




Article

Dynamic Motion Planning for Autonomous Assistive Surgical Robots

Alessio Sozzi ¹, Marcello Bonfè ^{1,*} , Saverio Farsoni ¹ , Giacomo De Rossi ² and Riccardo Muradore ² ¹ Department of Engineering, University of Ferrara, 44122 Ferrara, Italy² Department of Computer Science, University of Verona, 37134 Verona, Italy

* Correspondence: marcello.bonfe@unife.it; Tel.: +39-0532-974839

Received: 26 July 2019; Accepted: 26 August 2019; Published: 29 August 2019



Abstract: The paper addresses the problem of the generation of collision-free trajectories for a robotic manipulator, operating in a scenario in which obstacles may be moving at non-negligible velocities. In particular, the paper aims to present a trajectory generation solution that is fully executable in real-time and that can reactively adapt to both dynamic changes of the environment and fast reconfiguration of the robotic task. The proposed motion planner extends the method based on a dynamical system to cope with the peculiar kinematics of surgical robots for laparoscopic operations, the mechanical constraint being enforced by the fixed point of insertion into the abdomen of the patient the most challenging aspect. The paper includes a validation of the trajectory generator in both simulated and experimental scenarios.

Keywords: robot motion planning; obstacle avoidance; dynamical systems; surgical robotics

1. Introduction

Modern applications of robotic technologies involve more and more frequently the interaction of robots with a dynamic environment. The variability in the environment may be due to the uncertainty on the presence of obstacles and on their location or to the fact that such obstacles may be moving and, if so, in unknown or unpredictable ways. A notable source of environmental variability is the involvement of humans, whose behavior is inherently uncertain, in the operations of robots. For example, the manufacturing industry domain has been promoting for several years the expansion of human–robot collaborations, in terms of both workspace sharing and physical cooperation [1]. In this scenario, robots must be able to move while avoiding collisions with humans or, if contacts are acceptable (i.e., physical human–robot interaction (pHRI)), to limit the forces exerted during such contacts. The mentioned features are mandatory, of course, to guarantee the safety of humans interacting with robots, and their realization entails sophisticated control and perception technologies. Further advances on cognition and automated reasoning for industrial collaborative robotics are being intensively studied by several authors [2]. Medical surgery is another important context in which human safety is the primary issue and the presence of robots and their required level of cognitive capabilities are constantly increasing. In particular, surgical robots are currently improving their role from one of a sort of human arm extension to one of artificial assistants, able to analyze the operating scenario and autonomously plan and execute their actions in support of human surgeons. Indeed, the use of teleoperated robotics systems has already been a *gold standard* for minimally-invasive surgery (MIS) for 20 years, while autonomous or semi-autonomous robots for surgical tasks, especially considering operations involving the treatment of soft tissues, have not yet reached high technological readiness levels (e.g., TRL 8 or 9). Notable exceptions are ROBODOC [3], CyberKnife [4], or NeuroMate [5], but they operate on either rigid tissues (i.e., bones) or restricted body portions. Increasing the autonomy level of surgical robots for more general MIS applications

requires addressing both technical and ethical/legal issues (see [6,7] for updated reviews). In particular, from the technological point of view, soft-tissue surgery in non-rigid anatomical environments forces taking into account hardly predictable scene changes, complicating the tasks of collision-free motion planning and physical environment interaction (i.e., contact with objects with unknown and even variable viscoelastic properties).

The growth of investments in research and development projects for autonomous surgical robotics demonstrates the confidence and the expectations of the medical community regarding the benefits of such technologies. For example, the European Union has recently funded several projects related to the automation of surgical tasks, like I-SUR (*Intelligent Surgical Robotics*, FP7 Grant No. 270396, <http://www.isur.eu/isur>), MURAB (*MRI and Ultrasound Robotic Assisted Biopsy*, Horizon 2020 Grant No. 688188, <https://www.murabproject.eu>), and SARAS (*Smart Autonomous Robotic Assistant Surgeon*, Horizon 2020 Grant No. 779813, <https://saras-project.eu>). In particular, the I-SUR project addressed the automation of needle insertion and suturing tasks [8] by means of a dual-arm robot with hybrid parallel/serial kinematics. The cognitive control architecture proposed by I-SUR [9] was able to operate in either teleoperated [10] or autonomous mode [11], guaranteeing a stable switch between the two and an adaptive interaction with the environment in both modes [12]. The inherent relationship between surgical and industrial collaborative robotics is demonstrated by the fact that the same control methods (i.e., admittance control with variable dynamics) have also been applied by the same authors to enforce stability in pHRI [13,14]. Turning back to the specific case of the suturing task, the work presented in [11] proposed a motion planning solution based on a combination of previously-specified motion primitives for the dual-arm system, designed to mimic the bimanual gestures of a human surgeon, and collision-free paths generated with a *plan-and-move* strategy. Similar approaches to surgical robotic suturing were described in [15,16], investigating advanced learning techniques, or [17,18], addressing the task using more classical robot motion planning techniques and analytic geometry. Even though surgical suturing tasks have also been automated by designing specific devices, not mimicking at all human gestures [19], the solutions based on general-purpose multi-arm robots and appropriate motion planners are more flexible and can be applied to different operations. Another notable example emphasizing the latter aspects can be found in [20].

In all of the mentioned applications, a fundamental issue to address is the generation of collision-free robot motions, avoiding obstacles that may be static, but at a previously-unknown position within the workspace, or moving, along an either predictable or unpredictable trajectory. The latter condition would prevail if the motion of obstacles is related to human behavior, for example because the obstacle itself is a human or because the obstacle is a tool teleoperated by a human through a haptic device. In this paper, we will primarily consider the problem of motion generation for an assistive surgical robot operating in the same workspace of either a teleoperated surgical robot or a manually-driven surgical tool, which is the application domain of the previously-mentioned SARAS project. In particular, we aim to address laparoscopic MIS, a kind of surgery in which the surgical tools are inserted into the abdomen of the patient through so-called *trocars*. The trocar at the tool entry point imposes a constraint on the degrees of freedom (DOFs) of the surgical instruments, in particular on the lateral translations, a constraint that is essential to guarantee that the operation is minimally invasive. In other words, the tool can freely rotate around the trocar insertion point, but can only translate along the direction connecting its tip and the insertion point itself. Therefore, the proposed motion generation scheme assumes that the controlled robot is a rod-like manipulator whose motion is compliant with the kinematic constraints imposed by the trocar. Furthermore, the proposed solution takes into account possibly moving obstacles, whose position and velocity are not known a priori, but their measurements are available at the current processing instant. Indeed, the motion generation strategy described in this paper can be classified as a *real-time adaptive motion planner* (RAMP) [21].

The research on RAMP is relatively new, especially considering robots acting in a dynamic and/or unknown environment, for which the plan-and-move strategy is not applicable. The latter aspect also means that many of the collision-free path planning methods proposed in the literature (e.g., RRT/RRT-Connect and all of their variants [22], just to name the most well known) and even supported by state-of-the-art software libraries [23,24] cannot be straightforwardly used. Notable approaches introducing reaction capabilities into a path planning framework are those exploiting the elastic deformations of a specified geometric path, by applying repulsive forces [25] or adjusting traveling velocities along the path [26]. However, many robotic tasks in a dynamic environment cannot be specified a priori by a geometric path. In such cases, methods based on random or probabilistic sampling, like RRT, can be extended to implement fast selection/adaptation of the proper motion in real time [27]. On the other hand, such approaches inherently produce indeterministic behaviors, which could be a disadvantage in safety-critical applications like surgical and industrial collaborative robotics.

The rest of the paper will present a motion generation method that is designed to be compatible with the requirements of an autonomous robot for laparoscopic MIS. The proposed method is derived from the approach based on the Dynamical Systems (DS) of Khansari-Zadeh and Billard [28] and relies uniquely on the knowledge of the environment, including the position and velocity of possibly moving obstacles, at the same instant in which the desired motion of the robot is generated. The advantages of the DS-based approach are the guarantee of a deterministic behavior with formally-provable obstacle avoidance properties and the limited computational requirement, especially in comparison with sampling-based or optimization-based methods. Indeed, the DS-based method allows analytically (i.e., with a closed-form solution) computing the robot velocity avoiding the obstacles, based on their shapes. On the other hand, most of the DS-based motion planners previously described in the literature considered point-like robots without kinematic constraints, which is a model not adequate for a laparoscopic surgical robot constrained by a trocar. A more suitable model for such a robotic device is instead a rod (or, more precisely, a *capsule*) subject to the mechanical constraint of a remote center of motion (RCM). In our proposal, the collision-free property of DS-based-generated motions for the considered capsule-like robot is obtained by modulating, using the theoretical tools of [28], the velocity of the closest point on the capsule to the closest obstacle and by computing in real time a suitable waypoint to guide the obstacle avoidance maneuver. Therefore, the main contributions of the paper are:

- an extension of the DS-based approach to real-time obstacle avoidance for a robot geometrically modeled as a capsule with an RCM, which is suitable for surgical laparoscopic MIS;
- a detailed analysis of the collision-free properties of the proposed motion generation method in both static and dynamic environments, including simulations and experimental results on a multi-robot system composed by real surgical manipulators, in a non-clinical setup.

2. Related Works

The generation in real-time of trajectories for robot motion is a problem that has attracted the interest of the research community since more than two decades. As real-time or online trajectory generation (OTG), we consider the case in which the desired motion state (i.e., position, velocity, and possibly acceleration) is calculated in the same instant in which it is commanded as a set-point to the robot controller, on the basis of the desired motion state at the previous instant and a possibly time-varying set of constraints. Such time-varying constraints typically include at least the final motion target. Indeed, the intrinsic motivation for OTG is the necessity for prompt adaptations of the robot motion to fast and unpredictable reconfigurations of the task. Other constraints that could be relevant, especially for robotic applications in which the smoothness of motions or its compatibility with physical limitation of the robot is primary, are the bounds on the time derivatives of the generated trajectory (i.e., velocity, acceleration, jerk, etc.).

A possible approach to OTG is to describe the trajectories in terms of piece-wise polynomials or *spline* functions and then to design fast computational algorithms allowing the update of the trajectory parameters even during the execution of the trajectory itself [29,30]. However, while such a solution is suitable for fast adaptation to motion target changes, it cannot easily take into account time-varying bounds on velocity or acceleration. Another interesting approach that is instead able to cope with the latter issue is the generation of the desired motion by means of nonlinear filters. Nonlinear filters for OTG were introduced first in [31], with a solution based on decision trees. Later on, trajectory generation based on variable-structure (VS) dynamical systems, acting as time-optimal smoothing filters for rough reference signals, was introduced in [32]. The latter reference presents a second-order version of the VS filter, which is able to limit the velocity and acceleration of its output within symmetric bounds. Further extensions of the method were proposed in [33] (third-order continuous-time filter with bounded jerk), [34] (second-order discrete-time filter with asymmetric bounds on velocity and acceleration), and [35] (third-order discrete-time filter with asymmetric bounds on velocity, acceleration, and jerk). However, such nonlinear filters are inherently single-DOF systems, and therefore, they are not designed to solve the obstacle avoidance problem. An adaptation of the nonlinear filtering approach to the specific context of wheeled mobile robots was proposed in [36,37], including obstacle avoidance capabilities in the latter reference.

Another solution to the OTG problem is the one developed mainly by Kröger, fully presented in [21] and implemented by the well-known Reflexxes libraries (RML, <http://reflexxes.ws/>). The method of Kröger provides most of the features of the nonlinear filters derived from [32], but adopts decision trees like in [31]. An additional feature of the OTG supported by RML is the possibility to synchronize multiple DOFs so that they reach their own target simultaneously. However, even this kind of reactive planner is unable to generate collision-free trajectories for a manipulator in a cluttered environment.

Considering instead obstacle avoidance as the primary objective, a relevant approach (i.e., based on purely reactive behavior) that must be mentioned first is the seminal work on Velocity Obstacles (VO) [38], proposing a method to compute the set of velocities that a circular robot (moving in a 2D space) should not achieve, to avoid the collision with another moving agent with a currently-known velocity. The method has been applied and extended by many other authors (see [39] for a recent example), even considering motions in a 3D space. However, most of the works in the literature related to VO addressed motion planning for mobile robots or aerial vehicles, rather than manipulators. With the latter in mind, we would like to conclude this literature review by mentioning the two approaches to robot motion planning, sharing several common aspects, which mostly inspired our work: the one based on dynamic movement primitives (DMP) [40] and the one based on dynamical systems (DS) [28]. The basic idea of the first method is to encode a trajectory by means of the definition of a stable nonlinear differential equation, whose behavior is attracted by the desired motion target. Even though the basic objective of the work proposing DMP is to learn a human-like or, more generally, a biologically-inspired behavior from demonstrations (see [41] for a recent application in industrial collaborative robotics), the approach may include obstacle avoidance features. However, in the context of DMP, obstacles are generally modeled as single points in a 3D space. The approach proposed by Khansari-Zadeh and Billard in [28], instead, puts a strong emphasis on the geometrical properties of the obstacles and allows us to take into account objects with rather complex shapes, by embedding them into multi-dimensional ellipsoids. Such geometric properties are exploited to modulate the behavior of a DS, so that the obstacle avoidance feature is added to its original features (e.g., stability, convergence to a goal, etc.). It is interesting to note that also the approach based on DS has been applied to motion estimation and learning, for teaching by demonstration tasks [42].

Since the DS-based methodology is the starting point of our developments, the next section will recall its main theoretical aspects.

3. DS-Based Obstacle Avoidance

The basic assumptions of the method outlined by Khansari-Zadeh and Billard in [28] are that the desired velocity of the robot is generated by an admissible differential equation and that obstacle avoidance is enforced by applying to such velocity a modulation matrix, which is properly built according to a mathematical formulation that takes into account the shape of obstacles. In more detail, the procedure returns a modulated command signal defined in an m -dimensional robot state-space as:

$$\dot{\xi} = \overline{M}(\xi)f(.), \tag{1}$$

where $\xi \in \mathbb{R}^m$ is the current robot state, $\overline{M}(\xi)$ is the modulation matrix, and $f(.)$ is the function generating the original desired velocity, generally driving the robot towards the goal state. Assuming that the control is implemented in discrete time, the next state at time $t + 1$ of the robot is computed by integrating Equation (1) using the Euler formula:

$$\xi_{t+1} = \xi_t + \dot{\xi} \delta t \tag{2}$$

where t is the current time and δt is the time step for the integration.

The simplest function f can be defined as a first-order dynamics convergent to the goal state:

$$f(.) = \xi_g - \xi, \tag{3}$$

with ξ_g the goal of the task.

The modulation matrix is constructed on the basis of the geometric features of the obstacles and their positions in the workspace. Each obstacle can be modeled as an m -dimensional ellipsoid in which, for every point on its convex surface, the following expression holds:

$$\Gamma(\tilde{\xi}) : \sum_{i=1}^m \left(\frac{\tilde{\xi}_i}{\eta_i a_i} \right)^{p_i} = 1, \tag{4}$$

where Γ is a continuous function representing the level curves of the ellipsoid, a_i are the length of the axes of the ellipsoid, p_i are customizable exponents, and $\tilde{\xi} = \xi - \xi_0$ with ξ_0 the center of the ellipsoid. The coefficients η_i are scalar values ≥ 1 called safety factors, which increase the dimension of the corresponding axis in order to take into account the safety margins that should be maintained while avoiding the obstacles.

The choice of $\Gamma(\tilde{\xi})$ as in Equation (4) provides high flexibility in modeling different shapes, while maintaining an analytic expression. This leads to a simple computation of the parameters needed for building the modulation matrix, such as the normal vector of the surface, which identifies a deflection plane for the motion. The normal vector of the surface, in fact, is the gradient of $\Gamma(\tilde{\xi})$:

$$n(\tilde{\xi}) = \nabla\Gamma(\tilde{\xi}) = \left[\frac{p_1}{\eta_1 a_1} \left(\frac{\tilde{\xi}_1}{\eta_1 a_1} \right)^{p_1-1} \dots \frac{p_m}{\eta_m a_m} \left(\frac{\tilde{\xi}_m}{\eta_m a_m} \right)^{p_m-1} \right]^T. \tag{5}$$

Then, the modulation matrix can be constructed as follows:

$$\overline{M}(\xi) = \prod_{k=1}^K M^k(\xi^k); \tag{6}$$

with the $M^k(\xi^k)$ modulation matrix of the k^{th} obstacle and K the number of the obstacles. It is important to remark that the latter is the main factor that affects the computational effort required to calculate the collision-free motion with this method. Indeed, all of its processing steps are based on analytical expressions, but repeated K times (i.e., one time for each obstacle populating the robot workspace).

The modulation matrix $M^k(\tilde{\xi}^k)$ of an obstacle can be computed on the basis of $\Gamma^k(\tilde{\xi}^k)$ and $n^k(\tilde{\xi}^k)$:

$$M^k(\tilde{\xi}^k) = E^k(\tilde{\xi}^k)D^k(\tilde{\xi}^k)E^k(\tilde{\xi}^k)^{-1}, \tag{7}$$

with:

$$E^k(\tilde{\xi}^k) = \left[n^k(\tilde{\xi}^k) \ e^{k,1}(\tilde{\xi}^k) \ \dots \ e^{k,m-1}(\tilde{\xi}^k) \right], \tag{8}$$

$$e_j^{k,i}(\tilde{\xi}^k) = \begin{cases} -\frac{\partial \Gamma(\tilde{\xi}^k)}{\partial \xi_i^k} & \text{if } j = 1 \\ \frac{\partial \Gamma(\tilde{\xi}^k)}{\partial \xi_1^k} & \text{if } j = i \neq 1, i \in 1 \dots m-1, j \in 1 \dots m-1 \\ 0 & \text{if } j \neq 1, j \neq i \end{cases}, \tag{9}$$

and:

$$D^k(\tilde{\xi}^k) = \begin{bmatrix} 1 - \frac{\omega^k}{|\Gamma(\tilde{\xi}^k)|^{\frac{1}{\rho}}} & 0 & 0 \\ 0 & 1 + \frac{\omega^k}{|\Gamma(\tilde{\xi}^k)|^{\frac{1}{\rho}}} & 0 \\ 0 & 0 & 1 + \frac{\omega^k}{|\Gamma(\tilde{\xi}^k)|^{\frac{1}{\rho}}} \end{bmatrix}, \tag{10}$$

where:

$$\omega^k(\tilde{\xi}^k) = \prod_{i=1, i \neq k}^K \frac{\Gamma^i(\tilde{\xi}^i) - 1}{\Gamma^k(\tilde{\xi}^k) - 1 + \Gamma^i(\tilde{\xi}^i) - 1}, \tag{11}$$

is a weight that determines how much the modulation matrix influences the motion in the case of multiple obstacles and ρ is a scalar ≥ 1 that determines the reactivity of the modulation.

It is worth observing that the algorithm domain \mathbb{R}^m can be the Cartesian space, as well as the robot joint space. However, the forthcoming discussion will deal only with the Cartesian space \mathbb{R}^3 , since the obstacles, which are natively solids of the 3D Cartesian space, cannot be easily mapped into the joint space of the robot.

Furthermore, this method can be extended considering the velocities of the obstacles as follows:

$$\dot{\xi} = \overline{M}(\xi)(f(\cdot) - \dot{\xi}^0) + \dot{\xi}^0, \tag{12}$$

with:

$$\dot{\xi}^0 = \sum_{k=1}^K w^k(\tilde{\xi}^k) \dot{\xi}^{o,k} \tag{13}$$

where $\dot{\xi}^{o,k}$ is the net effect of the translational and rotational velocities of the k^{th} obstacle.

Finally, the algorithm allows disabling the modulation in case the robot has already passed the obstacle (i.e., to avoid the so-called tail effect). For further details on the algorithm and its mathematical formulation, see [28,43].

4. DS-Based Obstacle Avoidance for Laparoscopic Surgical Robots

The algorithm outlined in Section 3 can be used to generate a motion for a robot under the assumptions that its end-effector (EE) can be modeled as a point or a sphere, that it can move freely in the space, and that the obstacles can be described using the analytic expression of ellipsoids. However, in laparoscopic scenarios, these assumptions are limiting. Indeed, the laparoscopic tool has a non-negligible dimension, and its motion is constrained by the trocar at the point of insertion into the abdomen of the patient. Moreover, in laparoscopic surgery, at least four tools are generally used, and when controlling one of them in an autonomous robotic platform, the other three tools are to be considered as obstacles. Such obstacles can be modeled as ellipsoids; however, their analytic expressions would require a very high value of p_i exponents to fit the shape of the tools properly; this would lead to a strong increase in the computational effort of the planner algorithm, since it

is harder to compute the distances and the closest points using these analytic expressions (in the following subsections, it will become clear why this information is needed for the planner algorithm). Therefore, a simpler model for the description of the tools has been introduced in this paper. Each tool is bounded by a capsule, which is a cylinder with a hemispheric termination (see Figure 1 and 2 on top). Figure 1 shows also a generic scenario with different kind of obstacles and introduces the notation which will be used in the following discussion. This model fits better to the shape of the laparoscopic tool, while keeping the computation of distances and nearest points very efficient, as will be described in the next subsection.

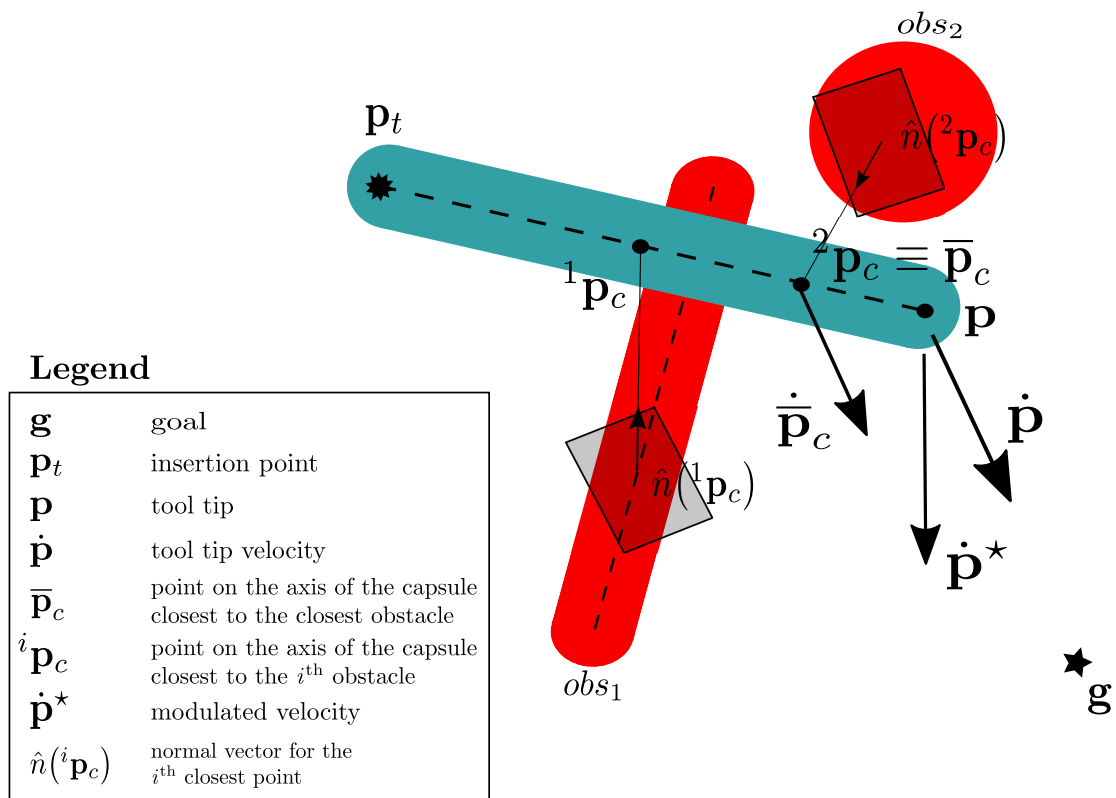


Figure 1. In blue: the controlled tool modeled as a capsule. In red: a spherical obstacle and a capsule-shaped obstacle.

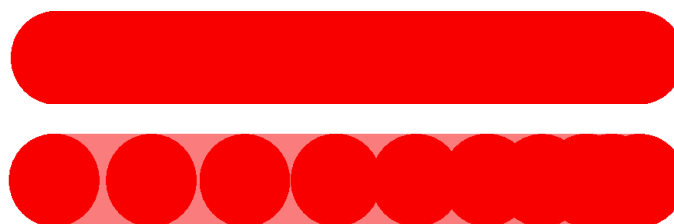


Figure 2. **Top:** a capsule. **Bottom:** a capsule sampled by its constituent overlapping spheres, with the growing density of samples going from left to right.

4.1. Computation of Distances between Solids

Computing the tool-obstacles distances and the position of their closest points is necessary to build the modulation matrix of the proposed method. Since the tool is modeled as a capsule, algorithms to compute the distance between two capsules and between a capsule and an ellipsoid are required; these can be implemented following two approaches:

1. using algorithms operating on the 3D meshes of the objects;
2. using algorithms implementing analytic solutions for the problem.

The first approach can be time consuming to execute at each control cycle, so for this application, the second approach is exploited.

The distance between two capsules $d_{1,2}$ can be computed by subtracting the radii r_1 and r_2 of the capsules from the distance between the two segments representing the axes of the capsules:

$$d_{1,2} = d_{a1,a2} - r_1 - r_2, \quad (14)$$

where $d_{a1,a2}$ is the distance between the axes of the capsules computed as explained in [44]. The closest points on the capsules \mathbf{p}_{c1} and \mathbf{p}_{c2} can be computed starting from the closest points on the segments \mathbf{p}_{a1} and \mathbf{p}_{a2} (which are found using the parameterization of the segments of [44]) by simply finding the line that passes through them and considering the radii r_1 and r_2 as follows:

$$\mathbf{p}_{c1} = \mathbf{p}_{a1} + r_1 \frac{\mathbf{p}_{a2} - \mathbf{p}_{a1}}{\|\mathbf{p}_{a2} - \mathbf{p}_{a1}\|}; \quad (15)$$

$$\mathbf{p}_{c2} = \mathbf{p}_{a2} + r_2 \frac{\mathbf{p}_{a1} - \mathbf{p}_{a2}}{\|\mathbf{p}_{a1} - \mathbf{p}_{a2}\|}. \quad (16)$$

The distances between a capsule and an ellipsoid can be analytically computed in an efficient way only if the ellipsoid is quadratic (i.e., $p_i = 2$ in Equation (4)), so in this application, only quadratic ellipsoids will be used. This distance and the closest points can be found by using geometric tools, like for example those implemented in the NASA Spice toolkit [45], combining the operations computing the distance between a line and an ellipsoid, the distance between a point and an ellipsoid, and the related closest points for both cases:

- if the line does not intersect the ellipsoid (i.e., the line is at a distance $d_{l,e} > 0$ from the ellipsoid), then the distance between the end-points of the axis of the capsule and the ellipsoid has to be computed; if both of them are greater than $d_{l,e}$, then the distance between the line and the ellipsoid is the minimum, and the relative closest points are kept; otherwise the end point with the minimum distance is the closest point of the segment, and the relatively closest point on the ellipsoid is the correct one.
- if the line intersects the ellipsoid, then one of the end points is the closest point, so for both of them, the distance is computed, and the minimum one is kept together with the relative closest points.

Then, the point on the capsule closest to the obstacle \mathbf{p}_c can be computed as follows:

$$\mathbf{p}_c = \mathbf{p}_a + r \frac{\mathbf{p}_e - \mathbf{p}_a}{\|\mathbf{p}_e - \mathbf{p}_a\|}, \quad (17)$$

where \mathbf{p}_a is the point on the axis of the capsule, \mathbf{p}_e is the point on the ellipsoid, and r is the radius of the capsule.

As a final remark, the general quadratic form of the ellipsoid can be used also to describe spheres (which in fact are ellipsoids with all the axes of the same length); however, it can be useful to treat spheres separately, since more efficient methods for solving this problem can be implemented. For example, a sphere can be considered as a degenerate capsule with an axis of null length, and then, a capsule-to-capsule distance function can be used. Alternatively, it is possible to compute the distance between a sphere and a capsule $d_{s,c}$ as follows:

$$d_{s,c} = d - r_s - r_c. \quad (18)$$

where d is the distance between a point (which is the center of the sphere) and a segment (which is the axis of the capsule) and r_s and r_c are the radii of the sphere and capsule. The distance d can be found by projecting the point with a normal projection onto the line that contains the segment and taking the distance between these two points; this distance is then compared with the distances between the point and the end points of the segment, and the minimum of the three is kept with the relative closest points. Points on capsule \mathbf{p}_c and on sphere \mathbf{p}_s can be computed again as follows:

$$\mathbf{p}_s = \mathbf{c}_s + r_s \frac{\mathbf{p}_a - \mathbf{c}_s}{\|\mathbf{p}_a - \mathbf{c}_s\|}; \quad (19)$$

$$\mathbf{p}_c = \mathbf{p}_a + r_c \frac{\mathbf{c}_s - \mathbf{p}_a}{\|\mathbf{c}_s - \mathbf{p}_a\|}, \quad (20)$$

where \mathbf{c}_s is the center of the sphere, \mathbf{p}_a is the point on the axis of the capsule, and r_s and r_c are respectively the radii of the sphere and the capsule.

4.2. Rigid Body Modulation

Since the robot is no longer modeled as a point, the modulation should take into account the whole tool as a rigid body. This introduces two problems:

1. how to manage the robot's velocity for the modulation: in fact, there is no longer a unique velocity for the EE, but each point of the tool has a different velocity (consistent with the rigid body motion);
2. how to take into account the influences of multiple obstacles properly, since different parts of the tool can be close to different obstacles.

Problem 1 was solved choosing as the velocity to be modulated the velocity $\dot{\bar{\mathbf{p}}}_c$ of the point $\bar{\mathbf{p}}_c$ on the axis of the tool closest to the closest obstacle, since this point resides in the tool area that will most likely collide with an obstacle. The dynamics used to control the EE \mathbf{p} were chosen as follows:

$$\dot{\mathbf{p}} = v_d \frac{\mathbf{g} - \mathbf{p}}{\|\mathbf{g} - \mathbf{p}\|}; \quad (21)$$

where v_d is the norm of the desired velocity and \mathbf{g} the goal of the task. Such dynamics, in case of no obstacles on the path, is always convergent to the goal of the task and determines a linear trajectory of the EE. The scalar quantity v_d can be designed, for example, to obtain an acceleration ramp at the beginning or a deceleration ramp at the end of the movement.

The velocity of the point on the axis of the tool closest to the closest obstacle $\dot{\bar{\mathbf{p}}}_c$ can be computed from Equation (21) considering the constraint of the fixed insertion point: the projection of the tip velocity on the tool direction corresponds to the translational velocity along the tool direction (the insertion/extraction velocity), then it can be subtracted from the tip velocity to obtain the tangential velocity. That component depends on the position of the considered point inside the tool, so it has to be properly scaled to obtain the tangential velocity of the desired point. Finally, the previously-computed insertion/extraction component, which is the same for all the tool points, is added to the tangential velocity to obtain $\dot{\bar{\mathbf{p}}}_c$.

A digest of this procedure is provided by the following equations:

$$\dot{\mathbf{p}}_{proj} = (\dot{\mathbf{p}} \cdot \hat{\mathbf{a}}) \frac{\hat{\mathbf{a}}}{\|\hat{\mathbf{a}}\|}; \quad (22)$$

$$\dot{\mathbf{p}}_{res} = \dot{\mathbf{p}} - \dot{\mathbf{p}}_{proj}; \quad (23)$$

$$\dot{\bar{\mathbf{p}}}_c = \frac{\|\bar{\mathbf{p}}_c - \mathbf{p}_t\|}{\|\hat{\mathbf{a}}\|} \dot{\mathbf{p}}_{res} + \dot{\mathbf{p}}_{proj}. \quad (24)$$

where $\dot{\mathbf{p}}_{proj}$ is the projected velocity, $\hat{\mathbf{a}}$ is the axis of the capsule, and $\dot{\mathbf{p}}_{res}$ is the resultant to be scaled. The velocity $\dot{\bar{\mathbf{p}}}_c$ is then modulated to enforce obstacle avoidance by means of a modulation matrix \bar{M} , built as described in Section 3:

$$\dot{\bar{\mathbf{p}}}_c^* = \bar{M} \dot{\bar{\mathbf{p}}}_c \quad (25)$$

Then, the modulated velocity $\dot{\mathbf{p}}^*$ at the end-effector tip is computed as follows:

$$\dot{\mathbf{p}}_{proj}^* = (\dot{\mathbf{p}}_c^* \cdot \hat{a}) \frac{\hat{a}}{\|\hat{a}\|}; \tag{26}$$

$$\dot{\mathbf{p}}_{res}^* = \dot{\mathbf{p}}_c^* - \dot{\mathbf{p}}_{proj}^*; \tag{27}$$

$$\dot{\mathbf{p}}^* = \frac{\|\hat{a}\|}{\|\dot{\mathbf{p}}_c^* - \dot{\mathbf{p}}_{proj}^*\|} \dot{\mathbf{p}}_{res}^* + \dot{\mathbf{p}}_{proj}^*. \tag{28}$$

where $\dot{\mathbf{p}}^*$ is the modulated projected velocity; $\dot{\mathbf{p}}_{res}^*$ is the modulated resultant to be scaled. This velocity is then integrated, and the next position of the EE tip is obtained.

If the obstacles are moving, it is possible to take into account their motion considering the velocity of the robot relative to the obstacle:

$$\dot{\mathbf{p}}_{rel} = \dot{\mathbf{p}}_c - \dot{\mathbf{p}}_{obs}, \tag{29}$$

where $\dot{\mathbf{p}}_{obs}$ is computed as shown in Equation (13), taking into account that, since the obstacles are rigid bodies as well, each obstacle velocity must be computed as the velocity of the point on the obstacle closest to the robot. For the capsules bounding the other laparoscopic tools, again Equations (22)–(24) can be used, since they maintain the insertion point constraint. The velocity vector $\dot{\mathbf{p}}_{rel}$ can be modulated by means of Equation (25), so that the modulated EE velocity relative to obstacles becomes:

$$\dot{\mathbf{p}}_{rel}^* = \dot{\mathbf{p}}_c^* + \dot{\mathbf{p}}_{obs} \tag{30}$$

Finally, $\dot{\mathbf{p}}_{rel}^*$ has to substitute $\dot{\mathbf{p}}_c^*$ into Equations (26)–(28).

As a solution to Problem 2, the influence of the obstacles was taken into account by computing the modulation matrix \overline{M} using each closest point on the tool axis and each minimum distance for each obstacle in order to maximize their influences on the tool. In fact, this will lead to using the smallest distance possible for each obstacle (so maximizing the weight and strength of their modulation matrix) and using the normal vector direction, which identifies the tangent plane of the obstacle, which will deviate the original motion in the best non-colliding direction (since it will be computed in the most critical position).

4.3. Modulation Matrix for Capsules

The modulation matrix that guarantees avoiding collisions between capsule-modeled tools can be constructed analogously to the modulation matrix of ellipsoidal-shaped obstacles as described in Section 3, taking care of computing $\Gamma(i\mathbf{p}_c)$ and the normal vector $\hat{n}(i\mathbf{p}_c)$ to the surface of the obstacle in the point on the tool axis closest to the considered capsule (i.e., $i\mathbf{p}_c$, where i is the obstacle index).

In order to compute this matrix easily, a capsule can be considered as a sequence of infinite overlapping spheres whose centers are the points laying on the axis of the capsule and whose radii are the radius of the capsule (see Figure 2): in this way, $\Gamma(i\mathbf{p}_c)$ is the value of the analytic expression of a sphere centered in the point of the axis of the capsule closest to the tool and $\hat{n}(i\mathbf{p}_c)$ is the line that contains both the point on the axis of the tool closest to the capsule and the center of the sphere and is directed towards the tool:

$$\Gamma(i\mathbf{p}_c) = \left[\sum_{k=1}^3 \left(\frac{i\mathbf{p}_{a,k} - i\mathbf{p}_{c,k}}{i r} \right)^2 \right] - 1; \tag{31}$$

$$\hat{n}(i\mathbf{p}_c) = i\mathbf{p}_c - i\mathbf{p}_a; \tag{32}$$

where $i\mathbf{p}_a$ is the point on the axis of the capsule closest to the tool and $i r$ is the radius of the obstacle.

4.4. Convergence

The modulation of the velocity of the point on the axis closest to the closest obstacle, as it is, does not guarantee the convergence of the EE to the goal of the task even if the original dynamics is convergent. Indeed, in the following cases, non-convergence may occur:

1. the desired velocity is collinear with the normal vector of the surface and directed towards the obstacle (which is a problem already known and tackled in the algorithm proposed in [28]);
2. the modulation makes the tool insert even more into the abdomen (see Figure 3 Case a): this kind of movement, although not colliding, clearly will not help the tool avoid the obstacle (since the tool is constrained by the insertion point).

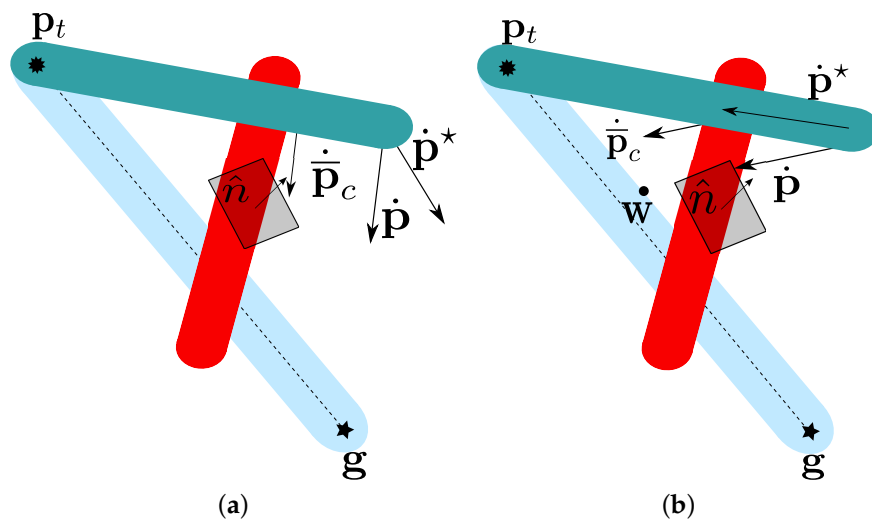


Figure 3. In red: an obstacle. In blue: the tool. In light-blue: the goal pose. Case (a) shows that in this configuration, \dot{p}^* makes the end-effector (EE) move away from the insertion point; in this case, the tool will never avoid the obstacle. Case (b) shows that placing a proper waypoint w will make \dot{p}^* be directed towards the insertion point; in this case, the tool will be able to avoid the obstacle.

In Case 1, we apply a task-specific heuristics: since the extraction of the tool is always a collision-free movement (in fact, it will make the points of the tool occupy positions that were occupied before by other points of the tools), we enforce a motion towards the insertion point. This collision-free movement should lead the tool to a position where its commanded velocity and the normal vector of the obstacle surface are no longer collinear.

Case 2 occurs mostly when the robot tries to avoid a capsule-shaped obstacle. In fact, the cylindrical part of such a capsule-shaped obstacle would never push the tool laterally, with the result of forcing the robot to be mainly inserted or extracted. A possible solution to cope with this problem consists of forcing the desired dynamics to pass near a waypoint, which has to be specifically positioned according to the relative configuration of the robot and obstacles (see Figure 3 Case b). The waypoint w is computed at each step on the basis of the current positions of the EE, of the obstacles and the goal of the task. The obstacle is surrounded by a set of points, and then, among them, the one closest to the insertion point of the tool is selected. Then, this point is projected onto the plane determined by the tool axis and the goal pose axis using as the direction of the projection a line parallel to the obstacle axis (see Figure 4); the result of this projection is the waypoint sought.

The projection can be found by simply parametrizing the line \hat{a}_c parallel to the capsule axis passing through the point r , which must be projected, and then finding the value of the parameter that determines the intersection of the line with the plane of the motion. The line is parametrized as follows:

$$\mathbf{p}_l = \mathbf{r} + u \frac{\mathbf{e}_2 - \mathbf{e}_1}{\|\mathbf{e}_2 - \mathbf{e}_1\|} \tag{33}$$

where \mathbf{e}_1 and \mathbf{e}_2 are the end points of the axis of the capsule. In order to use Equation (33), it is necessary to place the set of points aligned with \mathbf{e}_1 as in Figure 4.

The plane of the motion π is identified by its normal vector, which can be computed as follows:

$$\hat{n}_\pi = \frac{\mathbf{p} - \mathbf{p}_t}{\|\mathbf{p} - \mathbf{p}_t\|} \times \frac{\mathbf{g} - \mathbf{p}_t}{\|\mathbf{g} - \mathbf{p}_t\|} \tag{34}$$

The value u of the parameter that determines the intersection can be found with the following relation:

$$u = \frac{\hat{n}_\pi \cdot \mathbf{p}_t - \hat{n}_\pi \cdot \mathbf{p}_l}{\hat{a}_c \cdot \hat{n}_\pi} \tag{35}$$

The value is accepted only if $u \in [0, 1]$, since values outside this interval mean that the capsule is not intersecting the plane of the motion, and therefore, the waypoint is not necessary. In case u is outside of that interval, the waypoint is set to be equal to \mathbf{g} in order to have again the original dynamics; otherwise, the waypoint is computed using the value of u in Equation (33). The waypoint must be computed at each control cycle since the modulation does not guarantee that the modulated movement belongs to the original plane of the motion.

The dynamics used to control the EE is modified in order to make it point towards the waypoint, and then, when the EE reaches the waypoint (or when it is close enough), it is switched back towards to the original goal \mathbf{g} .

Since the waypoint lies on the motion plane, the corrected movement remains consistent with the original motion computed without taking into account the presence of obstacles. Furthermore, the introduction of the waypoint guarantees the convergence to the goal. Indeed, as depicted in Figures 3 and 4, when the tool tip reaches the waypoint, the dynamics will start pointing again to \mathbf{g} , but the modulation will deviate the movement under the obstacle, which is a region of free space.

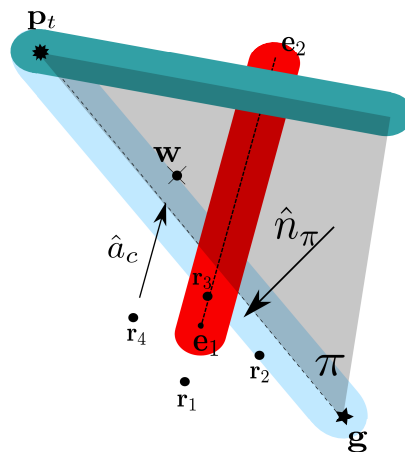


Figure 4. In red: an obstacle. In blue: the tool. In light-blue: the goal pose. In grey: the plane π determined by the actual pose of the robot and the goal pose. The figure above shows the points \mathbf{r}_i around the obstacle, for which the point within them is chosen, and how it is projected onto the plane in order to get the waypoint \mathbf{w} .

In case of multiple obstacles, the waypoint is computed as the centroid of the waypoints associated with each obstacle, using as the weight of each waypoint the inverse of the distance between the tool and the related obstacle. If a waypoint coincides with the goal of the task, it is associated with a null weight:

$$\alpha_i = \begin{cases} \frac{1}{d_i} & \text{if } \mathbf{w}_i \neq \mathbf{g} \\ 0 & \text{if } \mathbf{w}_i = \mathbf{g} \end{cases} \quad (36)$$

$$\mathbf{w} = \frac{\sum_{i=1}^N \alpha_i \mathbf{w}_i}{\sum_{i=1}^N \alpha_i} \quad (37)$$

Despite the fact that waypoints are external to the obstacle, it is possible (though unlikely) that their centroid, computed as in Equation (37), could lie inside one of the obstacles: in this case, the waypoint \mathbf{w} is not used, but the extraction heuristic is applied to bring the robot in a new position for a new waypoint computation.

As a matter of fact, Problem 2 can occur also with spherical or ellipsoidal obstacles, but this is a quite rare case, since the rounded parts of these shapes tend naturally to make the tool slide around the obstacle. However, this problem can be solved again with a waypoint approach or with heuristic methods that impose a lateral movement of the robot with respect to the obstacle, in order to unstick the tool from the critical configuration.

5. Simulations and Experiments

The algorithms described in Section 4 were first tested through simulations performed with MATLAB and then ported to C++ for testing on a realistic experimental setup.

5.1. Simulations

MATLAB simulations were designed to test the algorithm in different challenging scenarios beyond the laparoscopic one in order to assess its capabilities and limits. In all the simulations that will be described in the following sections, the EE moves from a start position to a goal position with a trapezoidal velocity profile. Moreover, for each obstacle, the safety factor η was set equal to 1.5, while reactivity ρ was set to 1.0.

5.1.1. Simulation of a Static Laparoscopic Scenario

A laparoscopic scenario was simulated using 4 capsules: one capsule bounds the robot, while the other 3 capsules bound the other laparoscopic tools. The diameters of the capsules were scaled according to the diameters of the real tools adopted in the experimental setup described in Section 5.2. In this simulation, the obstacles were static. Figure 5 shows the trajectory of the EE: it is possible to notice that the whole trajectory never collided with the obstacle, which was standing in the desired path (which was the linear one going from the starting position to the goal position).

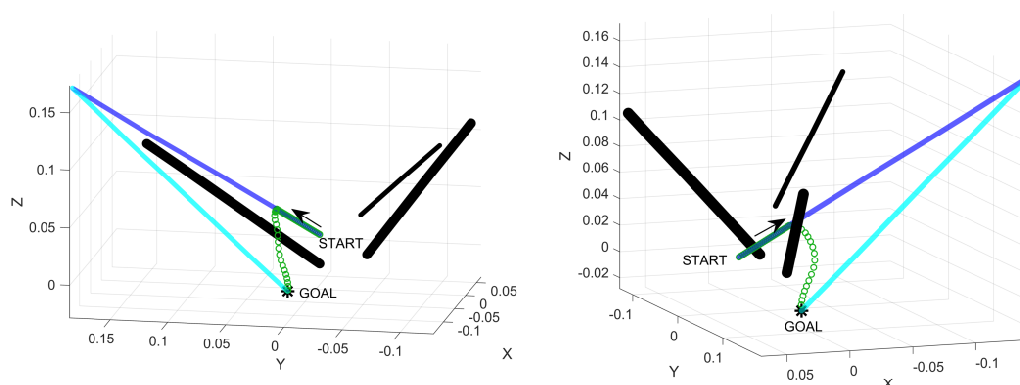


Figure 5. In blue, the starting pose of the robot, while in light-blue, the goal pose. In black, the obstacles. In green, the trajectory of the EE.

5.1.2. Simulation of a Moving Capsule-Shaped Obstacle

The algorithm was tested in a scenario where a capsule-shaped obstacle was moving and interferes with the desired movement of the robot. The movement of the obstacle was designed to imitate the movement of a laparoscopic tool that was working in a specific area, so the capsule bounding the obstacle had a fixed end-point, while its other end-point (the tip of the tool) moved on a linear trajectory. The norm of the velocity of the obstacle tip followed a slow sinusoidal profile designed to make the tip move in both directions. Figure 6 shows the evolution of the trajectories during the movements, including the complete trajectories of the EE and the obstacle in the last snapshot. The robot never collided with the obstacle.

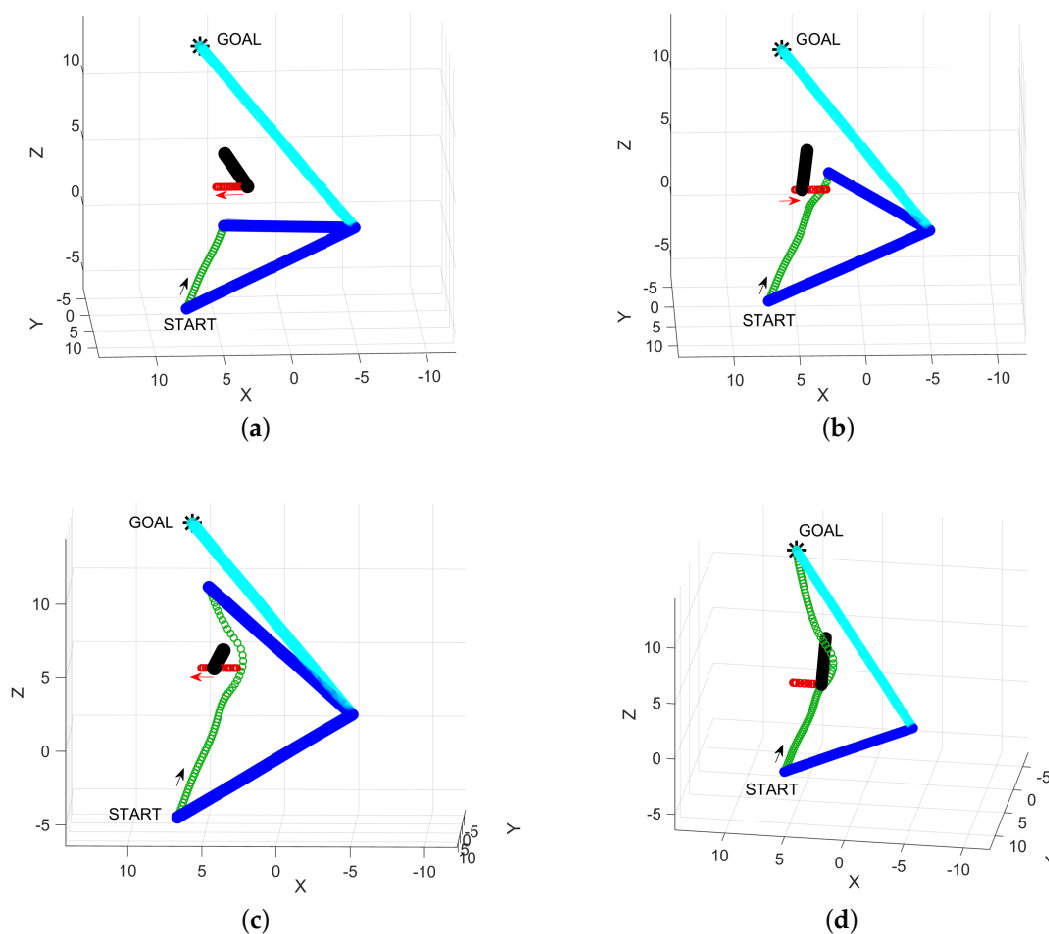


Figure 6. In blue, the starting position and the position in the snapshot of the robot, while in light-blue, the goal pose. In black, the obstacle. In red, the trajectory of the obstacle tip. In green, the real trajectory of the EE. The black arrows show the direction of the movement of the obstacle tip in that snapshot. (a) First snapshot; (b) second snapshot; (c) third snapshot; (d) final snapshot.

5.1.3. Simulation of a Dynamic Laparoscopic Scenario

Since during a laparoscopic surgery, all four tools are moving, the algorithm was also tested in a dynamic laparoscopic scenario. The movements of the tools were designed in order to generate linear trajectories for the tool tips with sinusoidal velocity profiles. Such movements were slow and with limited amplitude in order to replicate a realistic scenario. Figure 7 shows the evolution of the collision-free trajectory of the EE and of the obstacles' trajectories.

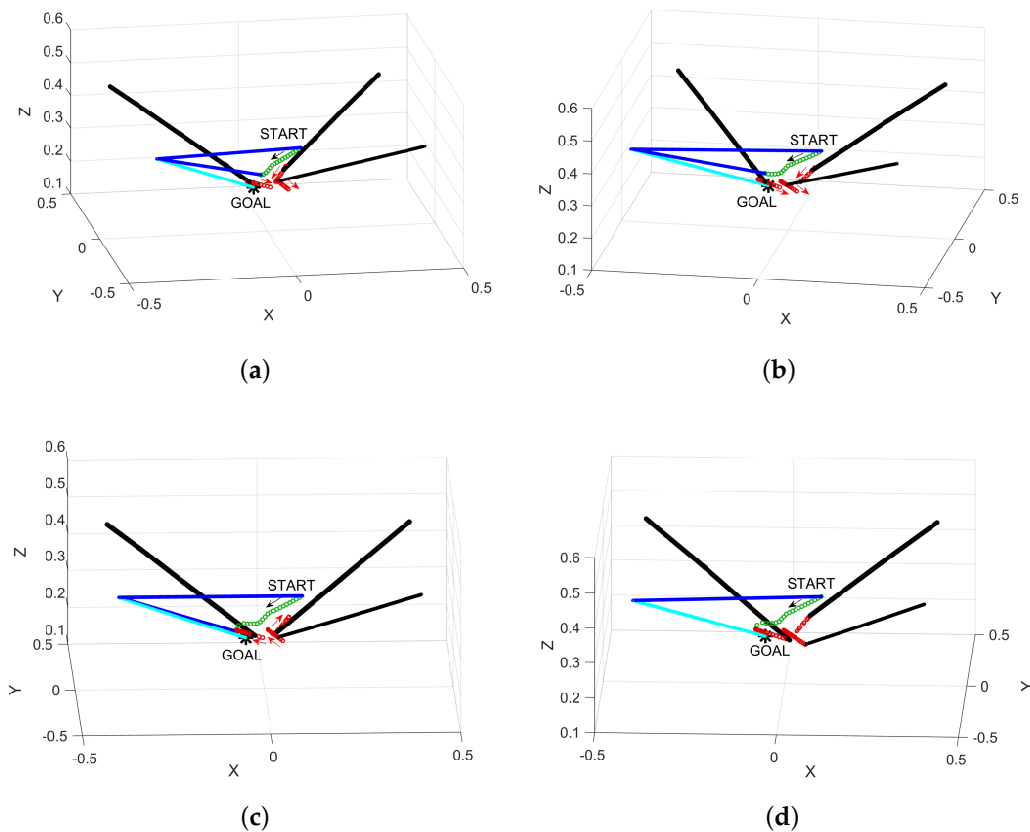


Figure 7. In blue, the starting position and the position in the snapshot of the robot, while in light-blue, the goal pose. In black, the obstacle. In red, the trajectory of the obstacle tip. In green, the real trajectory of the EE. The black arrows show the direction of the movement of the obstacle tip in that snapshot. (a) First snapshot; (b) second snapshot; (c) third snapshot; (d) final snapshot.

5.1.4. Simulation of a Moving Sphere Obstacle

In order to test the algorithm with an obstacle with a different shape and in a more challenging scenario, a simulation was performed using a moving spherical obstacle approaching the robot near the insertion point. The movement of the sphere was designed as a fast and wide linear movement with a sinusoidal velocity profile. Figure 8 shows the evolution of the trajectories of the EE and the sphere during the task. It is possible to see that the robot never collided with the sphere, even though it can be noticed that the EE made some loops with fast and wide movements in order to avoid the sphere. Indeed, in correspondence with the nearest point, the tool should at least have the same speed of the obstacle in order to avoid it, and that speed is amplified at the tip as a result of the leverage due to the insertion point constraint. This behavior may turn out to be unfeasible for the robot, because of either workspace constraints (see the subsequent discussion of Section 6.1) or limited velocity capabilities (see the subsequent discussion of Section 6.2).

A waypoint approach similar to that described in Section 4.4 can be introduced to overcome this problem. In more detail, a point was selected outside the sphere on the line that went from the center of the sphere to the insertion point, and then, it was projected on the plane of the motion using the sphere velocity as the direction of the projection, obtaining the desired waypoint. It is important to note that such a waypoint will always result in being inside the workspace of the robot, by construction, as will be discussed in Section 6.1. The experiments demonstrated that the amplitude of the movements and the velocities were strongly reduced making use of this approach, as shown in Figure 9. Indeed, the EE maintained a collision-free trajectory that did not involve the above-mentioned loops.

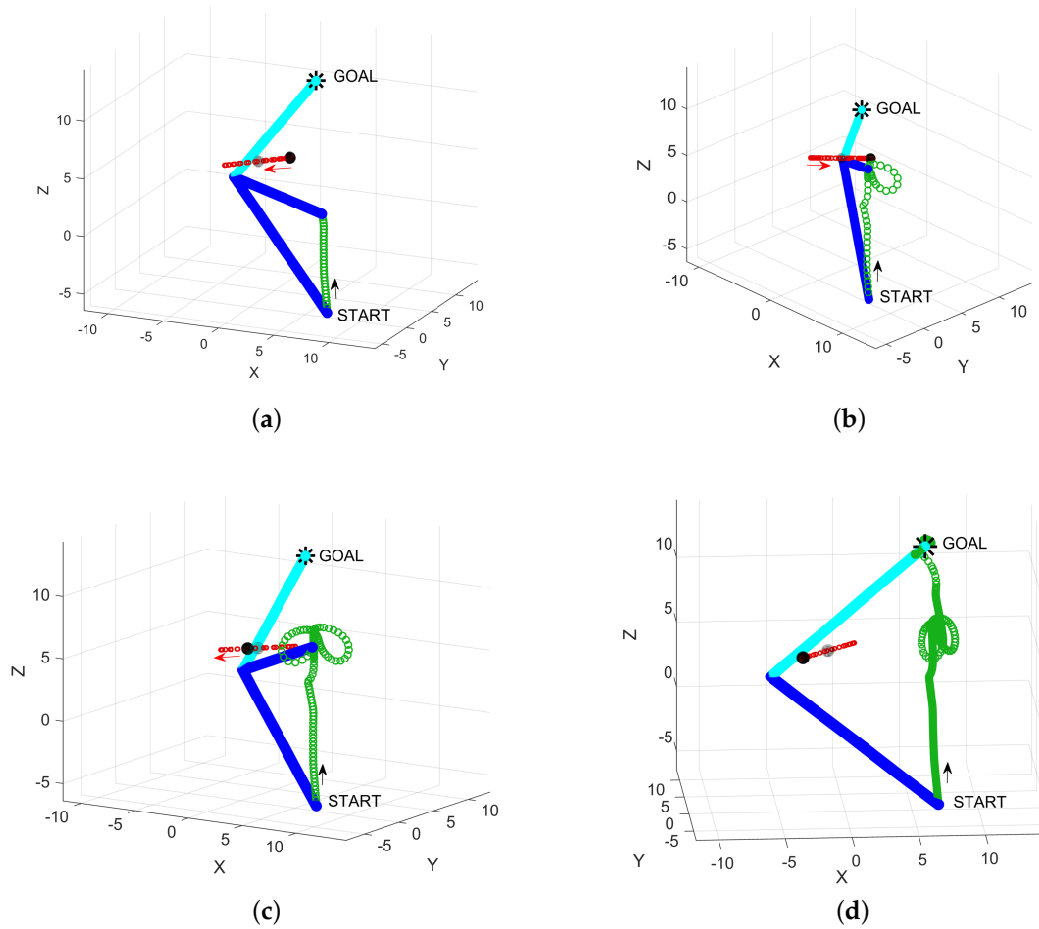


Figure 8. In blue, the starting position and the position in the snapshot of the tool, while in light-blue, the goal pose. In black, the position of the sphere in the snapshot, while in transparent black, its starting position. In red, the trajectory of the sphere. In green, the trajectory of the EE. The black arrows show the direction of the movement of the sphere in that snapshot; (a) First snapshot; (b) second snapshot; (c) third snapshot; (d) final snapshot.

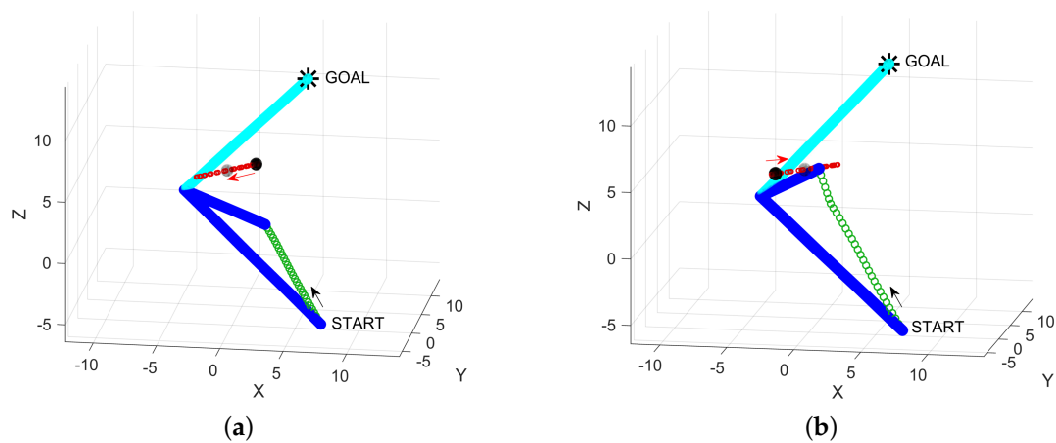


Figure 9. Cont.

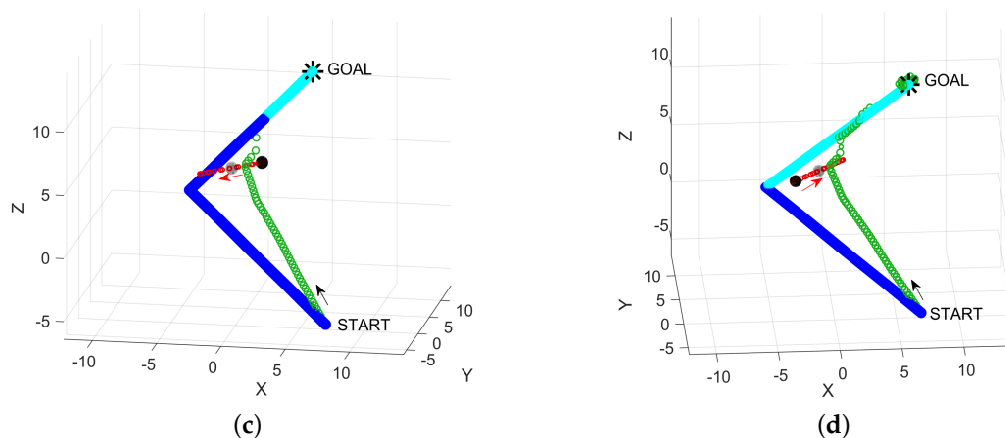


Figure 9. In blue, the starting pose of the robot, while in light-blue, the goal pose. In black, the position of the sphere in the snapshot, while in transparent black, its starting position. In green, the trajectory of the EE. The black arrows show the direction of the movement of the sphere in that snapshot. (a) First snapshot; (b) second snapshot; (c) third snapshot; (d) final snapshot.

Figure 10 shows how much the velocity of the EE was reduced with the use of a waypoint. It is worth noting that the tool stopped its motion at 4.1 s because the goal was reached, and then, at 4.3 s, it started a new movement again to avoid the approaching sphere.

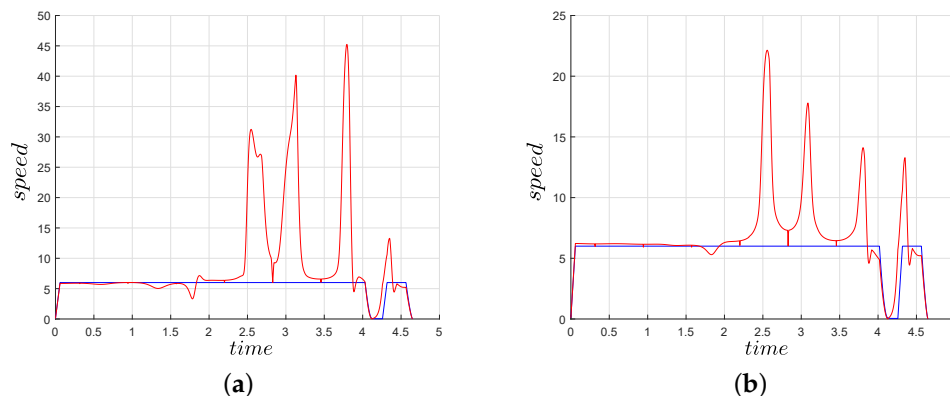


Figure 10. In blue, the norm of the desired velocity. In red, the norm of the modulated velocity. The plot (a) shows the velocity of the modulation without a waypoint for the sphere, while (b) shows the modulated velocity using a waypoint for the sphere.

5.2. Experiments on a Real Setup

The algorithm was tested on a real setup involving four laparoscopic tools mounted on four robotic arms for surgical applications. The setup was developed within the SARAS project (<https://saras-project.eu>) for research on the automation of assistive functions during prostatectomy and nephrectomy tasks. The SARAS setup was composed of the following elements:

- a DaVinci robot, which is a teleoperated surgical robot from Intuitive Surgical Inc., with two patient-side manipulators (PSM) and an endoscopic camera manipulator (ECM) on the slave side and two master tool manipulators (MTM) on the master side. The DaVinci was inserted in the whole system architecture thanks to the dVRK interface (<https://research.intusurg.com>);
- two SARAS robots, which were developed by Medineering GmbH (<http://www.medineering.de>) specifically for the objectives of the SARAS project. Each of these robots was composed of a passive 7-DOF arm for the positioning of an active 3-DOF head, which held a standard laparoscopic tool.

The overall control architecture of the SARAS robotic system was developed using the ROS (Robot Operating System, <http://www.ros.org>) environment. Therefore, the algorithm proposed in this paper was embedded in a C++ ROS node. Figure 11 shows the robots of the setup used for the experiments, while Figure 12 shows the surgical scenario seen through the dVRK endoscope. In the following, we describe experiments in which all the obstacles considered for collision-free motion generation belonged to the robotic structures of the setup (i.e., DaVinci PSM/ECM or SARAS arms). Therefore, the information about obstacles' position and velocity was computed from joint position sensors and forward kinematic calculations.

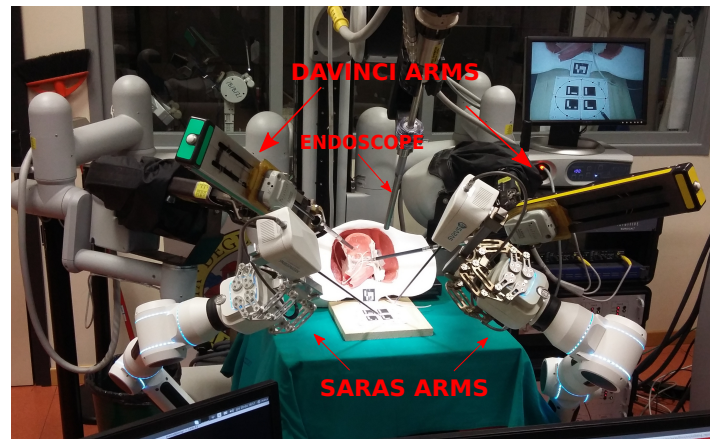


Figure 11. The setup used for the experiments. SARAS, Smart Autonomous Robotic Assistant Surgeon.

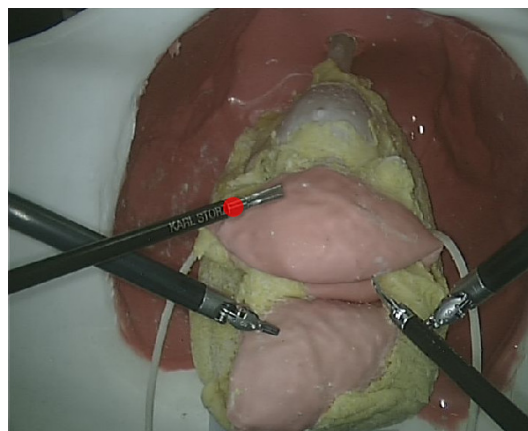


Figure 12. The surgical scenario from the point of view of the DaVinci endoscope; the red circle marks the controlled tool.

5.2.1. Static Laparoscopic Scenario

The robotic platform was configured to replicate a real static laparoscopic scenario, with the DaVinci tools and one of the SARAS tools used as obstacles and the other SARAS tool used as the commanded tool. The robot task was designed as a motion from a starting point to a goal point since this was one of the most frequent tasks that the autonomous assistant should perform.

In this experiment, the commanded velocity followed a trapezoidal profile. In order to take into account the uncertainties due to the calibration procedure (which is necessary to make the robots work correctly in the shared workspace), the safety factor η was raised from 1.5 of the simulations to 2.0, while the reactivity parameter ρ remained unchanged.

Figure 13 shows the actual collision-free trajectory performed by the SARAS robot. Note that such a trajectory is very close to the simulated trajectory of Figure 5. The experiment is also reported in the accompanying video (see Supplement Material).

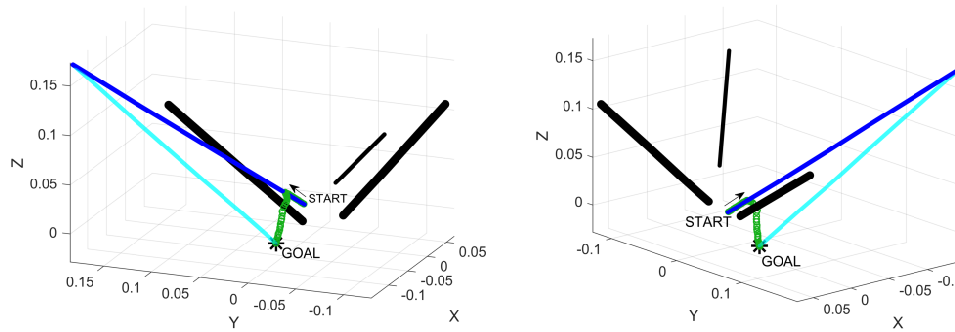


Figure 13. In blue, the starting pose of the robot, while in light-blue, the goal pose. In black, the other laparoscopic tools that are to be considered as obstacles. In green, the trajectory of the EE.

5.2.2. Dynamic Laparoscopic Scenario

An additional test was carried out on the real setup in order to try out the algorithm in a dynamic laparoscopic scenario. The test involved both the DaVinci tools and one of the SARAS tool as an obstacle, while the other SARAS tool was the controlled robot. One dVRK PSM was teleoperated using the related MTM (i.e., the haptic device handled by the human) in order to imitate the smooth movement performed by a surgeon, while the other two obstacles were kept static. The task of the robot was to travel from a starting position to a goal position on a path where the DaVinci tool was operating. Figure 14 shows the evolution of the trajectories of the moving obstacle and the controlled tool.

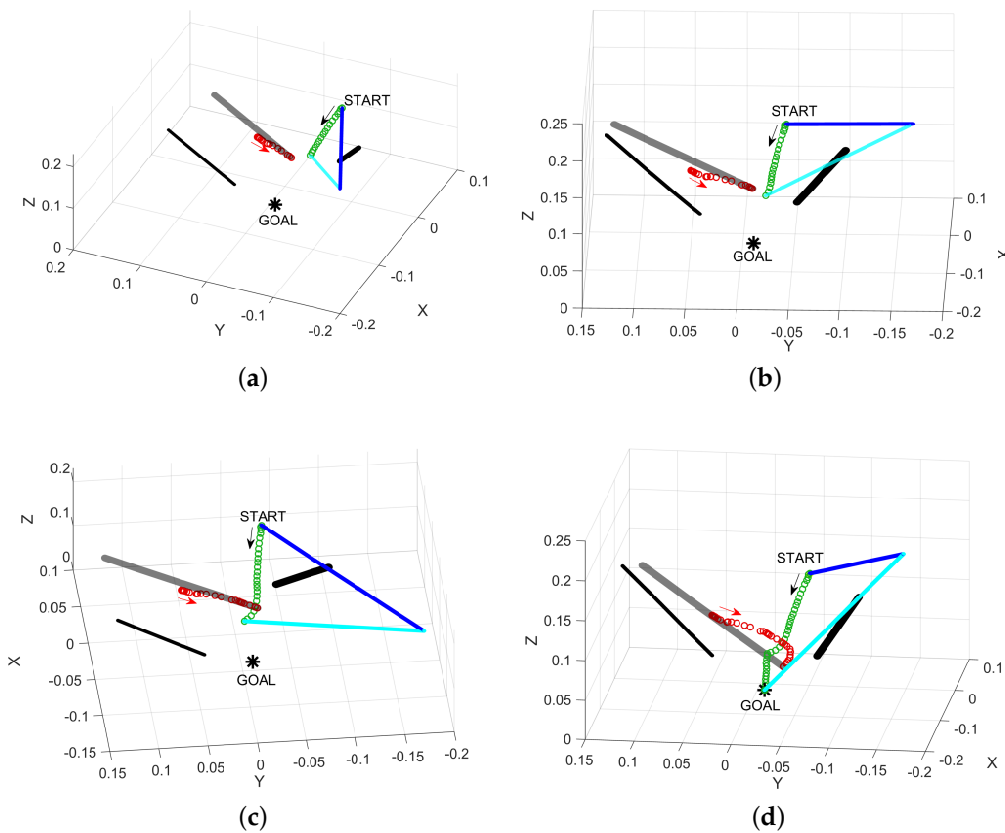


Figure 14. In blue, the starting pose of the robot, while in light-blue, the pose in the related snapshot. In black, the other laparoscopic tools, which are to be considered as obstacles (in transparent black, the moving one). In green, the trajectory of the EE. In red, the trajectory of the moving obstacle. (a) First snapshot; (b) second snapshot; (c) third snapshot; (d) final snapshot.

6. Discussion

The simulations and the experiments described in Section 5 showed that all the modulated trajectories were non-colliding, so the method worked properly in terms of dynamic obstacle avoidance. However, the method presented some limitations regarding the feasibility of the generated movement. Indeed, both the original algorithm and the extensions provided did not take into account the workspace of the robot and its velocity and acceleration constraints when modulating the original dynamics. Finally, since computing the closest points and the distances required high computational efforts, real-time algorithms must be implemented. As a matter of fact, only quadratic ellipsoids can be considered, since more complex ellipsoid formulations involve time-consuming numeric approaches.

6.1. Workspace Constraints

Even if the motion generated by the original dynamics was designed to be entirely contained in the workspace of the robot, the same was not guaranteed for the modulated motion. In order to prevent the modulated trajectory from going outside of the workspace, two different approaches can be applied (singularly or together):

1. forcing the trajectory to converge again into the workspace by modifying the original dynamics;
2. modeling the workspace limits as obstacles.

Approach 1 was basically related to the waypoint computation heuristics presented in Section 4.4, whose behavior was already shown in Section 5.1.4: in fact, the waypoint introduced led the tool into a specific region of space that was internal to the workspace. More precisely, the workspace (inside the abdomen) of a surgical robot for laparoscopic MIS was a cone with the vertex at the insertion point, height equal to the length of the laparoscopic tool, and a given opening angle. For example, the tools used in the SARAS setup had a length of 30 cm and an opening angle of about 30–40 degrees. Assuming that the goal of the desired motion (i.e., the target of the original dynamics) was reachable, the waypoint selected by the previously-described procedure and as depicted in Figure 4 will certainly be inside the conical workspace, since it lied on a triangle delimited by the insertion point, the current position of the tool tip, and the goal itself. This approach is very useful in static environments, while in dynamic environments, the modification of the original dynamics through a waypoint may not prevent the trajectory from going out of the workspace in the case of fast-approaching obstacles. However, in static environments, the problem of the workspace limitation is quite rare if the original dynamics is properly designed. Indeed, the modulation with the waypoint extension tended to generate trajectories that deviated from the original plane of the motion very little.

Approach 2, which was also more general, can be implemented modeling a limitation in the Cartesian space as an obstacle with a planar shape: this again can be inserted into the algorithm using an approach similar to the one explained in Section 4.3 for the computation of the modulation matrix of a capsule. The closest points on the plane and the tool can be computed as segment-plane distances, taking into account the radius of the tool.

6.2. Dynamic Constraints

The algorithm described in Section 3 did not natively take into account the velocity and acceleration constraints of the robot, leading to planning of a modulated motion that could not be feasible for the robot. The modulation matrix, as stated in [46], can amplify the original velocity at most by a factor of 2:

$$\|\dot{\mathbf{p}}\| \leq \frac{1}{2} \|\dot{\mathbf{p}}_{max}\|; \quad (38)$$

with $\dot{\mathbf{p}}$ the velocity to be modulated and $\|\dot{\mathbf{p}}_{max}\|$ the upper velocity bound of the robot. This relation imposes as a matter of fact an upper bound on the norm of the commanded velocity in order to satisfy the robot constraints on velocity. Even though the commanded velocity satisfied Equation (38), there were no guarantees about the feasibility of the motion in terms of acceleration; however, it is

possible to operate on the values of the reactivity parameter ρ (see Equation (10)) in order to make the modulation intervene beforehand, leading to possible reductions on the accelerations needed for the obstacle avoidance.

These arrangements are easily applicable only if the obstacles are static, while they are not so straightforward in the case of moving obstacles: in fact, the velocities of the obstacles were taken into account in the velocity $\dot{\mathbf{p}}$ that must be modulated (see Equation (13)), making Equation (38) dependent not only on the desired dynamics, but also on the obstacle's dynamics, which could not be known a priori.

The critical aspects on motion feasibility, which so far were discussed for the modulation algorithm presented in Section 3, were even more emphasized in the laparoscopic surgery case study: in fact, the velocity needed at the tool tip to avoid obstacles was strongly affected by the velocity of the obstacles and their approaching position along the tool, since the insertion point constraint forced the tip to move faster the nearer the approaching point was to the insertion point.

All of these aspects make the problem of ensuring a feasible non-colliding motion quite challenging; however, it is possible to apply task-based heuristics on the original dynamic (as shown in Section 5.1.4) in order to bring the EE in configurations where it would be possible to avoid the obstacle with lower velocities and accelerations.

7. Conclusions

The paper described a method for online trajectory generation that is able to compute collision-free motions for a robotic manipulator modeled as a capsule with a constrained end. The proposed approach was based on the modulation of the velocity of a dynamical system (DS), associated with the point on the capsule-like robot that was closest to an obstacle. A specific waypoint generation strategy was developed to cope with cluttered situations, in which the DS-based velocity modulation alone would not provide a feasible solution to avoid obstacles for the considered kind of manipulators.

The proposed motion planner was developed within the context of a research project on surgical robotics for laparoscopic applications. Therefore, many of the technical aspects described in the paper were focused on the peculiar issues of the application domain. On the other hand, the basic concepts of the proposed approach, namely the use of capsules as bounding boxes for the robot links and the real-time computation of waypoints to enforce suitable obstacle avoidance maneuvers, could be applied to collision-free trajectory generation for other kinds of robotic manipulators. In particular, the kinematic constraint imposed on a laparoscopic surgical robot by the trocar could be revisited to take into account more general kinematic models, like those of classical multi-DOF serial robots. In the future, we aim to investigate the latter aspect.

Supplementary Materials: The following are available at <http://www.mdpi.com/2079-9292/8/9/957/s1>.

Author Contributions: The author A.S. contributed to the paper by devising the method, by implementing the algorithms, by conducting the simulations and the experiments and by writing the paper. The author M.B. contributed to the paper by devising the method, by supervising and validating the simulations and the experiments and by writing and reviewing the paper. The author S.F. contributed to the paper by writing and reviewing the paper. The author G.D.R. contributed to the paper by conducting and validating the experiments. The author R.M. contributed to the paper by supervising and validating the experiments and by reviewing the paper.

Funding: This project received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 779813 (SARAS).

Acknowledgments: The authors would like to thank: Fabio Falezza, Nicola Piccinelli, and the AltairLab from the University of Verona for the technical support during the experiments; Johann Wigger, Sabine Hertle, and Medineering GmbH for the developments on the SARAS robots used in the experiments; Gernot Kronreif and ACMIT GmbH for the phantom model of the pelvic region visible in Figures 11 and 12.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Villani, V.; Pini, F.; Leali, F.; Secchi, C. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics* **2018**, *55*, 248–266. [[CrossRef](#)]
2. Vernon, D.; Vincze, M. Industrial Priorities for Cognitive Robotics. In Proceedings of the EUCognition 2016, Vienna, Austria, 8–9 December 2016; pp. 6–9.
3. Netravali, N.A.; Börner, M.; Bargar, W.L. The Use of ROBODOC in Total Hip and Knee Arthroplasty. In *Computer-Assisted Musculoskeletal Surgery: Thinking and Executing in 3D*; Ritacco, L.E., Milano, F.E., Chao, E., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 219–234.
4. Dieterich, S.; Gibbs, I. The CyberKnife in Clinical Use: Current Roles, Future Expectations. In *IMRT, IGRT, SBRT—Advances in the Treatment Planning and Delivery of Radiotherapy*; Meyer, J., Ed.; Karger AG: Basel, Switzerland, 2011; pp. 181–194.
5. Varma, T.R.K.; Eldridge, P. Use of the NeuroMate stereotactic robot in a frameless mode for functional neurosurgery. *Int. J. Med. Robot. Comput. Assist. Surg.* **2006**, *2*, 107–113. [[CrossRef](#)] [[PubMed](#)]
6. Yang, G.Z.; Cambias, J.; Cleary, K.; Daimler, E.; Drake, J.; Dupont, P.E.; Hata, N.; Kazanzides, P.; Martel, S.; Patel, R.V.; et al. Medical robotics—Regulatory, ethical, and legal considerations for increasing levels of autonomy. *Sci. Robot.* **2017**, *2*, 8638. [[CrossRef](#)]
7. Ficuciello, F.; Tamburrini, G.; Arezzo, A.; Villani, L.; Siciliano, B. Autonomy in surgical robots and its meaningful human control. *Paladyn. J. Behav. Robot.* **2019**, *10*, 30–43. [[CrossRef](#)]
8. Muradore, R.; Fiorini, P.; Akgun, G.; Barkana, D.E.; Bonfè, M.; Boriero, F.; Caprara, A.; De Rossi, G.; Dodi, R.; Elle, O.J.; et al. Development of a cognitive robotic system for simple surgical tasks. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 1–20. [[CrossRef](#)]
9. Preda, N.; Ferraguti, F.; De Rossi, G.; Secchi, C.; Muradore, R.; Fiorini, P.; Bonfè, M. A Cognitive Robot Control Architecture for Autonomous Execution of Surgical Tasks. *J. Med. Robot. Res.* **2016**, *1*. [[CrossRef](#)]
10. Ferraguti, F.; Preda, N.; Bonfè, M.; Secchi, C. Bilateral teleoperation of a dual arms surgical robot with passive virtual fixtures generation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 4223–4228. [[CrossRef](#)]
11. Preda, N.; Manurung, A.; Lamercy, O.; Gassert, R.; Bonfè, M. Motion planning for a multi-arm surgical robot using both sampling-based algorithms and motion primitives. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October, 2015; pp. 1422–1427.
12. Ferraguti, F.; Preda, N.; Manurung, A.O.; Bonfè, M.; Lamercy, O.; Gassert, R.; Muradore, R.; Fiorini, P.; Secchi, C. An Energy Tank-Based Interactive Control Architecture for Autonomous and Teleoperated Robotic Surgery. *IEEE Trans. Robot.* **2015**, *31*, 1073–1088. [[CrossRef](#)]
13. Talignani Landi, C.; Ferraguti, F.; Sabattini, L.; Secchi, C.; Bonfè, M.; Fantuzzi, C. Variable Admittance Control Preventing Undesired Oscillating Behaviors in Physical Human-Robot Interaction. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017. [[CrossRef](#)]
14. Ferraguti, F.; Talignani Landi, C.; Sabattini, L.; Bonfè, M.; Fantuzzi, C.; Secchi, C. A variable admittance control strategy for stable physical human–robot interaction. *Int. J. Robot. Res. (SAGE)* **2019**, *38*, 747–765. [[CrossRef](#)]
15. Reiley, C.; Plaku, E.; Hager, G. Motion generation of robotic surgical tasks: Learning from expert demonstrations. In Proceedings of the 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Buenos Aires, Argentina, 31 August–4 September 2010; pp. 967–970.
16. Schulman, J.; Gupta, A.; Venkatesan, S.; Tayson-Frederick, M.; Abbeel, P. A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013; pp. 4111–4117. [[CrossRef](#)]
17. Wang, H.; Wang, S.; Ding, J.; Luo, H. Suturing and tying knots assisted by a surgical robot system in laryngeal MIS. *Robotica* **2010**, *28*, 241–252. [[CrossRef](#)]
18. Nageotte, F.; Zanne, P.; Doignon, C.; de Mathelin, M. Stitching Planning in Laparoscopic Surgery: Towards Robot-assisted Suturing. *Int. J. Robot. Res.* **2009**, *28*, 1303–1321. [[CrossRef](#)]

19. Baili, Z.; Tazi, I.; Alj, Y. StapBot: An autonomous surgical suturing robot using staples. In Proceedings of the 2014 International Conference on Multimedia Computing and Systems (ICMCS), Marrakech, Morocco, 14–16 April 2014; pp. 485–489. [[CrossRef](#)]
20. Kehoe, B.; Kahn, G.; Mahler, J.; Kim, J.; Lee, A.; Lee, A.; Nakagawa, K.; Patil, S.; Boyd, W.; Abbeel, P.; et al. Autonomous multilateral debridement with the Raven surgical robot. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 1432–1439. [[CrossRef](#)]
21. Kröger, T. *On-Line Trajectory Generation in Robotic Systems—Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events*; Springer Tracts in Advanced Robotics; Springer: Berlin/Heidelberg, Germany, 2010.
22. Kuffner, J.; LaValle, S. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, USA, 24–28 April 2000; pp. 995–1001.
23. Şucan, I.A.; Moll, M.; Kavraki, L.E. The Open Motion Planning Library. *IEEE Robot. Autom. Mag.* **2012**, *19*, 72–82. [[CrossRef](#)]
24. Duan, Y.; Patil, S.; Schulman, J.; Goldberg, K.; Abbeel, P. Planning locally optimal, curvature-constrained trajectories in 3D using sequential convex optimization. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 5889–5895. [[CrossRef](#)]
25. Brock, O.; Khatib, O. Elastic Strips: A Framework for Motion Generation in Human Environments. *Int. J. Robot. Res.* **2002**, *21*, 1031–1052. [[CrossRef](#)]
26. Fraichard, T.; Delsart, V. Navigating Dynamic Environments with Trajectory Deformation. *J. Comput. Inf. Technol.* **2009**, *17*, 27–36. [[CrossRef](#)]
27. Zucker, M.; Kuffner, J.; Branicky, M. Multipartite RRTs for Rapid Replanning in Dynamic Environments. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1603–1609.
28. Khansari-Zadeh, S.M.; Billard, A. A Dynamical System Approach to Realtime Obstacle Avoidance. *Auton. Robot.* **2012**, *32*, 433–454. [[CrossRef](#)]
29. Macfarlane, S.; Croft, E. Jerk-bounded manipulator trajectory planning: Design for real-time applications. *IEEE Trans. Robot. Autom.* **2003**, *19*, 42–52. [[CrossRef](#)]
30. Dyllong, E.; Visioli, A. Planning and real-time modifications of a trajectory using spline techniques. *Robotica* **2003**, *21*, 475–482. [[CrossRef](#)]
31. Lloyd, J. *Trajectory Generation Implemented as a Non-linear Filter*; Technical Report TR-98-11; University of British Columbia, Computer Science Department: Vancouver, BC, Canada, 1998.
32. Zanasi, R.; Guarino Lo Bianco, C.; Tonielli, A. Nonlinear filters for the generation of smooth trajectories. *Automatica* **2000**, *36*, 439–448. [[CrossRef](#)]
33. Zanasi, R.; Morselli, R. Third Order Trajectory Generator Satisfying Velocity, Acceleration and Jerk Constraints. In Proceedings of the IEEE Conference on Control Applications (CCA), Glasgow, UK, 18–20 September 2002.
34. Gerelli, O.; Guarino Lo Bianco, C. Nonlinear variable structure filter for the online trajectory scaling. *IEEE Trans. Ind. Electron.* **2009**, *56*, 3921–3930. [[CrossRef](#)]
35. Guarino Lo Bianco, C.; Gerelli, O. A Discrete-Time Filter for the On-Line Generation of Trajectories with Bounded Velocity, Acceleration, and Jerk. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK, USA, 3–7 May 2010.
36. Bonfè, M.; Secchi, C. Online Smooth Trajectory Planning for Mobile Robots by Means of Nonlinear Filters. In Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 18–22 October 2010.
37. Bonfè, M.; Secchi, C.; Scioni, E. Online Trajectory Generation for Mobile Robots with Kinodynamic Constraints and Embedded Control Systems. In Proceedings of the 10th International IFAC Symposium on Robot Control (SYROCO), Dubrovnik, Croatia, 5–7 September 2012.
38. Fiorini, P.; Shiller, Z. Motion Planning in Dynamic Environments Using Velocity Obstacles. *Int. J. Robot. Res.* **1998**, *17*, 760–772. [[CrossRef](#)]

39. Piccinelli, N.; Vesentini, F.; Muradore, R. Planning with Real-Time Collision Avoidance for Cooperating Agents under Rigid Body Constraints. In Proceedings of the 2019 Design, Automation Test in Europe Conference Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 1261–1264.
40. Ijspeert, A.; Nakanishi, J.; Hoffmann, H.; Pastor, P.; Schaal, S. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Comput.* **2013**, *25*, 328–373. [[CrossRef](#)]
41. Talignani Landi, C.; Ferraguti, F.; Fantuzzi, C.; Secchi, C. A Passivity-Based Strategy for Manual Corrections in Human-Robot Coaching. *Electronics* **2019**, *8*, 320. [[CrossRef](#)]
42. Khansari-Zadeh, S.; Billard, A. Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. *IEEE Trans. Robot.* **2011**, *27*, 943–957. [[CrossRef](#)]
43. Khansari-Zadeh, S.; Billard, A. Realtime Avoidance of Fast Moving Objects: A Dynamical System-based Approach. In Proceedings of the Electronic Proc. of the Workshop on Robot Motion Planning: Online, Reactive, and in Real-Time, International Conference on Intelligent Robots and Systems (IROS), Algarve, Portugal, 7–12 October 2012.
44. Lumelsky, V.J. On fast computation of distance between line segments. *Inf. Process. Lett.* **1985**, *21*, 55–61. [[CrossRef](#)]
45. NASA. SPICE Toolkit. Available online: <https://naif.jpl.nasa.gov/naif/toolkit.html> (accessed on 28 August 2019).
46. Huber, L.; Billard, A.; Slotine, J.J. Avoidance of Convex and Concave Obstacles with Convergence ensured through Contraction. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1462–1469. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).