradient at the goal point. α is a parameter used to define the velocity of the trajectory in the **CP**. Finally, the DTR module computes the new trajectory evaluating the spline in the new **CP**s and sends the trajectory coefficients to the local planner.

B. Collision Risk Estimation module

This module determines the Collision Risk **CR** between the objects and tools inside the workspace. The collision risk estimation module computes the minimum distance vectors from tools to obstacles in all **CP**s describing a trajectory, following the methodology described in [14]. The collision risk is determined by the derivative of these vectors along a period of time to foresee the proximity, in time, of a possible collision.

The module requires the estimated trajectories of tools and obstacles (set of predicted poses) and generates a vector of $\mathbf{CR}_{ijk}$, where i represents the i-th tool, j represents the j-th obstacle and k the k-th predicted control points of these trajectories. The evolution of $\mathbf{CR}_{ij}$ along k determines the risk of two bodies (normally, a tool and an obstacle) to collide.

The depth map technique described in Section II can also be used to simplify and accelerate the computation of the tools **CR**s. These minimum distance vectors, their derivatives and the DMs are sent to the dynamic trajectory reconfiguration module to compute the new collision free trajectory. Figure 2 shows an example of a DM in a workspace with a laparoscopic tool as an obstacle.

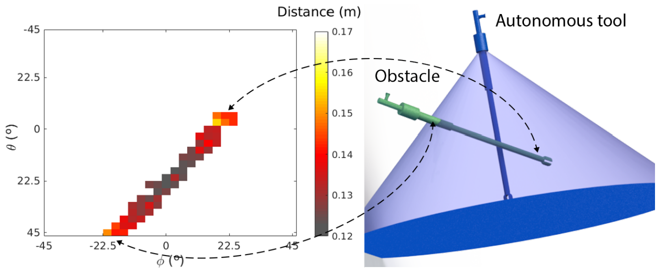


Fig. 2. Example of generated depth map of a possible intersecting tool. In this case the obstacle is represented by the green tool.

III. Local Modulation Planner

The DTR guarantees that the spline control points are collision free, but this property may not be preserved along a trajectory segment between two adjacent control points. From a practical point of view, it could be possible to reduce the risk of collisions by increasing the number of control points. Indeed, this operation would reduce the space between such control points, but it would also increase the computational effort for full trajectory reconfiguration, which may not be compatible with the timing constraints of the application.

On the other hand, the surgical scenario that we are considering includes obstacles with a rod-like shape and a very thin diameter (i.e. the surgical tools driven by the human surgeon) and the latter may be smaller than the distance between two adjacent control points. Therefore, a mechanism to enforce collision free execution of the trajectory computed by the DTR module is required.

In the proposed approach at each control step the trajectory is locally modified by a reactive local planner based on the algorithm outlined in [15]. This algorithm modulates the desired velocity of the robot to obtain a new velocity which drives the robot on a collision free trajectory.

The desired velocity of the robot $\dot{\mathbf{p}}(t_i)$ can be computed at the control step t_i by deriving the spline of the original trajectory with respect to time. The velocity $\dot{\mathbf{p}}(t_i)$, which is computed at the tip of the tool, is then scaled to obtain the velocity $\dot{\overline{\mathbf{p}}}_c$ of the point closest to the closest obstacle $\overline{\mathbf{p}}_c$:

$$\begin{aligned} \dot{\mathbf{p}}_{proj} &= (\dot{\mathbf{p}} \cdot m)\frac{m}{\|m\|} \qquad (4)\\ \dot{\mathbf{p}}_{res} &= \dot{\mathbf{p}} - \dot{\mathbf{p}}_{proj} \qquad (5)\\ \dot{\overline{\mathbf{p}}}_c &= \frac{\|\overline{\mathbf{p}}_c - \mathbf{p}_t\|}{\|m\|}\dot{\mathbf{p}}_{res} + \dot{\mathbf{p}}_{proj} \qquad (6) \end{aligned}$$

where $m = \mathbf{p} - \mathbf{p}_{tr}$, $\mathbf{p}_{tr}$ is the insertion point of the commanded tool, $\dot{\mathbf{p}}_{proj}$ and $\dot{\mathbf{p}}_{res}$ are the components of $\dot{\mathbf{p}}$ in the direction of m and perpendicular to m respectively.

Since it is possible to model the laparoscopic tools as capsules, i.e. cylinders with hemispheric terminations (as described in [15]), the point on the axis of the tool closest to the closest obstacle can be computed using the parameterization of the segments introduced in [16]. In case of moving obstacles we also take into account their motion considering the velocity of the robot relative to the obstacles

$$\dot{\overline{\mathbf{p}}}_{rel} = \dot{\overline{\mathbf{p}}}_c - \dot{\overline{\mathbf{p}}}_{obs} \tag{7}$$

where $\dot{\overline{\mathbf{p}}}_{obs}$ is computed as follows

$$\dot{\overline{\mathbf{p}}}_{obs} = \sum_{k=1}^{K} w^k \, \dot{\mathbf{p}}^{o,k} \tag{8}$$

and where $\dot{\mathbf{p}}^{o,k}$ is the velocity of the point on the obstacle closest to the robot. w^k is a scalar value which weights the influence of the k-th obstacle in relation to its distance from the commanded tool. Each weight w^k can be computed as

$$w^k = \prod_{i=1, i\neq k}^{K} \frac{\Gamma^i - 1}{(\Gamma^k - 1) + (\Gamma^i - 1)} \tag{9}$$

The scalar value Γ^k is the value of the analytic function that describes the surface of the k-th obstacle in the point of the tool closest to that obstacle. For capsule shaped obstacles, Γ^k can be computed as follows

$$\Gamma^k = \sum_{i=1}^{3} \left(\frac{{}^k\mathbf{p}_{a,i} - {}^k\mathbf{p}_{c,i}}{{}^k r} \right)^2 - 1 \tag{10}$$

where ${}^k\mathbf{p}_{c,i}$ is the point on the axis of the i-th obstacle closest to the tool, ${}^k\mathbf{p}_{a,i}$ is the point on the axis of the tool closest to the i-th obstacle and ${}^k r$ is the radius of the capsule. The velocity $\dot{\mathbf{p}}^{o,k}$ should be computed in the closest point of the k-th obstacle to the commanded tool; if the k-th obstacle is a laparoscopic tool, it can be computed using Eqs. 4, 5 and 6, since it has the same insertion point constraint of the commanded tool. The modulation matrix $\overline{M}$ is then applied to the velocity $\dot{\overline{\mathbf{p}}}_{rel}$

$$\dot{\overline{\mathbf{p}}}_c^{\star} = \overline{M} \, \dot{\overline{\mathbf{p}}}_{rel} \tag{11}$$

where $\dot{\overline{\mathbf{p}}}_{rel}$ is the original velocity relative to the velocities of the obstacles, and $\dot{\overline{\mathbf{p}}}_c^{\star}$ is the modulated velocity. The velocity $\dot{\overline{\mathbf{p}}}_c^{\star}$ is then taken back into the robot reference frame using the relation

$$\dot{\overline{\mathbf{p}}}_{rel}^{\star} = \dot{\overline{\mathbf{p}}}_c^{\star} + \dot{\overline{\mathbf{p}}}_{obs} \tag{12}$$

and then scaled back to obtain the velocity at the tool tip (with operations similar to the operations in Eqs. 4, 5, 6). The modulation matrix $\overline{M}$ is constructed taking into account all the obstacles in the scene

$$\overline{M} = \prod_{k=1}^{K} M^k \tag{13}$$

with M^k modulation matrix of the k-th obstacle and K the number of the obstacles. Each modulation matrix M^k can be computed as

$$M^k = E^k D^k \left(E^k\right)^{-1} \tag{14}$$

where

$$E^k = \left[n^k \; e^{k,1} \; e^{k,2} \right] \tag{15}$$

$$e_j^{k,i} = \begin{cases} -\frac{\partial \Gamma^k}{\partial \xi_i^k} & \text{if } j = 1 \\ \frac{\partial \Gamma^k}{\partial \xi_1^k} & \text{if } j = i \neq 1, i \in 1 \dots m\text{-}1, j \in 1 \dots m\text{-}1 \\ 0 & \text{if } j \neq 1, j \neq i \end{cases} \tag{16}$$

and

$$D^k = \begin{bmatrix} 1 - \frac{\omega^k}{|\Gamma^k|^{\frac{1}{\rho}}} & 0 & 0 \\ 0 & 1 + \frac{\omega^k}{|\Gamma^k|^{\frac{1}{\rho}}} & 0 \\ 0 & 0 & 1 + \frac{\omega^k}{|\Gamma^k|^{\frac{1}{\rho}}} \end{bmatrix} \tag{17}$$

where w^k are the weights introduced in (9). The vector n^k is the normal to the surface of the k-th obstacle computed in the point of the axis tool closest to that obstacle. This vector can be obtained using the relation

$$n^k = {}^k\mathbf{p}_c - {}^k\mathbf{p}_a \tag{18}$$

where ${}^k\mathbf{p}_c$ is the point on the axis of the obstacle closest to the tool and ${}^k\mathbf{p}_a$ is the point on the axis of the tool closest to the obstacle.

Since the modulation deviates the motion of the robot from the original trajectory, at the end of the modulation it is necessary to recompute the spline to obtain a new C^1 trajectory which connects the new position of the robot with the following control point. This new spline will be used in the next control step t_{i+1} to compute the desired velocity.

IV. Case study and Experimental Validation

The proposed solution has been tested in the context of semi-autonomous surgery using the experimental setup developed in the EU funded project SARAS. The experimental setup shown in Figure 3 consists of:

- a daVinci surgical platform which has three independent arms. Two of them are called the Patient-Side Manipulators (PSM) and move the robotic tools. The third one is called Endoscopic Camera Manipulator (ECM) and holds the stereo endoscope. The master console has two controllers, called Master Tool Manipulators (MTM), that allow the surgeon to teleoperate the tools. The whole system is interfaced using the dVRK framework.
- two SARAS robotic arms, designed by Medineering GmbH. Each SARAS arm is composed of a 7 dof passive serial manipulator (SBS) holding a 4 dof manipulator (EGS). The EGS is a parallel manipulator which allows the creation of a virtual remote centre of motion and it holds the assistant laparoscopic tool. The SARAS robots are positioned in order to have the surgical scenario completely contained in their workspaces and to avoid singularities.

A surgical procedure is subdivided into several tasks (called phases) and our collision avoidance and motion planning module have been used to perform one of them. The task consists of pushing down the bladder during the

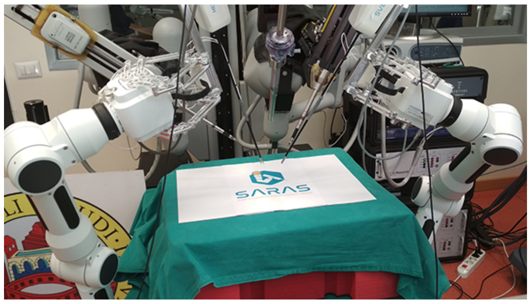


Fig. 3. SARAS experimental setup.

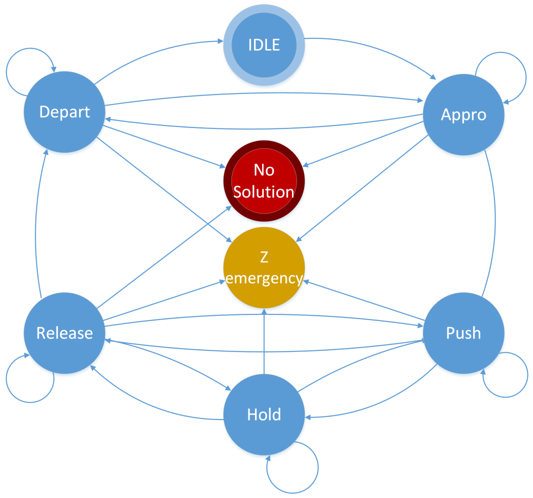


Fig. 4. Finite state machine for bladder pushing task in RARP surgery.

prostatectomy. The description of the full surgical procedure, in the context of the SARAS project, can be found in [17]. In the bladder pushing phase the assistant surgeon, in our case the autonomous system, performs the sequence of actions shown in Figure 4.

A. Common reference frame

The DTR and the DS module need a common reference frame to work properly. This reference frame is built using multi-robot base frame calibration. We built a calibration platform with a set of known points placed on a circumference with a known radius. Each arm records the relative position of the fiducial points moving the end effector and touching them. After that, we fit a plane for each point set and use it to create the base transformation for each arm.

B. Finite state machine

Robotic Assisted Radical Prostatectomy (RARP) can be described as a set of structured surgical tasks composed of atomic surgical actions or sub-tasks. This surgery description generates a decomposition into two complexity levels. These sub-tasks can be common to several surgical tasks. Each task (bladder manipulation in this case study) can be described as a finite state machine containing all atomic surgical actions augmented with two control states to resolve critical situations.

Figure 4 illustrates the finite state machine of a robot interacting with the bladder in RARP surgery. The FSM is composed of the following states:

Idle Initial state to start the bladder manipulation.

Appro Approximation to the desired bladder surface where the force will be applied. Defined by a smooth trajectory, the end-effector of the tool reaches the destination pose in the direction of the normal to the surface at that point.

Push Tool pushes the bladder surface with a straight trajectory to a pre-defined depth.

Hold Tool remains to push the bladder and adapting pose if a bladder movement occurs.

Release Antagonic to Push action. The tool describes a straight trajectory to let the bladder free.

Depart Antagonic to Appro, tool departs from the contact point following the normal restriction and reaches a pre-defined safe pose.

Z emergency Special state that generates an alert message to central teleoperation control warning indicating that the only safe solution is an emergency escape trajectory in Z_{tool}, extracting the laparoscopic tool from the workspace.

No solution Special state that generates a critical message to central teleoperation control indicating that SARAS robots cannot continue with the task. Consequently, an external solution is required (e.g. altering main tool trajectories, restricting access to some volume of the workspace, etc.).

C. Software architecture

The overall control architecture is developed using ROS (Robot Operating System, `http://www.ros.org`), therefore the two parts of the proposed method explained in the Sections II, III have been embedded into two C++ ROS nodes. The dynamic trajectory planner is composed of two threads:

- a thread working at 100 Hz which computes the spline starting from the control points and the position computed by the Local Modulation node. This node uses the last DMs calculated until next one is updated,
- a thread working at 20 Hz which performs the CR and the DMs.

The local modulation is also composed of two threads:

- a thread working at 200 Hz which modulates the desired velocity of the spline computed by the Dynamic Trajectory Planner node and computes the next position,
- a thread working at 20 Hz which sends the commands to the robot. This frequency is determined by the robot controller.

D. Results

The most challenging phases for a motion planner in the bladder pushing task are undoubtedly the **Appro** phase and the **Depart** phase since they need a trajectory planning in the whole workspace. In this section, we show the results of the experiments performed on the **Appro** phase of the task.

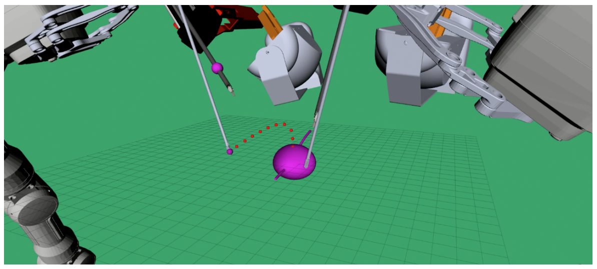
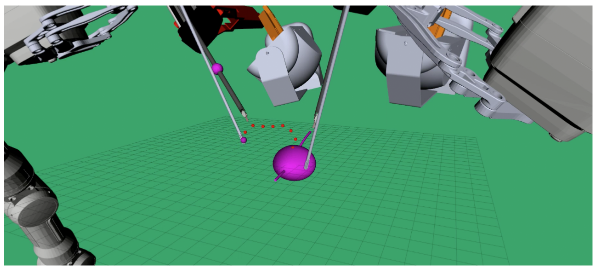
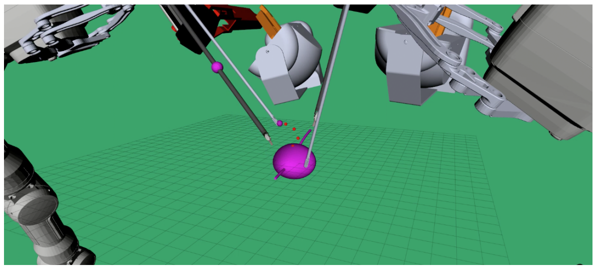


Fig. 5. The initial trajectory computed by the DTR. The grey stick is the commanded tool, the black stick is the dynamic obstacle and in red the control points.



(a) Trajectory reconfiguration during the dynamic obstacle approach



(b) Trajectory reconfiguration while approaching the goal

Fig. 6. Trajectory reconfiguration example. The grey stick is the commanded tool, the black stick is the dynamic obstacle and in red the control points adapted during the movement.

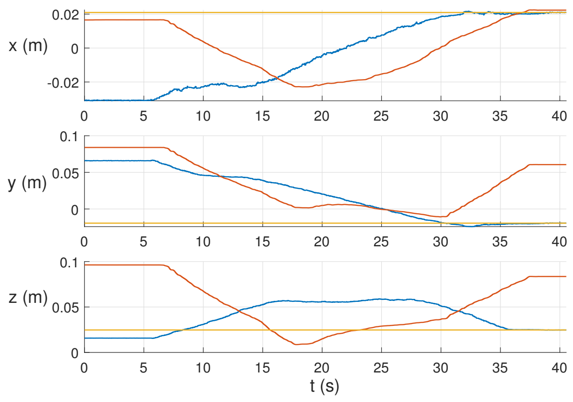


Fig. 7. Evolution of the cartesian coordinates of the commanded tool tip (blue line) and obstacle tip (red line) during the movement. In yellow: the coordinates of the goal.

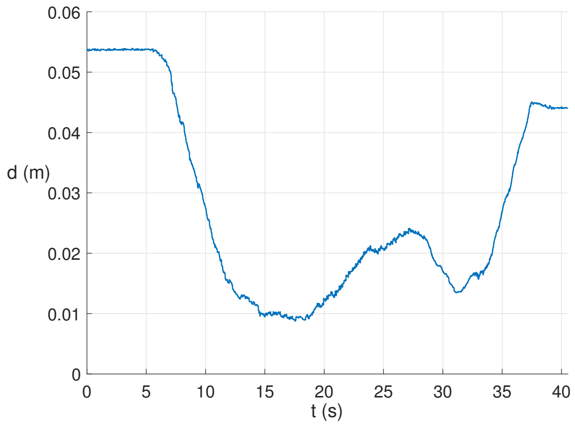


Fig. 8. Distance between the commanded tool and the obstacle during the movement.

At the beginning of the approach, the robot starts following the initial spline computed by the global planner (Figure 5, in red). During the execution of the trajectory, one of the daVinci arms approaches the commanded tool and then follows it for a short track, making the global planner gradually adapt the control points of the original trajectory into a new trajectory (Figure 6a and Figure 6b, in red). In all of the segments of the trajectory between two adjacent control points, the local planner reacts to possible unaccounted obstacles ensuring the collision free movement of the tool.

Figure 7 shows the evolution of the coordinates of the tool tip and the obstacle tip in the common reference frame. The goal is successfully reached and the evolution of the robot testifies the smoothness of the commanded trajectory. Finally, Figure 8 shows the distance d between the tool and the obstacle during the movement: since the value is always $d > 0$, the path is collision free.

For further details and the application of the proposed approach in a realistic surgical scenario please refer to the accompanying video.

V. Conclusions and Future Work

In this paper, we proposed an innovative approach to motion planning in a dynamic environment to be applied in R-MIS. This motion planner is based on a two layer architecture:

- a global planner which computes a smooth trajectory for the execution of the task and updates its control points accordingly to the obstacles using potential fields;
- a local planner which ensures obstacle avoidance through the use of an analytical modulation matrix while moving from a control point to the following one.

The proposed method has been successfully validated in a realistic surgical scenario. As a future development, we will extend the proposed method for the motion planning of coordinated tasks performed by both of the tools of the autonomous robotic assistant.

Acknowledgment

The authors would like to thank Johann Wigger, Sabine Hertle and Medineering GmbH for the development of the SARAS robots used in the experiments.

References

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on humanrobot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248 – 266, 2018.

[2] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach for evaluating distance to objects – with application to human-robot collision avoidance," *J. of Intelligent and Robotic Systems*, vol. 80, no. Suppl. 1, pp. 7–22, 2015.

[3] F. Ficuciello, G. Tamburrini, A. Arezzo, L. Villani, and B. Siciliano, "Autonomy in surgical robots and its meaningful human control," *Paladyn, Journal of Behavioral Robotics*, vol. 10, no. 1, pp. 30–43, 2019.

[4] T. Haidegger, "Autonomy for surgical robots: Concepts and paradigms," *IEEE Transactions on Medical Robotics and Bionics*, vol. 1, no. 2, pp. 65–76, May 2019.

[5] R. Muradore, P. Fiorini, G. Akgun, D. E. Barkana, M. Bonf, F. Boriero, A. Caprara, G. De Rossi, R. Dodi, O. J. Elle, F. Ferraguti, L. Gasperotti, R. Gassert, K. Mathiassen, D. Handini, O. Lambercy, L. Li, M. Kruusmaa, A. O. Manurung, G. Meruzzi, H. Q. P. Nguyen, N. Preda, G. Riolfo, A. Ristolainen, A. Sanna, C. Secchi, M. Torsello, and A. E. Yantac, "Development of a cognitive robotic system for simple surgical tasks," *International Journal of Advanced Robotic Systems*, vol. 12, no. 37, pp. 1–20, 2015.

[6] M. E. M. K. Abdelaziz, D. Kundrat, M. Pupillo, G. Dagnino, T. M. Y. Kwok, W. Chi, V. Groenhuis, F. Siepel, C. Riga, S. Stramigioli, and G. Yang, "Toward a versatile robotic platform for fluoroscopy and mri-guided endovascular interventions: A pre-clinical study," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[7] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[8] A. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[9] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[10] E. Catmull and R. Rom, "A class of local interpolating splines," in *Computer Aided Geometric Design*, R. Barnhill and R. Riesenfeld, Eds. Academic Press, 1974, pp. 317 – 326.

[11] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[12] R. R. Murphy, "Potential fields methology," in *Introduction to AI robotics (1st Ed.)*. MIT Press, 2000, pp. 105 – 153.

[13] N. Zhang, Y. Zhang, C. Ma, and B. Wang, "Path planning of six-dof serial robots based on improved artificial potential field method," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2017, pp. 617–621.

[14] A. Hernansanz, A. Casals, and J. Amat, "A multi-robot cooperation strategy for dexterous task oriented teleoperation," *Robotics and Autonomous Systems*, vol. 68, pp. 156 – 172, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889014003042

[15] A. Sozzi, M. Bonfè, S. Farsoni, G. De Rossi, and R. Muradore, "Dynamic motion planning for autonomous assistive surgical robots," *Electronics*, vol. 8, p. 957, 2019.

[16] V. J. Lumelsky, "On fast computation of distance between line segments," *Information Processing Letters*, vol. 21, pp. 55–61, 1985.

[17] E. Oleari, A. Leporini, D. Trojaniello, A. Sanna, U. Capitanio, F. Dehó, A. Larcher, F. Montorsi, A. Salonia, F. Setti, *et al.*, "Enhancing surgical process modeling for artificial intelligence development in robotics: the saras case study for minimally invasive procedures," in *2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)*. IEEE, 2019, pp. 1–6.