2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Towards Safe Autonomous Surgical Tasks with Control Barrier Functions

Cristina Iacono[1], Paolino De Risi[2], Rocco Moccia[3], Bruno Siciliano[2], Fanny Ficuciello[2]

***Abstract*— Safety is of utmost importance in surgical robots, as they operate in a complex and dynamic environment that directly impacts the patient's health based on the surgical procedure's success. One of the main difficulties in the control of surgical manipulators is in efficiently encoding dynamic nonlinear safety constraints into trajectory planning and robot control strategies. Control Barrier Functions (CBFs) represent a valuable control method for safety-critical environments such as the surgical one since its rigorous formulation aims at ensuring safety in controlled dynamic systems. This work represents a step forward in autonomous surgical task execution since it defines Lipschitz-continuous critical and autonomously prioritized dynamic constraints enforced through a CBF framework for the safe execution of surgical robotic tasks. The proposed framework, moreover, leverages Dual Quaternion (DQ) algebra for a unified and computationally efficient representation of geometric tasks and constraints, allowing for the straightforward definition of complex, time-varying surgical constraints. The safety framework is tested in simulation on the da Vinci Research Kit (dVRK) CoppeliaSim simulator and with the real dVRK robot in several surgical sub-tasks.**

I. Introduction

Safety is of paramount importance in surgical robotics, given the critical nature of the clinical environment, which means that it is not an additional feature but a primary design consideration. Safety measures are mandatory to prevent severe consequences, as harm to the patient and damage to the robot. Ensuring safe task completion is a critical aspect of robotic capability, which involves combining task accomplishment while adhering to safety conditions. Safety constraints in surgical robotics are influenced by various factors, including the robot's mechanical and control systems as well as the dynamic external environment. For instance, a robot's joints may have a restricted range of motion, and exceeding these limits could result in mechanical failure or loss of control. Similarly, the presence of anatomical obstacles, such as organs or bones and soft tissues, introduces additional layers of complexity, requiring the implementation of reliable safety mechanisms. Moreover, surgical procedures consist of a series of challenging and kinematically complex sub-tasks that need to be performed with varying priority, depending on the surgical scene [1], [2] on the current needle grasping pose and trajectory to be executed for suturing tasks [3]. This dual requirement—safety and complex subtasks management—poses significant challenges in the development of autonomous or semi-autonomous surgical robots. Given these challenges, Control Barrier Functions (CBFs) have emerged as a promising control strategy for ensuring safe operation in complex, dynamic environments such as the surgical setting. CBFs provide a mathematical framework that enables the design of controllers capable of ensuring safety by imposing real-time constraints on the system's behavior. Specifically, CBFs can be used to encode safety conditions as state-barriers that the system should not cross, preventing unsafe actions. Such barriers can conveniently be designed to enforce geometry-based constraints modeling the surgical environment, e.g., preventing surgical tools from colliding with sensitive tissues or entering forbidden areas. Moreover, they can be integrated with task-specific controllers to ensure that the robot performs surgical sub-tasks like suturing, retraction, and tissue manipulation with high precision while adhering to safety requirements. Their strength lies in the rigorous foundation and formal framework, in which safety is formulated as a constraint within an optimization problem, determining the controller's action. This paper presents a safety framework for autonomous surgical robotics that aims to enhance the execution of surgical sub-tasks while autonomously and dynamically prioritizing safety. The

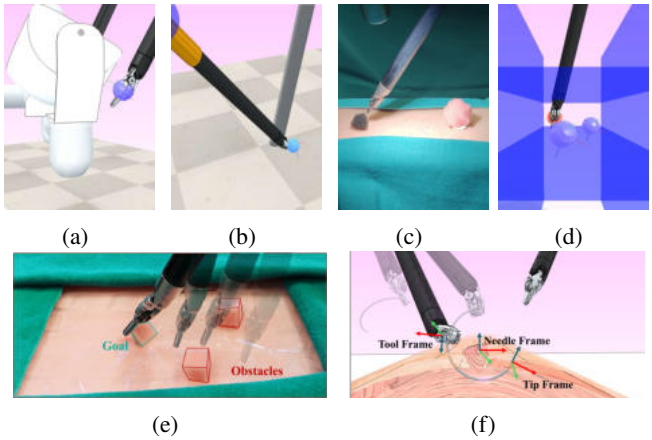

Fig. 1: Overview of the experimental scenarios. **(a)** Surgical Tool Alignment. **(b)** Vital Structure Avoidance. **(c)** Real-World Obstacle Avoidance. **(d)** Simulated Constrained Corridor. **(e)** Real-World Constrained Corridor. **(f)** Simulated Suturing Trajectory.

The research leading to these results has been partially funded by the Horizon Europe programme framework under grant agreement No. 101070596 (euROBIN), partially by European Union - ERC-2023-SyG Endotheranostics (n. 101118626) and partially by the TI-RED "Towards Intelligent Robotic Endoscopic Dissection" project, PRIN 2022- PNRR, identification code Prot. 20229N5YW8

[1] Cristina Iacono is with the C.R.E.A.T.E. Consorzio di Ricerca per l'Energia, l'Automazione e le Tecnologie dell'Elettromagnetismo, Via Claudio 21, 80125, Naples, Italy. cristina.iacono@unina.it

[2] Paolino De Risi, Bruno Siciliano and Fanny Ficuciello are with the Department of Electrical Engineering and Information Technology, University of Naples Federico II, via Claudio 21, 80125, Naples, Italy. paolino.derisi@unina.it, bruno.siciliano@unina.it, fanny.ficuciello@unina.it

[3] Rocco Moccia is with the Center of Advanced Biomaterials for Healthcare (CABHC@CRIB), Italian Institute of Technology, Largo Barsanti E. e Matteucci C. 53, 80125, Naples, Italy. rocco.moccia@iit.it

framework leverages an optimization-based approach based on CBFs ensuring task execution for safe robot motion control. A key methodological contribution of this work is the adoption of Dual Quaternion (DQ) algebra [4] to model all motion and safety constraints. Unlike traditional representations, DQs provide a compact and singularity-free formulation for both position and orientation. The contribution is threefold: i) the adoption of CBFs to the surgical field, which is a renown safety-critical environment, modeling its classical control tasks; ii) a time-varying formulation for the autonomous prioritization of tasks, guaranteeing the continuity of the control during priority switches; iii) the adoption of DQ primitives to model CBFs, allowing for a convenient formulation of task Jacobians. The proposed method is tested on the da Vinci® Research Kit (dVRK) robot simulator on CoppeliaSim [5], [6] and ultimately, on a real dVRK setup [7], targeting known, rigid environments with pre-specified obstacle and target poses.

II. Related Works

The CBF framework is a powerful tool for ensuring the safety of robotic systems, with applications ranging from autonomous cars [8] to robotic manipulators [9], [10]. Specific uses in surgical robotics include orientation control in teleoperation [11], ureteroscopy [12], and suturing [13]. The latter work compares CBFs to Model Predictive Control (MPC) [14]. MPC optimizes a sequence of control actions over a future horizon, enabling proactive collision avoidance and the generation of highly optimal trajectories—valuable features when working with dynamic environments or with deformable tissues [15]. However, MPC's main drawbacks are its high computational burden when working with non-linear constraints—posing challenges for real-time surgical systems that demand high-frequency control—and the limited safety guarantees due the prediction horizon and the accuracy of the system model. These drawbacks are instead strength points for CBFs. A foundational contribution in this domain is the multi-task CBF-Quadratic Programming (QP) framework in [10], which builds on the concepts of forward invariance [16] and set stability [17] to manage multiple tasks through a single, constrained optimization problem, where tasks are encoded with control barrier functions. In the formulation, multi-task priority is enabled by a time-varying prioritization matrix $\boldsymbol{K}(t)$, required to be Lipschitz-continuous to ensure stable priority switching. However, this approach relies on a pre-programmed, time-based prioritization schedule, limiting its applicability in unstructured environments where priorities must be established autonomously based on the evolving state of the system. While other optimization-based methods have been used in surgical robotics, for instance, using DQ algebra for constraint definition [18], [19], they can face challenges when simultaneously managing conflicting guidance and avoidance tasks [20]. The key limitation in the state-of-the-art CBF framework remains the lack of a mechanism for autonomously determining priorities based on the system's state. This paper addresses these limitations by introducing a novel, time-varying formulation for the autonomous prioritization of tasks. We propose a state-dependent function that continuously and smoothly adjusts the elements of the prioritization matrix, guaranteeing controller continuity while enabling the robot to dynamically adapt its priorities to the current context. This automatic priority switching framework extends the applicability of the proposed CBF approach to real-world scenarios where it is not possible to pre-program the constraints to prioritize, but a dynamic priority state-based scheduling of the task is needed. Furthermore, while prior work in this area has been primarily validated in simulation [10], [13], we take the critical next step of implementing and validating our autonomous prioritization framework on a physical dVRK, proving its feasibility and robustness in a real-world setting.

III. Materials and Method

A. Ensuring Safety with Control Barrier Functions

To formally guarantee the safety of a robotic system, we employ the CBF framework. Our approach models the robot's dynamics using a standard control-affine system:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}, \tag{1}$$

where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ is the system state, $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^p$ is the control input, and $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ are the Lipschitz continuous vector fields describing the system's dynamics. The core of this framework is a continuously differentiable function, $h : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$, which is designed to define a safe region of the state space, denoted by the set $C = \{\boldsymbol{x} \in \mathcal{D} \mid h(\boldsymbol{x}) \geq 0\}$. By this definition, any state is considered safe as long as $h(\boldsymbol{x})$ remains non-negative. To ensure the system never leaves this safe set — i.e. forward invariance property — we must constrain the control input $\boldsymbol{u}$ such that the value of $h(\boldsymbol{x})$ is prevented from becoming negative. This is achieved by imposing a lower bound on the time derivative of $h(\boldsymbol{x})$. For the system in (1), this safety condition is formulated as:

$$\dot{h}(\boldsymbol{x}) = L_{\boldsymbol{f}} h(\boldsymbol{x}) + L_{\boldsymbol{g}} h(\boldsymbol{x})\boldsymbol{u} \geq -\gamma(h(\boldsymbol{x}))\,, \tag{2}$$

where, $L_{(\cdot)}h(\boldsymbol{x}) = \nabla h(\boldsymbol{x}) \cdot (\cdot)$ denotes the Lie derivative, and γ is an extended class $\mathcal{K}_\infty$ function that dictates how strongly the system state is repelled from the safety boundary. Any Lipschitz continuous controller $\boldsymbol{u}(\boldsymbol{x})$ that satisfies the constraint in (2) is guaranteed to render the safe set C forward invariant, and asymptotically stable within its domain $\mathcal{D}$.

B. Task Formulation as Control Barrier Functions

For an n-DoF robotic manipulator, we define a task by a variable $\boldsymbol{\sigma}(t) \in \mathcal{T} \subseteq \mathbb{R}^m$ (e.g., end-effector pose), which is related to the robot's joint positions $\boldsymbol{q}(t) \in \mathbb{R}^n$ via the forward kinematics $\boldsymbol{\sigma}(t) = \boldsymbol{k}(\boldsymbol{q}(t))$. The differential relationship is given by the standard task Jacobian:

$$\dot{\boldsymbol{\sigma}}(t) = \boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}}(t) \tag{3}$$

where $\boldsymbol{J}_\sigma(\boldsymbol{q}) \in \mathbb{R}^{m \times n}$ is the task Jacobian. Our primary control objective is to steer the task variable $\boldsymbol{\sigma}(t)$ to track a desired trajectory $\boldsymbol{\sigma}_d(t)$. While goal-reaching is often formulated using Control Lyapunov Functions (CLFs), the CBF framework provides a powerful alternative that allows us to unify both trajectory tracking and safety avoidance within a single set-theoretic paradigm. To achieve this, we re-frame

the tracking objective itself as a safety task. The goal is to drive the task error $\boldsymbol{e}_\sigma = \boldsymbol{\sigma} - \boldsymbol{\sigma}_d(t)$ to zero. We encode this objective using a continuously differentiable function $h(\boldsymbol{\sigma}, t)$, whose zero superlevel set $C = \{\boldsymbol{\sigma} \in \mathcal{T} \mid h(\boldsymbol{\sigma}, t) \geq 0\}$ represents the desired goal. A natural choice for this function is the negative squared error:

$$h(\boldsymbol{\sigma}, t) = -\frac{1}{2}\|\boldsymbol{e}_\sigma\|^2 . \tag{4}$$

With this definition, the set C collapses to the single point set $C = \{\boldsymbol{\sigma} \in \mathcal{T} \mid \boldsymbol{\sigma} = \boldsymbol{\sigma}_d(t)\}$, where the task error is zero. Consequently, making this set forward invariant and asymptotically stable is mathematically equivalent to ensuring the tracking error converges to zero. We consider the standard kinematic model for the manipulator, where the joint velocities $\dot{\boldsymbol{q}}$ are the direct control inputs $\boldsymbol{u}$. This corresponds to a simple integrator system, where the state is $\boldsymbol{x} = \boldsymbol{q}$, with $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ and $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{I}$ in the general system model (1). To make the set C invariant, we enforce the CBF condition from (2) on the function h defined in (4), whose time derivative is:

$$\dot{h} = -\boldsymbol{e}_\sigma^T \dot{\boldsymbol{e}}_\sigma = -\boldsymbol{e}_\sigma^T(\dot{\boldsymbol{\sigma}} - \dot{\boldsymbol{\sigma}}_d) = -\boldsymbol{e}_\sigma^T(\boldsymbol{J}_\sigma \dot{\boldsymbol{q}} - \dot{\boldsymbol{\sigma}}_d) . \tag{5}$$

Substituting this into the CBF inequality (2) gives the constraint that our controller $\dot{\boldsymbol{q}}$ must satisfy to achieve the task. To find a suitable control input, we synthesize a velocity controller by solving an optimization problem. We seek the minimum-norm joint velocity vector that satisfies the task-fulfilling CBF constraint, leading to the following QP:

$$\begin{aligned} &\underset{\dot{\boldsymbol{q}}}{\text{minimize}} \quad \|\dot{\boldsymbol{q}}\|^2 \\ &\text{subject to} \quad -\boldsymbol{e}_\sigma^T \boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}} \geq -\gamma(h(\boldsymbol{\sigma}, t)) - \boldsymbol{e}_\sigma^T \dot{\boldsymbol{\sigma}}_d(t) \end{aligned} \tag{6}$$

C. Managing Multiple Conflicting Tasks

While the formulation in (6) is effective for a single objective, any practical robotic application requires the concurrent execution of multiple tasks. A naive approach would be to simply stack the CBF constraint for each of the R tasks into a single optimization problem, as shown in (7).

$$\begin{aligned} &\underset{\dot{\boldsymbol{q}}}{\text{minimize}} \quad \|\dot{\boldsymbol{q}}\|^2 \\ &\text{subject to} \quad -\boldsymbol{e}_{\sigma_r}^T(t)\ \boldsymbol{J}_{\sigma_r}(\boldsymbol{q})\dot{\boldsymbol{q}} \geq -\gamma(h_r(\boldsymbol{\sigma}_{d_r}, t)) + \\ &\qquad\qquad -\boldsymbol{e}_{\sigma_r}^T(t)\dot{\boldsymbol{\sigma}}_{d_r}(t) \qquad \forall r \in \{1, ..., R\}. \end{aligned} \tag{7}$$

However, in any non-trivial scenario, these tasks will inevitably conflict. For example, moving towards a surgical target might require getting closer to a fixture, rendering the set of constraints in (7) mutually exclusive. When such conflicts occur, the QP becomes infeasible, and no solution for the control input $\dot{\boldsymbol{q}}$ can be found. To resolve this critical issue and allow the system to gracefully manage trade-offs between conflicting objectives, a set of slack variables $\boldsymbol{\delta} \in \mathbb{R}^R$ is introduced. Each slack variable δ_r is added to its corresponding task constraint, allowing it to be violated or "relaxed" when necessary, to maintain the feasibility of the optimization. The magnitude of the slack variables is penalized in the cost function, encouraging the solver to keep all relaxations as small as possible. The relaxation mechanism is the key to establishing a task hierarchy. By imposing relative constraints on the slack variables, we can dictate which tasks are allowed to be relaxed more than others. This hierarchy is encoded in a set of linear constraints, $\boldsymbol{K}\boldsymbol{\delta} \geq \boldsymbol{0}$, where the matrix $\boldsymbol{K}$ is the prioritization matrix. Each row in $\boldsymbol{K}$ defines a pairwise priority relationship. For instance, to enforce that task T_i has a higher priority than task T_j, we impose the constraint $\eta\delta_i \leq \delta_j$ for some $\eta > 1$, ensuring that the constraint for task j will be significantly relaxed with respect to the constraint for task i. This leads to our final, robust formulation for prioritized multi-task execution:

$$\begin{aligned} &\underset{\dot{\boldsymbol{q}}, \boldsymbol{\delta}}{\text{minimize}} \quad \|\dot{\boldsymbol{q}}\|^2 + l\|\boldsymbol{\delta}\|^2 \\ &\text{subject to} \quad -\boldsymbol{e}_{\sigma_r}^T \boldsymbol{J}_{\sigma_r}\dot{\boldsymbol{q}} \geq -\gamma(h_r) - \boldsymbol{e}_{\sigma_r}^T \dot{\boldsymbol{\sigma}}_{d_r} - \delta_r, \quad \forall r \\ &\qquad\qquad \boldsymbol{K}\boldsymbol{\delta} \geq \boldsymbol{0} \end{aligned} \tag{8}$$

where $l > 0$ is a scalar weight. Importantly, any task can be made a hard constraint (i.e., safety-critical) by enforcing its corresponding slack variable $\delta_r = 0$. For example, to enforce the priority stack $T_2 \succ T_1 \succ T_4 \succ T_3$ (where $\succ$ denotes "has higher priority than") among four tasks, we can construct the prioritization matrix $\boldsymbol{K}$ shown in (9):

$$\boldsymbol{K} = \begin{bmatrix} 1/\eta & -1 & 0 & 0 \\ -1 & 0 & 1/\eta & 0 \\ -1 & 0 & 0 & 1/\eta \\ 0 & -1 & 1/\eta & 0 \\ 0 & -1 & 0 & 1/\eta \\ 0 & 0 & 1/\eta & -1 \end{bmatrix} . \tag{9}$$

For all tasks in this work, we employ a specific class $\mathcal{K}_\infty$ function $\gamma(h)$ defined by the piecewise formulation in (10).

$$\gamma(h(\boldsymbol{\sigma}_d, t) = \begin{cases} \alpha\sqrt{h(\boldsymbol{\sigma}_d, t)} & \text{if} \quad h(\boldsymbol{\sigma}_d, t) \geq 0 \\ -\alpha\sqrt{-h(\boldsymbol{\sigma}_d, t)} & \text{if} \quad h(\boldsymbol{\sigma}_d, t) < 0 \end{cases} . \tag{10}$$

This choice provides a lower bound on the exponential evolution of h and exhibits faster convergence when the state is close to the goal boundary ($\boldsymbol{\sigma} \approx \boldsymbol{\sigma}_d$) compared to a simpler linear function like $\gamma(h) = \alpha h$.

D. Autonomous Dynamic Priorities

In this work, a time-varying formulation for the prioritization matrix $\boldsymbol{K}(t)$ is introduced. Unlike [10], the priority between tasks is switched automatically according to the values of the functions h_r. Considering the formulation in (9), in order to get a priority swap between two tasks, the non-zero elements of a row of $\boldsymbol{K}$ have to be switched. According to [10], $\boldsymbol{K}(t)$ has to be Lipschitz continuous over time to guarantee the continuity of the controller during the priority switching and thus, it cannot be instantaneous. Therefore, in this study a solution to the definition of a Lipschitz continuous prioritization matrix $\boldsymbol{K}(t)$ is provided, allowing to swap priorities among tasks automatically. First, a prioritization variable $\beta_{i,j}(t) = |h_i(\boldsymbol{\sigma}_{d_i}, t)| - |h_j(\boldsymbol{\sigma}_{d_j}, t)|$ is defined as the difference between two h functions, each associated to its task T_i and T_j. Then, $\boldsymbol{K}(t)$ is defined as:

$$\boldsymbol{K}(t) = \boldsymbol{K}_o + \boldsymbol{\Lambda}(t), \tag{11}$$

where $\boldsymbol{K}_o$ is the prioritization matrix for fixed priorities as employed in (9), $\boldsymbol{\Lambda}(t)$ is a time varying matrix whose elements

TABLE I: CBF notation overview

h	CBF function	γ	bounding $\mathcal{K}^{\infty}$ function
δ	slack variable	l	slack scalar weight
$\boldsymbol{K}$	prioritization matrix	$\boldsymbol{\Lambda}$	switching matrix
η	prioritization ratio	β	prioritization variable
α	gain of the bounding function	s	switching slope

$\lambda_{i,j}(t)$ are defined as follows:

$$\lambda_{i,j}(t) = \begin{cases} 0 & \text{if } k_{i,j} = 0 \\ (1+\frac{1}{\eta})\frac{1}{1+e^{-s\beta_{i,j}(t)}} & \text{if } k_{i,j} = -1 \\ -(1+\frac{1}{\eta})\frac{1}{1+e^{-s\beta_{i,j}(t)}} & \text{if } k_{i,j} = 1/\eta \end{cases}, \quad (12)$$

where s is a parameter representing the slope of the sigmoid functions. This approach prioritizes task T_i over T_j based on $\beta_{i,j}(t)$, which acts as a composite metric of the proximity of the system state to the boundaries of the safe space $\partial\mathcal{C}_i$ and $\partial\mathcal{C}_j$. If $\beta_{i,j}(t) > 0$, the system state is closer to $\partial\mathcal{C}_j$ respect to $\partial\mathcal{C}_i$, and vice versa. When $\beta_{i,j}(t)$ decreases to zero, it indicates that the state space is approaching the boundary of the safe set associated with the lower-priority task, prompting a switch between tasks. Each row of $\boldsymbol{K}(t)$ is controlled by its autonomously varying unique pair of sigmoid functions set by $\lambda_{i,j}(t)$, effectively managing the priority between all task pairs depending on $\beta_{i_j}(t)$, from -1 to $1/\eta$, and vice versa. By adding $\boldsymbol{\Lambda}(t)$ to the initial $\boldsymbol{K}_o$ we obtain a continuous update of the elements in (9), and given the Lipschitz continuity of the sigmoid functions, the continuity of the control output during priority switching is guaranteed. Furthermore, by incorporating the redundant constraints in (9), the feasibility of the overall constraints is maintained. Note that, without loss of generality, we assumed that the $\boldsymbol{K}_o$ matrix is assigned by prioritizing the tasks in ascending order of $|h_r(\sigma_{d_r}, 0)|$ functions.

E. Geometric Constraint Definition with Dual Quaternions

A key methodological contribution of this work is the adoption of DQ algebra to model all motion and safety constraints. Unlike traditional representations, DQs provide a compact and singularity-free formulation for both position and orientation. A unit DQ provides a compact, 8-parameter representation of a rigid body transform, unifying both rotation and translation into a single algebraic entity. This unified structure is highly advantageous over traditional 4×4 homogeneous transformation matrices, as it is not only more memory-efficient but also allows for the composition of rigid transforms through a computationally cheaper multiplication operation. Crucially, DQs are free from the rotational singularities (i.e., gimbal lock) that can affect other representations, ensuring robustness. This property, combined with their computational efficiency, makes them an ideal tool for defining the geometry-based CBFs used in our real-time, safety-critical surgical application. A unit dual quaternion, denoted as $\hat{\boldsymbol{q}}$, is composed of two quaternions: a real part $\mathbf{q}_r$ that represents rotation, and a dual part $\mathbf{q}_d$ that encodes translation. It is defined as:

$$\hat{\mathbf{q}} = \mathbf{q}_r + \varepsilon\mathbf{q}_d \quad (13)$$

where ε is the dual unit, which has the defining property $\varepsilon^2 = 0$. For a rigid transform described by a rotation and a translation, the two components are constructed as follows. Regarding the Real Part (Rotation) we define $\mathbf{q}_r$ as standard unit quaternion representing the rotation. For a rotation of angle θ around a unit axis $\mathbf{n} = \left[n_x, n_y, n_z\right]^T$ it is given by $\mathbf{q}_r = \left(\cos\left(\frac{\theta}{2}\right), \mathbf{n}\sin\left(\frac{\theta}{2}\right)\right)$. For the Dual Part (Translation) we define $\mathbf{q}_d$ derived from the rotation quaternion $\mathbf{q}_r$ and the translation vector $\mathbf{t} = [t_x, t_y, t_z]^T$. First, the translation vector is embedded in a pure quaternion $\mathbf{t}_q = (0, \mathbf{t})$. The dual part is then calculated as:

$$\mathbf{q}_d = \frac{1}{2}\mathbf{t}_q\mathbf{q}_r \quad (14)$$

This formulation encapsulates the full 6-DOF rigid body pose, and it is ideal for the efficient definition of geometric constraints and of task Jacobians [18], thereby allowing for an elegant, mathematical definition of geometry-based CBFs for various primitives (spheres, cylinders, planes), which are critical for modeling surgical environments. In this framework, each task-specific formulation $h(\sigma)$ depends on the geometry of the task, and it has been formulated employing the DQ algebra. Table II summarizes the CBF definitions for the geometric primitives used in our experiments, along with the associated task Jacobians. In all cases, $\boldsymbol{p}_e \in DQ$, represents the DQ pose of the robot's end-effector, and vec4$(\cdot)$ extracts the vector part of a DQ's translational component. Regarding (23), expressing a CBF for a cylindrical obstacle avoidance, as in [18] and [19], the cylinder central axis is described as a Plücker line $\underline{\boldsymbol{l}} \in \mathbb{Q}_p$, expressed as a unit DQ: $\underline{\boldsymbol{l}} = \boldsymbol{l} + \varepsilon\boldsymbol{m}$, where $\boldsymbol{l}$ is the unit norm line direction and $\boldsymbol{m} = \boldsymbol{p}_l \times \boldsymbol{l}$ is the line moment with $\boldsymbol{p}_l$ as the cylinder end-point on the line. Regarding the Jacobian column, we define $\boldsymbol{J}_{pos}$ and $\boldsymbol{J}_{ori}$ as the translational and orientation components of the robot Jacobian $\boldsymbol{J}(\boldsymbol{q})$, respectively.

IV. Experimental Validation

The experimental validation involves experiments on the dVRK CoppeliaSim physics simulator, and on the real setup, as shown in Fig. 1. Each Patient Side Manipulator (PSM) of the dVRK features seven Degree of Freedoms (DOFs) in the RRPRRRR configuration, where "R" represents revolute joints and "P" denotes a prismatic joint. The dVRK arms are equipped with 8 *mm Large Needle Driver (400006)* surgical tool from Intuitive, Inc. The following section presents a series of experiments that replicate realistic tasks involved in minimally invasive robotic surgery, as complex surgical tasks can be decomposed into multiple low-granularity-level gestures devoid of medical sense [21]. The target and obstacles positions are collected by manually placing the end effector of the manipulator on them. These positions, relative to the PSM reference frame, are then input into the framework for real-world experiments to compute the CBFs. As a result, no external perception systems or calibration procedures have been used for the experimental validation. One significant challenge in the surgical context lies in reaching postures proximal to the joint limits, due to the high level of dexterity required for surgical tasks, e.g., needle insertion. This limitation is especially critical when executing tasks automatically, as the completion of the task must be aligned with the mechanical constraints of the robot.

TABLE II: Control Barrier Function Definitions for Geometric Constraints, using DQ

Constraint	CBF Definition: $h(\sigma)$		Parameters	Description	Jacobian	
Joint Limit (joint i)	$\frac{(q_{M,i}-q_i)(q_i-q_{m,i})}{(q_{M,i}-q_{m,i})^2}$	(15)	$q_{m,i}$; $q_{M,i}$	Min limit; Max limit	$\frac{q_{M,i}+q_{m,i}-2q_i}{(q_{M,i}-q_{m,i})^2}$	(16)
Target Position	$-\frac{1}{2}\left(-\|\text{vec4}(\mathbf{p}_e-\mathbf{p}_d)\|^2\right)$	(17)	$\mathbf{p}_d$	Target pose;	$\boldsymbol{J}_{pos}$	(18)
Target Orientation	$-\frac{1}{2}\|\|\boldsymbol{r}_d * \boldsymbol{r}_e^{-1}\|\|^2$	(19)	$\boldsymbol{r}_d$	Target quaternion	$\boldsymbol{J}_{ori}$	(20)
Spherical Obstacle	$\frac{(\|\|\text{vec}_4(\boldsymbol{p}_e-\boldsymbol{p}_o)\|\|^2-\Delta_s^2)}{2}$	(21)	$\boldsymbol{p}_o$; Δ_s	Obstacle pose; Margin	$\boldsymbol{J}_{pos}$	(22)
Cylindrical Obstacle	$\frac{\left(\|\|\text{vec}_4(\boldsymbol{p_e}\times\boldsymbol{l}-\boldsymbol{m})^T\|\|^2-\Delta_s^2\right)}{2}$	(23)	$\mathbf{l}, \mathbf{m}$; Δ_s	Plücker Axis; Margin	$\text{vec}_4(\boldsymbol{p_e}\times\boldsymbol{l}-\boldsymbol{m})^T\boldsymbol{S}(\boldsymbol{l})^T\boldsymbol{J}_{pos}$	(24)
Planar Constraint	$\frac{\left\|\frac{(\boldsymbol{p_e}\boldsymbol{n}_{\pi_i})+(\boldsymbol{n}_{\pi_i}\boldsymbol{p_e})}{2}-\boldsymbol{d}_{\pi_i}\right\|^2}{2}$	(25)	$\mathbf{n}_\pi$; d_π	Normal vector; Distance	$(\text{vec}_4(\boldsymbol{n}_{\pi_i}))^T\boldsymbol{J}_{pos}$	(26)

TABLE III: dVRK joint limits.

q_{lim}	q_1	q_2	q_3	q_4	q_5	q_6
q_M	$\pi/3$	$\pi/2$	0.25	π	$\pi/2$	$\pi/2$
q_m	$-\pi/3$	$-\pi/2$	0.05	$-\pi$	$-\pi/2$	$-\pi/2$

Therefore, in every experiment we conducted, avoiding joint limits was encoded as a critical constraint. The joint limits of the robotic arm are shown in Table III. Kinematic model and DQ algebra are defined using DQ Robotic C++ library [20]. The optimization problem is solved using IBM ILOG CPLEX. The optimization is solved in real-time at 1000 Hz frequency. For each experiment presented in the following, the set of euristically chosen parameters is reported.

A. ***Sim*** - *Surgical Tool/Needle-Driver Alignment - Dynamic*

This test simulates the fundamental surgical task of aligning a tool, such as a needle driver or electrocautery instrument, with a target pose (T_{pos}, T_{ori}), while avoiding joint limits ($T_{lim_1}, ..., T_{lim_6}$), as shown in Fig. 1a. For each of the joint limits task, we employ a CBF $h_{lim,i}$ and the relative Jacobian, as in (15) and (16). The task T_{pos} can be defined through the h_{pos}-function as in (17) and (18). The task T_{ori} is defined considering the CBF in (19) and the Jacobian in (20). The stack of tasks, excluding the safety-critical ones, is defined as $T_{pos} \succ T_{ori}$. Figure 2 presents the error values of the considered task, showing that the position and orientation CBFs h_{pos} and h_{ori} are driven to 0 executing the position and orientation tasks. The bottom plot illustrates the behavior of our dynamic priority system. In the initial phase ($t < 0.8s$), the orientation task is sacrificed (indicated by its high slack value, δ) to allow the system to complete the position task as quickly as possible. At $t = 0.8\,s$, as soon as the h_{pos} is lower than a threshold of 0.002, an autonomous priority hand-off occurs: the system begins to enforce the orientation constraint, as shown by the rapid decrease in its slack variable. This ensures that the orientation also converges to its desired value. The employed simulation parameters are: $\alpha_{lim} = 0.1$, $\alpha_{pos} = 0.1$, $\alpha_{ori} = 0.5$, $\eta = 500$, $l = 100$, $s = 8 \cdot 10^3$.

B. ***Sim*** - *Surgical Tool/ Vital Structure Avoidance - Static*

In this section, we simulate avoiding a cylindrical structure, such as an endoscope, while maneuvering the surgical tool towards its target (T_{pos}). However, the same constraints hold for elongated anatomical structures like major blood vessels or nerve bundles, modeled as a cylinder primitive (T_l). Figure 1b shows the proposed test. The tasks T_{pos} and $T_{lim_1}, ..., T_{lim_6}$ are defined as in (17), (18) and (15), (16) respectively. The task T_l is obtained by preventing the tool tip from colliding with the cylindrical body and its associated CBF and Jacobian are defined as in (23) and (24). The stack of tasks, excluding the safety-critical ones, is defined as $T_l \succ T_{pos}$. Figure 3 presents the h functions values of the considered task, showing that the robot app safety reaches the target, while the distance from the cylindrical shaft remains positive, ensuring no collision between the robot and the obstacle. The employed simulation parameters are: $\alpha_{lim} = 0.1$, $\alpha_{pos} = 0.1$, $\alpha_l = 0.5$, $\eta = 500$, $l = 100$, $\Delta_s = 0.007\,m$.

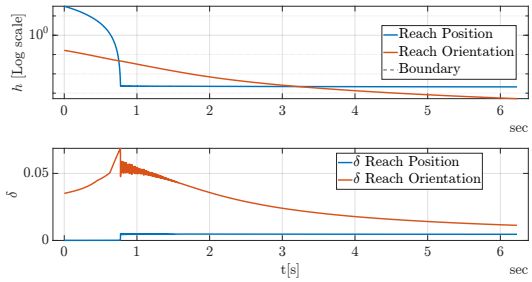

Fig. 2: Reach Target Test: h functions evolution, together with slack variables. The function h_{pos} is computed by considering distance in cm. For improved clarity the h functions are represented in log scale, requiring the plot of $|h_{pos}|$ as, given (17), h_{pos} is negative definite.

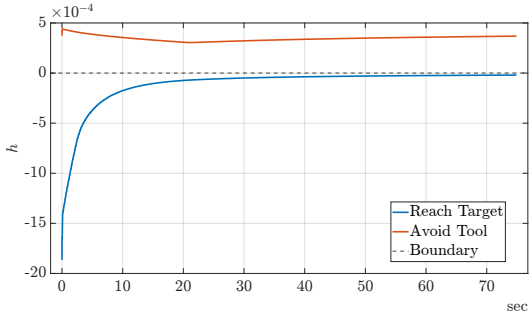

Fig. 3: Obstacle Avoidance Test: Position error and obstacle distance during the test.The functions h are computed by considering distances in m.

C. ***Real*** - *Reach Surgical site in Position and Avoid Tissues - Dynamic*

This test has been performed on the real setup of the dVRK. As in the previous tests, the robot tool tip has to reach a target in position (T_{pos}) and avoid a known spherical obstacle

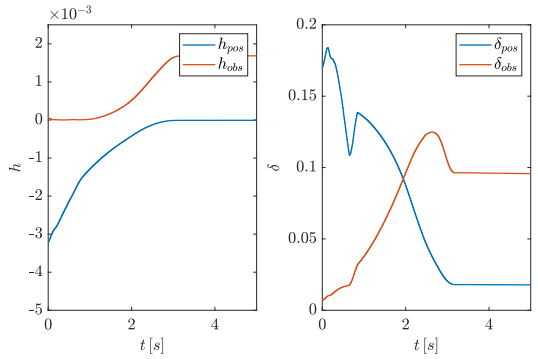

Fig. 4: Reach Target and Obstacle Avoidance Test on the real setup: Left - h functions evolution, Right - slack variables $\boldsymbol{\delta}$ evolution.The functions h are computed by considering distances in m.

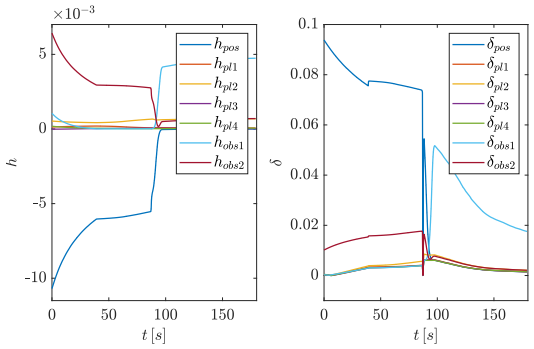

Fig. 5: Reach Target in Position while Avoiding Obstacles and Planes - Dynamic Priorities: Left - h functions evolution, Right - slack variables $\boldsymbol{\delta}$ evolution. The functions h are computed by considering distances in m.

placed in the scene (T_{obs}). Meanwhile, the joint values have to be maintained in their joint limits ($T_{lim_1}, ..., T_{lim_6}$). As in the previous tests, task T_{pos} is determined by the CBF function and Jacobian defined in (17) and (18), while the task T_{obs} is obtained as in (21) and (22). Figure 1c presents the test. The initial stack of tasks, excluding the safety-critical tasks, is $T_{obs} \succ T_{pos}$, as the PSM's initial position is closer to the obstacle than to the target. While subsequent tests explore more complex scenarios, this experiment was specifically designed to provide an unambiguous demonstration of the dynamic priority switching mechanism. By reducing the non-critical tasks to just two, we can isolate and visualize the core behavior of our framework. This transition is perfectly captured in Fig.4: the intersection of the two slack variables (δ_{obs} and δ_{pos}) respresents the system autonomously and smoothly switching priority from safety (T_{obs}) to task completion (T_{pos}), as the parameter $\beta_{obs,pos}$ gets negative and the sigmoids perform the priority switching. The employed parameters for the test are: $\alpha_{lim} = 0.1$, $\alpha_{pos} = 0.1$, $\alpha_l = 0.5$, $\eta = 500$, $l = 100$, $\Delta_s = 0.007\, m$.

*D. **Sym-Real** Operating in a Constrained Surgical Corridor - Fixed/Dynamic*

This experiment mimics a complex surgical scene where the tool must remain within a safe corridor defined by multiple anatomical structures (two spherical obstacles, T_{obs_1}, T_{obs_2}) and fixtures (four planes ($T_{pl_1}, ..., T_{pl_4}$)) to avoid collateral damage, while avoiding joint limits ($T_{lim_1}, ..., T_{lim_6}$) and reaching the surgical site (T_{pos}). For each of the four planes, the constraint is defined with the CBF function and Jacobian in (25) and (26). The scene is shown in Fig. 1d. The stack of tasks, excluding the safety-critical ones, is defined as $T_{obs_1} - T_{pl_1} - T_{pl_2} - T_{pl_3} - T_{pl_4} - T_{obs_2} - T_{pos}$. In this test, the proposed autonomous dynamic priority framework is implemented to enable priority switching during execution. As shown in Fig. 5, multiple intersections of the slack variables indicate priority switching between tasks. Specifically, at $t = 92\, s$, the intersection of δ_{obs1} and δ_{obs2} occurs when the first obstacle is overcome and $\beta_{obs1,obs2}$ approaches zero, while the PSM approaches the second obstacle. A similar transition takes place at $t = 94$, when δ_{obs2} and δ_{pos} switch, allowing the PSM to proceed toward the target. During the whole experiment, the dynamic switching priority framework ensures compliance with the four planar constraints. The employed parameters for the test are: $\alpha_{lim} = \alpha_{pos} = \alpha_{pl,i} = \alpha_{obs,i} = 1.0$, $\eta = 100$, $l = 100$, $s = 500$, $\Delta_s = 0.001$. To rigorously validate the core contribution of this work—robustness through autonomous prioritization—we designed a similar real-world experiment to directly compare our method against the conventional fixed-priority CBF controller, demonstrating that our dynamic approach provides superior robustness and task success. The experimental scenario is shown in Fig. 1e. The robot tool tip must reach a goal pose while navigating a narrow corridor defined by two spherical obstacles and two parallel planes at 3.8 cm distance (one representing the operating tissue). For a rigorous and fair comparison, we conducted 10 trials for both the fixed-priority baseline and our proposed dynamic method. In each trial, the robot started from a different initial pose to test robustness against varying conditions, while all controller gains were kept identical ($\alpha_{lim} = \alpha_{pos} = \alpha_{pl} = \alpha_{obs} = 1.0$, $\eta = 10$, $l = 10^3$, $s = 10^4$.). For both approaches, the initial priority stack was determined by the initial values of the h-functions, consistently making the target-reaching task one of the lowest priorities at the start. The results for the fixed-priority baseline, averaged over 10 trials, reveal a critical failure mode. As shown in Fig. 6a, the controller exhibits an **80% task failure rate**. The mean value and the standard deviation of the h_{pos} goal-reaching function, proximal to -0.03 and 0.01, respectively, prove that the tool never reaches the safety boundary, indicating that the target is not reached. This failure arises because the use of a static priority stack leads to a state of constraint-induced paralysis. When the tool is positioned between two or more obstacles and faces multiple high-priority avoidance constraints, the QP, which follows the fixed hierarchy, determines optimal solutions, resulting in the robot's motion stopping almost completely. Consequently, the experiment creates a "trap" for the tool, where all safety constraints are met at the expense of the goal-reaching task. In contrast, our autonomous prioritization framework demonstrates a **100% success rate** across all trials. The averaged results in Fig. 6b show that while all safety constraints ($h_{obs1}, h_{obs2}, h_{pl1}, h_{pl2}$) are strictly satisfied, with minimum values of $\bar{h}_{obs1} = 0.0006$, $\bar{h}_{obs2} = 0.003$, $\bar{h}_{pl1} = 0.006$, $\bar{h}_{pl2} = 0.07$, the goal-reaching task (h_{pos}) successfully

converges to zero in every trial. The underlying mechanism for this success is visualized in a representative trial in Fig. 6c. The plot reveals how the system navigates the clutter first and, once the obstacles are cleared (around $t = 4.5s$), the slack variable for the reaching task (δ_{Reach}) drops precipitously as the system autonomously elevates it to the highest priority, successfully completing the task. Moreover, the dynamic prioritization is completely autonomous and state-dependent, allowing the framework to robustly adapt to different initial states and environmental conditions, proving the superior robustness of our approach when compared to pre-defined time-based switches as those in [10].

E. Proof of concept: Suturing Task

This test serves as a proof of concept, demonstrating how a complex surgical procedure can be broken down into a sequence of low-level geometric primitive sub-tasks executed by our framework. Specifically, we focus on the task of inserting a needle into tissue. The experiment is conducted in a simulation within a simplified environment, as illustrated in Fig. 1f. This simulation does not take into account the forces generated by interactions with the environment. However, it allows us to accurately determine the transform between the tip needle frame and the tool frame, which could potentially be estimated through vision in future real-world applications.

The entire experiment is designed to control the center of the needle rather than the end effector pose. This allows us to divide the task into the following three sequential steps: **i)** *Needle Alignment (NA)*: guidance to the target pose located midway between the insertion and extraction points, **ii)-iii)** *Rotation 1 (R1)* and *Rotation 2 (R2)*: needle's tip insertion into the tissue, via two subsequent 90 deg rotations about the needle center. The *NA* step is simulated with a combination of the following tasks: reaching a target pose with the needle center ($T_{pos,ori}$), while performing a multi-body plane avoidance ($T_{pl,needle}, T_{pl,tool}$), where the plane is placed tangent to the external surface of the tissue. Meanwhile, the joint values have to be maintained in their joint limits ($T_{lim_1}, ..., T_{lim_6}$). During this step, $T_{lim_1}, ..., T_{lim_6}$, $T_{pl,n}$ and $T_{pl,ee}$ are performed as critical tasks, while ($T_{pos,ori}$) is non critical. The *R1* and *R2* steps, executed sequentially, are designed to simulate the needle's penetration and rotation through the tissue. This rotational motion, while keeping the needle center position, is controlled as a non-critical task ($T_{pos,ori}$). The critical constraints for both steps are the joint limits ($T_{lim_1}, ..., T_{lim_6}$) and a go-to-plane constraint ($T_{pl,tip}$), which is defined orthogonally to the tissue's surface. This go-to-plane task ensures the needle path remains constrained to a specific plane, mimicking the correct insertion angle and trajectory. The employed parameters for the test are: $\alpha_{lim,i} = \alpha_{pl,i} = 1.0$, $\alpha_{pos,ori,i} = 8.0$, $\alpha_{lim,ii} = \alpha_{lim,iii} = 0.1$, $\alpha_{pl,ii} = \alpha_{pl,iii} = 1.0$, $\alpha_{pos,ori,ii} = \alpha_{pos,ori,iii} = 5.0$, $\eta = 10$, $l = 100$, $s = 1000$, $\Delta_s = 0.0$. The framework executes these steps autonomously, transitioning from one step to the next, when a threshold is reached by the function h. Figure 7 shows the evolution of the tasks h-function and the slack variable. Since each step has only one non-critical task, no prioritization method is adopted.

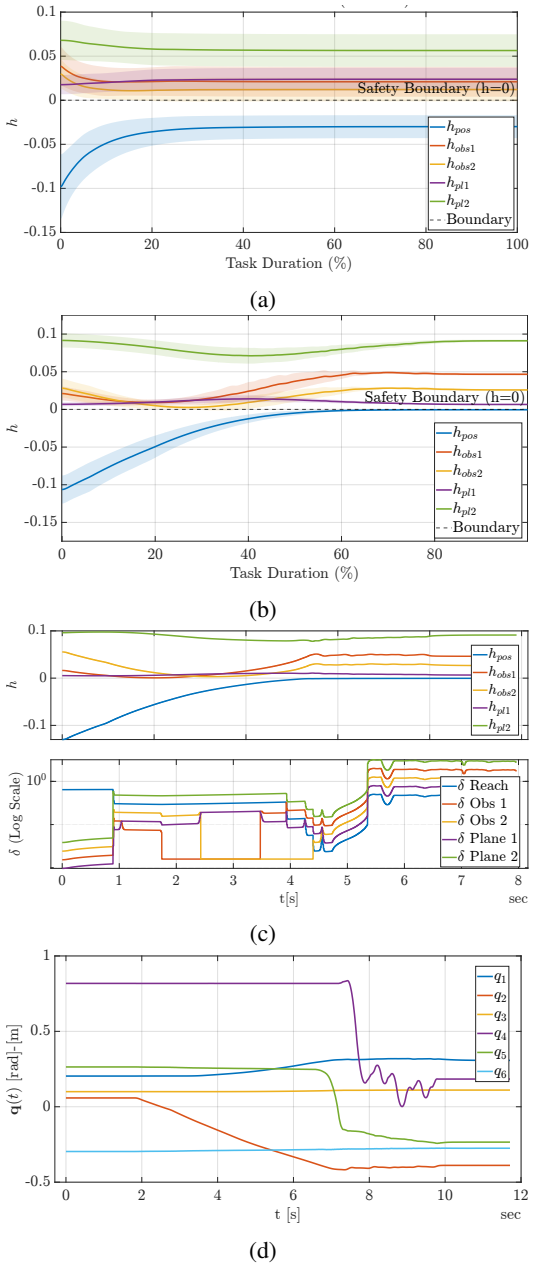

Fig. 6: Comparative analysis: (a) Averaged results for the **Fixed-Priority** test over 10 trials. The h_{pos} function fails to converge. (b) Averaged results for the **Dynamic Priority** test over 10 trials. The function h_{pos} successfully converges to zero. (c) Representative trial: h-functions and slack variables δ. The functions h are computed by considering distances in dm.(d) Representative trial: joint states evolution.

V. Conclusion

This paper presents a method for multiple-task execution on surgical robots that ensures the accomplishment of tasks involving both target-reaching and joint limit and obstacle avoidance tasks. The method uses CBFs for task definition and execution, ensuring that the safe set is asymptotically stable and forward invariant. DQ algebra allows a simple representation of the surgical scene with time-varying constraints that reflect the environment geometry. The introduced autonomous dynamic task prioritization ensures the

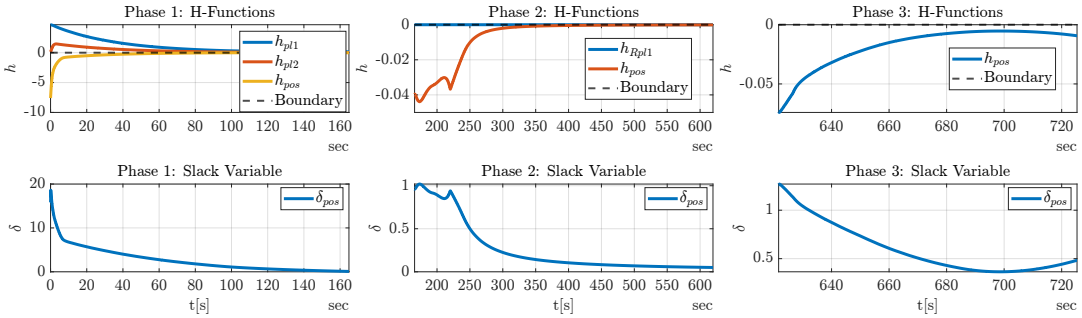

Fig. 7: Proof-of-concept needle insertion: sequence of autonomously executed sub-tasks. The framework completes three phases: (1) Needle Alignment while respecting planar constraints, followed by two sequential (2-3) Rotations for insertion. The evolution of the h-functions (top) and the slack variable (bottom) confirms the robust execution of the entire sequence.

possibility of switching the priority of the defined tasks during execution. The proposed strategy is evaluated on different experiments on the dVRK robot both in simulation, utilizing the CoppeliaSim physics simulator, and on the real dVRK robotic setup. Future works will employ the proposed framework on a specific surgical procedure, showing the benefits of multiple-task execution in real surgical scenarios. Crucially, our method advances the field by removing key barriers that have limited the deployment of autonomous systems in complex surgical scenarios. First, it overcomes the critical failure mode of "constraint-induced paralysis", where conventional fixed-priority controllers become immobilized by conflicting safety constraints, as demonstrated in our comparative analysis. Our dynamic approach breaks this deadlock, ensuring that task progress is always made when it is safe to do so. Future work will build on this robust foundation by incorporating full dynamics for soft-tissue interaction and vision-based perception to fully automate task definition, bringing adaptable autonomous surgery closer to clinical reality.

References

[1] C. Iacono, M. Caianiello, S. Bartiromo, A. Smaldone, and F. Ficuciello, "Design and validation of a multimodal dataset of robot-assisted suturing gestures based on kinematic and force information," in *2024 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. IEEE, 2024, pp. 98–103.

[2] M. Caianiello, C. Iacono, A. Imperato, and F. Ficuciello, "Exploring the use of deep reinforcement learning algorithms for wound-approaching trajectories in robot-assisted minimally invasive surgery," in *2023 21st International Conference on Advanced Robotics (ICAR)*. IEEE, 2023, pp. 285–290.

[3] M. Sallam, G. A. Fontanelli, A. Gallo, R. La Rocca, A. D. S. Sardo, N. Longo, and F. Ficuciello, "Prototype realization of a human hand-inspired needle driver for robotic-assisted surgery," *IEEE Transactions on Medical Robotics and Bionics*, 2023.

[4] J. G. Farias, E. De Pieri, and D. Martins, "A review on the applications of dual quaternions," *Machines*, vol. 12, no. 6, 2024. [Online]. Available: https://www.mdpi.com/2075-1702/12/6/402

[5] G. Fontanelli, M. Selvaggio, M. Ferro, F. Ficuciello, M. Vendittelli, B. Siciliano *et al.*, "Portable dvrk: An augmented v-rep simulator of the da vinci research kit," *Acta Polytechnica Hungarica*, vol. 16, no. 8, pp. 79–98, 2019.

[6] M. Ferro, A. Mirante, F. Ficuciello, and M. Vendittelli, "A coppeliasim dynamic simulator for the da vinci research kit," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 129–136, 2022.

[7] O. F. Argin, R. Moccia, C. Iacono, and F. Ficuciello, "da vinci research kit patient side manipulator dynamic model using augmented lagrangian particle swarm optimization," *IEEE Transactions on Medical Robotics and Bionics*, 2024.

[8] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[9] F. Ferraguti, C. Talignani Landi, A. Singletary, H. C. Lin, A. Ames, C. Secchi, and M. Bonfè, "Safety and Efficiency in Robotics: The Control Barrier Functions Approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 139–151, 2022.

[10] G. Notomista, S. Mayya, M. Selvaggio, M. Santos, and C. Secchi, "A Set-Theoretic Approach to Multi-Task Execution and Prioritization," *Proc. IEEE International Conference on Robotics and Automation*, 2020.

[11] Z. Shen, M. Saveriano, F. J. Abu-Dakka, and S. Haddadin, "Safe execution of learned orientation skills with conic control barrier functions," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 13 376–13 382.

[12] X. Wang, Y. Fang, N. Yu, and J. Han, "Barrier function-based adaptive control of twisted tendon-sheath actuated system with unknown rigid–flexible coupling for robotic ureteroscopy," *IEEE/ASME Transactions on Mechatronics*, 2023.

[13] M. Caianiello, M. Ricci, A. Smaldone, S. Hussain, C. Iacono, and F. Ficuciello, "Optimizing safety and efficiency in the suturing task: A comparison of model predictive control and control barrier function framework," in *2024 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. IEEE, 2024, pp. 104–109.

[14] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109821001175

[15] M. Dominici and R. Cortesão, "Model predictive control architectures with force feedback for robotic-assisted beating heart surgery," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2276–2282.

[16] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," *Proc. in IEEE Conference on Decision and Control*, pp. 6271–6278, 2014.

[17] X. D. Xu, J. W. Tabuada, P. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *Proc. in IEEE Conference on Decision and Control*, vol. 48, no. 27, pp. 54–61, 2015.

[18] M. M. Marinho, B. V. Adorno, k. Harada, and M. Mitsuishi, "Active constraints using vector field inequalities for surgical robots," *Proc. IEEE International Conference on Robotics and Automation*, pp. 5364–5371, 2018.

[19] ——, "Dynamic Active Constraints for Surgical Robots using Vector Field Inequalities," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1166–1185, 2019.

[20] B. V. Adorno and M. M. Marinho, "DQ Robotics: a Library for Robot Modeling and Control," *IEEE Robotics & Automation Magazine*, pp. 1–15, 2020.

[21] O. Dergachyova, "Knowledge-based support for surgical workflow analysis and recognition," Ph.D. dissertation, Université de Rennes, 2017.