2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Demonstration to Adaptation: A User-Guided Framework for Sequential and Real-Time Planning

Kuanqi Cai[1], Riddhiman Laha[1], Yuhe Gong[2], Lingyun Chen[1], Liding Zhang[1], Luis F.C. Figueredo[1,2], and Sami Haddadin[1]

***Abstract*— This paper introduces a comprehensive user-guided planning framework designed for robots operating in dynamic, human-centered environments – where the ability to execute sequential tasks flexibly and adaptively is paramount. Our planner enables robots to (i) encode object-centric constraints and user preferences via multiple demonstrations, (ii) transfer geometric features and implicit relaxations to novel scenarios while reacting to unforeseen events, and (iii) adapt to changing task conditions in real-time, including the real-time replanning and tracking of moving targets. Our approach relies on C^1 screw linear interpolation, which generates smooth paths satisfying the underlying geometric constraints that characterize the task. The prescribed path is combined with a hierarchical quadratic programming-based controller which explores the user demonstrations's stochastic variability to relax task constraints while ensuring real-time whole-body collision avoidance. Our framework continuously checks for dynamic changes in task targets, ensuring appropriate planning or control actions, and tending to the prescribed screw path. This comprehensive approach is deployed in different task conditions which are available at https://youtu.be/F0cMr1n1D9k.**

I. Introduction

Future robots operating in fast-changing human-centered environments will need the ability to perform a sequence of elaborate manipulation tasks in a flexible, reactive, and adaptive fashion, while tending to changing user-preferences and geometric and dynamic constraints in its surroundings. This challenging scenario requires robotic systems that are able to solve the following key problems: (i) To plan sequential manipulation tasks involving multiple object-centric constraints, including user's preferences and flexibilities on how to perform a task; (ii) To transfer such task information and flexibilities to new scenarios while reacting to unforeseen events such as dynamic obstacles; (iii) To adapt to changing task conditions by real-time replanning or tracking of moving goals. In this paper, we propose a planner that addresses all the aforementioned challenges in a unified framework.

Designing elaborate sequential tasks, as shown in Fig. 1, requires experienced roboticists to conceptualize and preprogram the robot. Yet, the generalization of this task across the workspace (different instances) is extremely hard [1], [2]. A more convenient approach is to allow end-users to teach such tasks, together with their preferences, to the robots. One of the most well-know strategies is learning from demonstrations (LfD) [3] which is often acquired through teleoperation, kinesthetic or passive observation. A typical way to encode the trajectory information is through motion primitives, e.g., Dynamic Movement Primitives (DMPs) [4], [5]. Taking a probabilistic approach such as Stable Estimator of Dynamical Systems (SEDS), [6], or Probabilistic Movement Primitives (ProMPs), [7], improves generalization by exploring the variance information from multiple demonstrations. Such stochastic confidence interval can be explored to improve task execution – as we previously showed in [8]. Notwithstanding, the convergence of methods such as ProMPs is still only guaranteed within the demonstration region. It is also worth mentioning that tasks often embedded in joint-space [9], [10], which largely limits its generalization to different kinematics or scenarios. Most importantly, these methods are predominantly offline, necessitating parameter tuning and offering limited autonomy in dynamic environments. This considerably limits their application in real-world scenarios where reactiveness to dynamic obstacles is critical.

In our previous work, [1], [11], we proposed an alternative solution encoding implicit geometric constraints underlying task features from a single demonstration. Our method supported real-time reaction to dynamic obstacles by means of modifying key points in the original trajectory, akin to elastic strip methods [12]. Nonetheless, this approach is still restricted to known scenarios such as (i) and (ii), with a single demonstration and stationary targets and goals.

Real-time dynamically changing targets have not been a primary focus within LfD, in general. Typically, challenges as (iii) are addressed through visual servoing techniques, which excel in tracking moving targets by integrating vision with (often) decoupled positions and orientations [13], [14]. In [15], a proper pose coupling has been introduced therefore improving performance in sensitive tasks such as grasping – yet without fully capturing the intrinsic task constraints or preferences. Despite the excel results, particularly in highly dynamic scenarios, existing methods lack the ability to encode geometric features or constraints underlying a task execution which also need to be satisfied during execution.

In this work, we propose a novel user-guided planning framework that captures implicit geometric features underlying object-centric task constraints from human demonstrations. We extend our previous results [1] by supporting sequential and multiple-task demonstrations and making full use of the stochasticity of those. Our method capitalizes on these multiple demonstrations to capture the inherent geometric flexibility, allowing for varied paths that improve specific instances of a given task. We further extend our approach with a real-time vector-field-based reactive planner based on our previous results but extended for whole-body collision avoidance deployed within hierarchical quadratic programming (HQP)-based controller for set-based trajectory tracking. Finally, our framework also addresses task adaption and tracking of moving targets. Particularly, we offer a geometric and task-consistent way to replan subtasks, blend task modifications reactively, and track moving targets, while respecting the prescribed constraints underlying different stages of the task. This comprehensive approach to task representation, execution, and adaptation addresses the complexities of real-world robot operation, offering a significant

[1]The authors are with the Technical University of Munich (TUM), Munich Institute of Robotics and Machine Intelligence (MIRMI), Germany. This work was funded by the Lighthouse Initiative Geriatronics by StMWi Bayern (Project X, 5140951), LongLeif GaPa gGmbH (Project Y, 5140953). The authors would like to thank the KI.FABRIK Bayern (grant DIK0249) project.

[2]The authors are with the Sch. of Computer Science, University of Nottingham, UK. L. Figueredo is also an Associated Fellow at TUM.

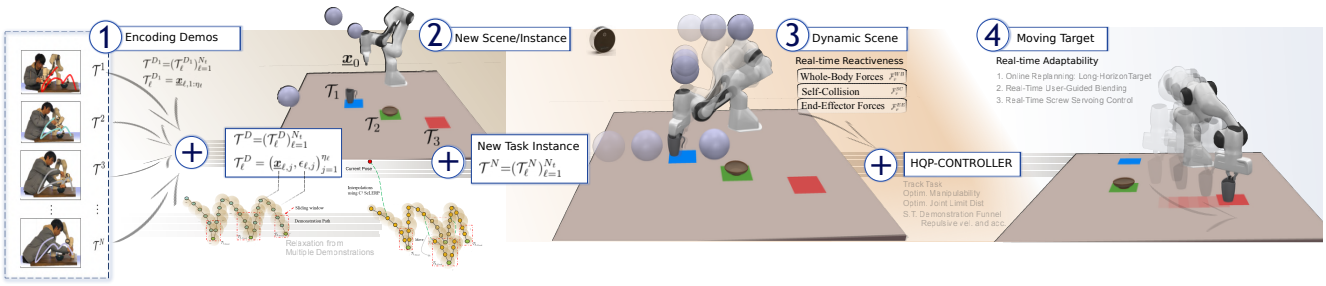


Fig. 1. Overview of proposed user-guided adaptive and relaxed planning and control approach. A sequential manipulation task $\mathcal{T}^{\mathcal{D}}$, composed by a sequence of subtasks $\mathcal{T}_\ell^{\mathcal{D}}$, $\ell{=}\{1,2,\ldots,N_t\}$, is demonstrated by the user as in ①. Each ith-demo encodes task-object-centric rigid body transformations $\mathcal{T}_\ell^{D_i} = \underline{\boldsymbol{x}}_{\ell,1:\eta_\ell}$, where η_ℓ is the number of keypoints in the subtask ℓ. Whenever multiple task demo are available, we can also extract their interval range $\epsilon_{\ell,1:\eta_\ell}$, e.g., 2σ geometric distance variance, for each η_ℓ element of each subtask ℓ. From a new task instance, $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$, the proposed Sequential and Relaxed Task-Space Imitation Algorithm (Section III) converts the $\mathcal{T}^{\mathcal{D}}$ into the imitated path with a funnel resulting in object-centric flexibilities. In addition, it also blends the novel initial pose into those new paths through C^1-ScLERP. An HQP-based constraint reactive controller (Section V) optimizes for manipulability and joint-limit avoidance while holding the funnel constraint and performing whole-body collision avoidance. Finally, the adaptive behavior of our approach addresses changes in subtasks, by 1. replanning in long-horizon subtasks, 2. real-time C^1-ScLERP to a new path, or 3. real-time tracking the desired changes whenever the goal is to close (Section IV).

advancement to the robotics community.

II. Problem Definition

This work addresses real-time motion planning problems that enable a manipulator to complete a geometrically constrained task, or a sequence of subtasks, under dynamic and changing task conditions. We define a task, $\mathcal{T} = (\mathcal{T}_i)_{i=1}^{N_t}$, as a sequence of subtasks $\mathcal{T}_i$, $i{=}\{1,2,\ldots,N_t\}$, following geometric conditions acquired from single or multiple successful task demonstrations, as seen in Fig.1. The dynamic and fast-changing environment includes the influence of multiple dynamic obstacles over the whole body of the manipulator, as well, as dynamically changing goals/tasks during different stages of the task execution, as shown in ④ in Fig. 1.

A. Preliminaries

First, we introduce core geometric concepts and definitions, highlighting the role of dual quaternion algebra (DQ) and its application towards screw-theory-based interpolation and path blending. Central to our approach, these concepts exploit the advantages of unit dual quaternions, such as translation and orientation coupling [16], singularity-free representation, global convergence [17], and wrench, twists and geometric primitive embeddings [18]. Notably, DQ also provides a more computationally efficient and simply-connected topology compared to $SE(3)$, [19]. Building on these mathematical foundations, we then define the problem statement for this paper.

In our framework, a task is composed of a sequence of subtasks consisting of a sequence of rigid-body transformations embedding the object-centric geometric constraints defining the (sub-)task. The rigid body transformation is described using unit dual quaternions, $\underline{\boldsymbol{x}} \in \mathrm{Spin}(3) \ltimes \mathbb{R}^3$,

$$\underline{\boldsymbol{x}} = \boldsymbol{r} + \tfrac{1}{2}\varepsilon \boldsymbol{p}\boldsymbol{r}, \tag{1}$$

where $\boldsymbol{p}$ is a pure quaternion that represents the translation, $\boldsymbol{r}{=}\cos(\phi/2){+}\sin(\phi/2)\boldsymbol{n} \in \mathrm{Spin}(3)$ represents a unit-quaternion rotation with angle ϕ around the axis $\boldsymbol{n}$, and ε is such that $\varepsilon{\neq}0$ but $\varepsilon^2{=}0$, [20]. The $\mathrm{Spin}(3) \ltimes \mathbb{R}^3$ is a Lie group with inverse $\underline{\boldsymbol{x}}^*{=}\boldsymbol{r}^*{+}\frac{1}{2}\varepsilon\boldsymbol{r}^*\boldsymbol{p}^*$, and identity $\underline{\mathbf{1}}$.

Now, given a sequence $\underline{\boldsymbol{x}}_i{\in}\mathrm{Spin}(3) \ltimes \mathbb{R}^3$ of poses describing a path, we seek to generate a smooth curve in $SE(3)$. To seamlessly connect and blend the spatial transformations $\underline{\boldsymbol{x}}_i$, we first explore the geodesic line generated by the bi-invariant metric in SE(3) [21]. The advantages of the bi-invariance for trajectory generation are highlighted in the works [22]. This geodesic leads to the well-known screw motions [23], and by connecting two points, e.g., $\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2 \in \mathrm{Spin}(3) \ltimes \mathbb{R}^3$, by means of a constant velocity within the geodesic line, we have a **screw linear interpolation**. Our goal therefore is to obtain equally spaced waypoints in the path $\underline{\boldsymbol{x}}(\tau) : [0,1] \rightarrow \mathrm{Spin}(3) \ltimes \mathbb{R}^3$ with $\underline{\boldsymbol{x}}(0) = \underline{\boldsymbol{x}}_1$ and $\underline{\boldsymbol{x}}(1) = \underline{\boldsymbol{x}}_2$. To this aim, we first map $\underline{\boldsymbol{x}}_2$ along the geodesic on $\mathrm{Spin}(3) \ltimes \mathbb{R}^3$ through $\underline{\boldsymbol{x}}_1$ onto the tangent space at $\underline{\boldsymbol{x}}_1$. This corresponds to a vector mapping in the geodesic direction of $\underline{\boldsymbol{x}}_2$ with respect to $\underline{\boldsymbol{x}}_1$ within $\mathcal{T}_{\underline{\boldsymbol{x}}_1}\mathrm{Spin}(3) \ltimes \mathbb{R}^3$. Hence,

$$\log_{\underline{\boldsymbol{x}}_1}(\underline{\boldsymbol{x}}_2) = \underline{\boldsymbol{x}}_1 \log(\underline{\boldsymbol{x}}_1^* \underline{\boldsymbol{x}}_2), \tag{2}$$

where $\log_{\underline{\boldsymbol{x}}_1}$ is computed using parallel transport using the exp and log maps from the tangent space, at the identity, i.e., $\mathcal{T}_{\underline{\mathbf{1}}}\mathrm{Spin}(3) \ltimes \mathbb{R}^3$ which are given by the dual vector representing the axis of screw motion and the dual angle containing both the translation length and the angle of rotation, see further details in [24]. Thus, we can linearly interpolate points in (2) and compute any point along the geodesic screw direction starting from $\log_{\underline{\boldsymbol{x}}_1}(\underline{\boldsymbol{x}}_1)$ towards $\log_{\underline{\boldsymbol{x}}_1}(\underline{\boldsymbol{x}}_2)$, i.e., $(\log_{\underline{\boldsymbol{x}}_1}(\underline{\boldsymbol{x}}_2) - \log_{\underline{\boldsymbol{x}}_1}(\underline{\boldsymbol{x}}_1))\tau + \log_{\underline{\boldsymbol{x}}_1}(\underline{\boldsymbol{x}}_1)$, with $\log_{\underline{\boldsymbol{x}}_1}(\underline{\boldsymbol{x}}_1) = 0$. Hence, using parallel transport to map the vector in $\mathcal{T}_{\underline{\boldsymbol{x}}_1}\mathrm{Spin}(3) \ltimes \mathbb{R}^3$ back to the unit DQ manifold following the geodesics through $\underline{\boldsymbol{x}}_1$ yields

$$\underline{\boldsymbol{x}}(\tau) = \exp_{\underline{\boldsymbol{x}}_1}(\underline{\boldsymbol{x}}_1 \log(\underline{\boldsymbol{x}}_1^* \underline{\boldsymbol{x}}_2)\tau) = \underline{\boldsymbol{x}}_1 \exp(\log(\underline{\boldsymbol{x}}_1^* \underline{\boldsymbol{x}}_2)\tau),$$

which is often represented by

$$ScLERP(\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2, \tau) = \underline{\boldsymbol{x}}_1 (\underline{\boldsymbol{x}}_1^* \underline{\boldsymbol{x}}_2)^\tau, \tag{3}$$

returning any rigid pose transformation along the geodesic direction from $\underline{\boldsymbol{x}}_1$ to $\underline{\boldsymbol{x}}_2$ scaled linearly along $\tau \in [0,1]$. Now, it is worth mentioning that, in contrast to our previous results [11], in this work, we extend the proposed interpolation algorithm to a C^1 interpolation [25], ensuring smoother transitions, which is critical for the dynamic scenarios envisioned. This comes with a caveat: we would require more control over the curve than just the start and end poses. Formally, one local segment in this curve can be uniquely defined by 4 points: start pose, end pose, and two control poses [26]. Intuitively, control points determine the shape of the curve. The more control points we add, the greater our control over the final shape of the curve. Extending Shoemake's idea of drawing great arcs and proportions of arcs on a sphere to the SE(3) space, it is possible to formalize a de Casteljau-type scheme for C^1-ScLERP [27]. This rigid body motion curve parameterized with dual quaternions results in a series of in-between poses that are identical and coordinate invariant. Like Bezier curves, we can guarantee first-order continuity

by ensuring that as one moves along this curve through its endpoints, it is tangent to its end segments [28].

Lemma 1. *For two rigid body poses (start and end of a local segment) $\underline{\boldsymbol{x}}_1$ and $\underline{\boldsymbol{x}}_2$ and the corresponding control poses $\underline{\boldsymbol{x}}_{a,1}$ and $\underline{\boldsymbol{x}}_{b,2}$, the C^1-ScLERP function is defined as*

$$\begin{aligned} C^1\text{-}ScLERP =& ScLERP(ScLERP(ScLERP(\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_{a,1}, \tau), \\ & ScLERP(\underline{\boldsymbol{x}}_{a,1}, \underline{\boldsymbol{x}}_{b,2}, \tau), \tau), ScLERP(ScLERP \\ & (\underline{\boldsymbol{x}}_{a,1}, \underline{\boldsymbol{x}}_{b,2}, \tau), ScLERP(\underline{\boldsymbol{x}}_{b,2}, \underline{\boldsymbol{x}}_2, \tau), \tau), \tau). \end{aligned} \tag{4}$$

For additional formulas about control points and more detailed proofs, we refer readers to [25].

B. Overview of Approach

In this study, we focus on tasks defined by implicit, often object-centric constraints, exemplified by the tasks of pouring from a glass into a bowl and subsequently placing it on a table. We offer a geometrically consistent way to embed such tasks through rigid body transformations obtained from demonstrations – for instance, when pouring, the axis and angle of rotation are implicitly constrained w.r.t. the bowl. This path can be later smoothly blended through C^1-ScLERP from Lemma 1.

Our framework allows for multiple demonstrations that embed geometric constraint flexibilities – e.g., multiple paths when moving the glass – which can be iteratively updated without additional storage costs as only the average path and the obtained bounds prescribing the task relaxation are required. Those can be defined by a tuple sequence $\mathcal{T}^D = (\underline{\boldsymbol{x}}_i, \epsilon_i)_{i=1}^{\eta}$, where $\underline{\boldsymbol{x}}_i$ and ϵ_i depicts the average and the bounds of the $i-$th element of the sequence. Finally, after a single or multiple successful task demonstrations, our planner is able to generate a new path from a new initial pose $\underline{\boldsymbol{x}}_0$ to a new final goal $\underline{\boldsymbol{x}}_f$, satisfying the implicit task constraints and flexibilities observed in $\mathcal{T}^D$. A full demonstration may involve a sequence of subtasks, that is, $\mathcal{T}^D{=}(\mathcal{T}_i^D)_{i=1}^{N_t}$ which are sequentially connected, as illustrated in step ①-② (numbers inside circles) in Fig. 1. Further details are given in Section III.

While the resulting task-representation satisfies the implicit constraints from (potentially) multiple demonstrations, real-world deployment still calls for real-time reactions and adaption to unforeseen scenarios. Particularly critical are collision avoidance – considering self-collision, end-effector and whole-body avoidance against single or multiple obstacles – and adaptation to task changes, e.g., dynamically changing goals often addressed in visual servoing controllers.

In our approach, the reactive real-time response to self-collision and dynamic obstacles builds on our previous work in vector-field-based reactive planners [29], [30]. The resulting repulsive force required for collision avoidance is mapped to the average trajectory at the end-effector and/or nullspace levels. A HQP-based controller, as seen in ③ and in Section V, is then defined to track the modified trajectory, while optimizing the manipulability and joint-configuration in the task-nullspace within the prescribed task-bounds ϵ_i.

This approach considerably extends existing user-guided methods such as [1], [11], [31], as well as most LfD deployments [3], [9]. However, they still require an offline definition of goals, not being suitable for integration with visual servoing methods [14], [15], [32]. In contrast, visual servoing methods while efficient in addressing task modifications often don't satisfy task constraints. In this work, we propose a novel adaptation method that considers different task conditions, as shown in ④ in Fig. 1, and detailed in Section IV. More specifically, we offer a geometric and task-constraint consistent method to (Case 3) replan subtasks, whenever possible – that is, when they haven't started – (Case 2) blending task modifications in a reactive/online visual-servoing planning approach which satisfies the demonstrated constraints; and (Case 1) to react and track the demonstrated trajectory in a traditional visual servoing tracking control approach – yet based on the geodesic of the screw motion which couples translation and rotation constraints.

III. Sequential User-Guided Path Planning

This section outlines our path-planning strategy aimed at finding a sequence of rigid body transformations that takes the end-effector from an initial configuration $\underline{\boldsymbol{x}}_0$ to a final goal $\underline{\boldsymbol{x}}_f$, going through multiple ℓ-subtasks, herein defined by key-intermediated-poses $\underline{\boldsymbol{x}}^{\ell}$, while satisfying the constraints observed in previous demonstrations.

A. Embedding Task Demonstrations

The algorithm takes, at least, a single successful task demonstration—defined by a sequence of poses

$$\mathcal{T}^{\mathcal{D}}{=}\{\underline{\boldsymbol{x}}_1^{\mathcal{D}}, \underline{\boldsymbol{x}}_2^{\mathcal{D}} \ldots, \underline{\boldsymbol{x}}_{\eta}^{\mathcal{D}}\}, \quad \underline{\boldsymbol{x}}_i^{\mathcal{D}} \in \mathrm{Spin}(3) \ltimes \mathbb{R}^3. \tag{5}$$

Since the task demonstration is a successful one, it satisfies the task constraints, and this knowledge is implicitly embedded in the demonstration. A key point to highlight is that no other information, or knowledge, about the task or its constraints is required or needs to be provided.

When a demonstration path (5) includes multiple complex subtasks, especially when these are constrained by the pose of external or intermediate objects, merely exploring the entire $\mathcal{T}^{\mathcal{D}}$ using traditional methods [1], [11] is insufficient to fully describe the task – particularly, if the positions of these external objects change. Therefore, we segment the $\mathcal{T}^{\mathcal{D}}$ into a series of N_t sequential object-centric subtasks, $\mathcal{T}^{\mathcal{D}} = \left(\mathcal{T}_i^{\mathcal{D}}\right)_{i=1}^{N_t}$. For each subtask, we have designed a geometric-constrained window that encompasses the trajectory and implicit geometric constraints the manipulator needs to fulfill. The trajectory within the subtask ℓ is denoted as $\mathcal{T}_{\ell}^{\mathcal{D}} = \underline{\boldsymbol{x}}_{\ell,1:\eta_\ell}^{\mathcal{D}}$ [1], where $\underline{\boldsymbol{x}}_{\ell,o}^{\mathcal{D}} \triangleq \underline{\boldsymbol{x}}_{\ell,1}^{\mathcal{D}}$ and $\underline{\boldsymbol{x}}_{\ell,f}^{\mathcal{D}} \triangleq \underline{\boldsymbol{x}}_{\ell,\eta_\ell}^{\mathcal{D}}$ are the starting and goal poses of the subtask. Notice that $\mathcal{T}_{\ell}^{\mathcal{D}} \in \mathcal{T}^{\mathcal{D}}$, $\forall \ell$. These poses share geometric and sequential constraints with task-related objects and are subject to change depending on the environment. For instance, take the task in Fig. 1 where $\underline{\boldsymbol{x}}_{1,\eta_1}^{\mathcal{D}}$ defines the grasping pose, from subtask $\mathcal{T}_1^{\mathcal{D}}$.

Notice that, in many tasks, a task is also defined by a pre-goal configuration, for instance, a pre-grasping pose. Therefore, in addition to key poses $\underline{\boldsymbol{x}}_{\ell,o}^{\mathcal{D}}, \underline{\boldsymbol{x}}_{\ell,f}^{\mathcal{D}}$, each subtask also contains a key adaptive point, $\underline{\boldsymbol{x}}_{\ell,\alpha}^{\mathcal{D}}$ from where the task must be fully imitated, e.g., a pre-grasping pose in $\mathcal{T}_1^{\mathcal{D}}$. The adaptive poses are critical for the task adaption in Section IV, where they are further detailed.

Assumption: In this work, we consider subtasks defined by the affordances between the end-effector and pair of objects. More specifically, given a threshold around the proximity between objects during a demonstration, e.g., the cup and the coaster, we define key intermediate poses (or subtask goals, $\underline{\boldsymbol{x}}_{\ell,\eta_\ell}^{\mathcal{D}}$) which lead to the subtask segmentation.

[1] In all our task variables definition, we employ the following notation: $\underline{\boldsymbol{x}}_{\ell,i}^{Mk}$, where ℓ denotes the subtask index; i refers to the keypoint within the subtask trajectory; M represents the type of task instance description, with $M \in \{D, I, N\}$ indicating Demonstration, Imitation, or New task instance, respectively; and k specifies the demonstration number for D, e.g., Dk refers to the k-th demonstration.

Notwithstanding, the same segmentation can be achieved, for instance, by means of the stochastic variance from multiple demonstrations or a pre-defined task-representation.

B. Sequential & Relaxed Task-Space Imitation Alg. (srTSIA)

During real-world deployment, we will have both novel initial and final goals, for instance, the placement of cups, bowls and coasters will differ from every task instance. To map the task constraints from (5) and, particularly, its subtasks, we start computing the object-goal-centric transformations, between the last pose on the demonstrated subtask, $\underline{\boldsymbol{x}}^{\mathcal{D}}_{\ell,\eta_\ell} \in \mathcal{T}^{\mathcal{D}}_{\ell}$, and every other pose within $\mathcal{T}^{\mathcal{D}}_{\ell}$, that is,

$$\underline{\boldsymbol{\delta}}^{\mathcal{D}}_{\ell,i} = (\underline{\boldsymbol{x}}^{\mathcal{D}}_{\ell,i-1})^{*}(\underline{\boldsymbol{x}}^{\mathcal{D}}_{\ell,\eta_\ell}),\ i = 2,\dots,\eta_\ell. \tag{6}$$

Then, given a new sub-task instance goal pose $\underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,f}$ the resulting imitated path leading to the new object-centric goal can be described by the following transformations

$$\underline{\boldsymbol{x}}^{\mathcal{I}}_{\ell,i} = \underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,f}(\underline{\boldsymbol{\delta}}^{\mathcal{D}}_{\ell,i})^{*},\ i = 2,\dots,n. \tag{7}$$

The **Subtask Final Path** then needs to smoothly connect the new initial pose $\underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,0}$ to the new goal $\underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,f}$ following the prescribed constraints from the demonstration. Similarly to [1], this is achieved by blending $\underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,0}$ within the resulting imitated trajectory $\underline{\boldsymbol{x}}^{\mathcal{I}}_{\ell,1:\eta_\ell}$ which embeds the geometric constraints from $\mathcal{T}^{\mathcal{D}}_{\ell}$. In contrast to our previous results, this blending follows the C^1-ScLERP from Lemma 1, which is critical for the adaptive step in Section IV. The resulting path is also defined by the blending guiding pose $\underline{\boldsymbol{x}}^{\mathcal{I}}_{\ell,g}$, with $0<g\leq\eta_\ell$. The resulting target pose is then given by

$$\underline{\boldsymbol{x}}_{\ell,t}(\tau) = C^1\text{-ScLERP}\left(\underline{\boldsymbol{x}}_{\ell,c}, \underline{\boldsymbol{x}}^{\mathcal{I}}_{\ell,g}, \tau\right), \tag{8}$$

where C^1-ScLERP is defined in Lemma 1, and $\underline{\boldsymbol{x}}_{\ell,c}$ is the current pose, starting from $\underline{\boldsymbol{x}}_{\ell,o}$, within the new instance of the subtask ℓ, i.e., $\mathcal{T}^{\mathcal{N}}_{\ell} = \underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,1:\eta_\ell}$, and $\tau \in [0,1]$ is the time primitive. Different τ gives different target poses, i.e., $\tau = 0$ leads to $\underline{\boldsymbol{x}}_{\ell,c}$ and $\tau = 1$ corresponds to $\underline{\boldsymbol{x}}^{\mathcal{I}}_{\ell,g}$. The guiding pose defines where to blend within the imitated path, i.e., $0<g\leq\eta_\ell$. Iteratively, the blending evolves with $g \to \eta_\ell$. This process yields the final sub-task trajectory.

Notice that since our blending is based on the geodesic screw line between key points, the translation and rotation information from the screw is preserved, and kept between the boundaries defined by both points. In other words, if both $\underline{\boldsymbol{x}}_i$ and $\underline{\boldsymbol{x}}_{i+1}$ satisfy the demonstrated task constraints, then so will the resulting path. This property is intrinsic to our planning strategy satisfying demonstrated geometric constraints without explicit definition, which in turn would require an expert task design. As previously highlighted in [1], [25], this property is not guaranteed in decouple interpolation methods usually found in the literature.

The **Final Path** is designed by the sequential subtasks, i.e.,

$$\mathcal{T}^{\mathcal{N}} = (\mathcal{T}^{\mathcal{N}}_{i})^{N_t}_{i=1}. \tag{9}$$

To ensure consistency, the subtask reconstruction starts from $N_t \to 1$ connecting new goal poses to new starting ones.

C. Incorporating Flexibility: srTSIA through Multiple Demonstrations

Integrating flexibility during motion execution is crucial for task success in constrained spaces. To this end, we take advantage of capturing the inherent variability present in multiple human demonstrations – whenever available. This variability reveals the task's internal range and underscores the potential for adaptability.

Therefore, to analyze multiple trajectories and infuse flexibility, we apply stochastic data processing into the end-effector workspace. In the workspace, the poses in each subtask $\underline{\boldsymbol{x}}^{\mathcal{D}}_{\ell}$, should be mapped from DQ into Riemannian vector space through function $\boldsymbol{x^p} = \gamma(\underline{\boldsymbol{x}}^{D}_{\ell})$, where γ represents the mapping from the $SE(3)$ to a given geodesic line depicting a geometric primitive, such as a Cartesian vector space, line-to-line distance angles, orientation geodesics or even the screw geodesic in $\mathbb{R}^6$. Thus, $\boldsymbol{x^p}$ is the geometric primitive mapped on the tangent space within $\mathbb{R}^N$. $\boldsymbol{x^p}$ is approximated by a linear model based on a set of Gaussian distributions.

$$\boldsymbol{x^p} = \boldsymbol{\omega}\Psi + \boldsymbol{\kappa}, \boldsymbol{\kappa} \sim \mathcal{N}(0, \boldsymbol{\Sigma_x}), \tag{10}$$

where ω is the weights of the model and Ψ represents the set of Gaussian basis functions. κ brings a zero-mean Gaussian noise with variance $\Sigma_{\boldsymbol{x}}$. By weighing the basis functions, the term $\omega\Psi$ offers the average of the multiple trajectories. The probability of a demonstration (5) is given by accumulating all time steps from $i = 1$ to η.

$$p(\mathcal{T}^{\mathcal{D}}; \boldsymbol{\omega}) = \prod_{i=1}^{\eta} \mathcal{N}(\boldsymbol{x^p}|\boldsymbol{\omega}\Psi, \boldsymbol{\Sigma_x}). \tag{11}$$

The higher the probability $p(\mathcal{T}^{\mathcal{D}}; \omega)$ is, the better the model fits the task demonstration. Besides, another Gaussian distribution (12) over the weight vector $\boldsymbol{\omega}$ captures the variance of multiple trajectories.

$$p(\boldsymbol{\omega}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu_\omega}, \boldsymbol{\Sigma_\omega}) \tag{12}$$

Thus, the probability $p(\mathcal{T}^{\mathcal{D}}; \boldsymbol{\omega})$ can be marginalized by a Hierarchical Bayesian Model (HBM) with

$$p(\mathcal{T}^{\mathcal{D}}; \boldsymbol{\theta}) = \int p(\mathcal{T}^{\mathcal{D}}|\boldsymbol{\omega}) p(\boldsymbol{\omega}|\boldsymbol{\theta}) d\boldsymbol{\omega}. \tag{13}$$

By accumulating two Gaussian models in (10) and (12),

$$p(\mathcal{T}^{\mathcal{D}}; \boldsymbol{\theta}) = \mathcal{N}(\underline{\boldsymbol{x}}_i, \epsilon). \tag{14}$$

Learned from multiple trajectories, the parameter $\boldsymbol{\theta} = \{\boldsymbol{\mu_\omega}, \boldsymbol{\Sigma_\omega}\}$ guarantees tracking mean trajectory $\underline{\boldsymbol{x}}_i = \Psi\boldsymbol{\mu_\omega}$ of multiple demonstrations. And the variance $\epsilon = \Psi^T\boldsymbol{\Sigma_\omega}\Psi + \boldsymbol{\Sigma_x}$ provides us the flexibility we exploit under diverse scenarios. Such stochastic data processing is also used by Probabilistic Movement Primitives [8] in multiple demonstrations analysis.

IV. Dynamic User-Guided Visual Servoing Planning and Control

This section first presents the main algorithm that integrates the srTSIA together with the proposed adaption to moving targets and changing conditions.

The first step is srTSIA, which firstly takes the last subtask and builds a path newly detected final point of each of the previous subtask, blending the trajectory with C^1-ScLERP, as shown in lines 3-7 in Alg.1. This process moves until the first subtask. Following steps from Section III, a new path is generated.

Now, to achieve adaptability throughout a sequence of tasks, the Adaptive Sequential and Relaxed Task Imitation Algorithm (ASR-TSIA) is proposed to dynamically plan real-time trajectories that accommodate changes in task-related objects while ensuring the geometric constraint of the task. In ASR-TSIA, we continuously check the final pose of each subtask ℓ. If it changes from the previous timestep, that is, changing in task-conditions, we compute the spatial transformation from the previous step. We also extract the adaptive point $\underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,a}$ that the robot must be fully imitated – e.g., the sequential transformation between pre-grasp $\underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,a}$ and grasping $\underline{\boldsymbol{x}}^{\mathcal{N}}_{\ell,f}$. We categorize the subtask changing into

Algorithm 1: Adaptive Sequential and Relaxed Task Imitation Algorithm (ASR-TSIA)

1 **Procedure srTSIA**: ($\underline{\boldsymbol{x}}_o^{\mathcal{N}}$, $\underline{\boldsymbol{x}}_{1:\eta_\ell,f}^{\mathcal{N}}$, $\underline{\boldsymbol{x}}_{\ell,1:\eta_\ell}^{D_{1:Nd}}$, $\Sigma_{\ell,1:\eta_\ell}^{D}$);
2 $\ell_0 \leftarrow 1$;
3 **for** $\ell : \ell_0 \rightarrow \eta_\ell$ **do**
4 $\quad \underline{\boldsymbol{x}}_{\ell,0}^{\mathcal{N}} \leftarrow \underline{\boldsymbol{x}}_{\ell-1,f}^{\mathcal{N}}$;
5 $\quad \underline{\boldsymbol{x}}_{\ell,1:\eta_\ell}^{\mathcal{I}} \leftarrow$ Compute imitated path ($\underline{\boldsymbol{x}}_{\ell,f}^{\mathcal{N}}$), (6)-(7);
6 $\quad \underline{\boldsymbol{x}}_{\ell,1:\eta_\ell}^{\mathcal{N}} \leftarrow$ Compute new path ($\underline{\boldsymbol{x}}_{\ell,0}^{\mathcal{N}}, x_{\ell,1:\eta_\ell}^{\mathcal{I}}$), (8);
7 $\quad \mathcal{T}_\ell^{\mathcal{N}} \leftarrow (\underline{\boldsymbol{x}}_{\ell,1:\eta_\ell}^{\mathcal{N}}, \epsilon_{\ell,1:\eta_\ell})$;
8 **Procedure ASR-TSIA**: ($\underline{\boldsymbol{x}}_{\ell,c}$, $\mathcal{T}_\ell^{\mathcal{N}}$, $\underline{\boldsymbol{x}}_{1:\eta_\ell,f}^{\mathcal{N}}$);
9 **for** *All subtasks* ℓ **do**
10 $\quad$ **if** $\underline{\boldsymbol{x}}_{1:\eta_\ell,f}^{\mathcal{N}}[k] \neq \underline{\boldsymbol{x}}_{1:\eta_\ell,f}[k-1] \rightarrow$ *Get* ℓ **then**
11 $\quad\quad \delta_c = \underline{\boldsymbol{x}}_{1:\eta_\ell,f}[k-1]^* \, \underline{\boldsymbol{x}}_{1:\eta_\ell,f}[k]$;
12 $\quad\quad$ Get $\underline{\boldsymbol{x}}_{\ell,a}^{\mathcal{N}}$ from $\underline{\boldsymbol{x}}_{1:\eta_\ell,f}^{\mathcal{N}}[k]$;
13 $\quad\quad$ **if** $\underline{\boldsymbol{x}}_{\ell,c} \in \mathcal{T}_\ell^{\mathcal{N}}$ **then**
14 $\quad\quad\quad$ **if** $\underline{\boldsymbol{x}}_{\ell,c} \in \mathcal{T}_{\ell,a:f}^{\mathcal{N}}$ **then**
15 $\quad\quad\quad\quad \underline{\boldsymbol{x}}_{\ell,a:f}^{\mathcal{N}}[k] \leftarrow$ Compute δ_c spatial displacement and feed-forward-term;
16 $\quad\quad\quad$ **else**
17 $\quad\quad\quad\quad \underline{\boldsymbol{x}}_{\ell,g:f}^{\mathcal{I}}[k] \leftarrow \underline{\boldsymbol{x}}_{\ell,g:f}^{\mathcal{I}}[k-1]^* \, \delta_c$;
18 $\quad\quad\quad\quad \underline{\boldsymbol{x}}_{\ell,1:\eta_\ell}^{\mathcal{N}} \leftarrow$ Compute new path, (8);
19 $\quad\quad$ **else**
20 $\quad\quad\quad$ **for** $\ell : Nt \rightarrow (\ell+1)$ **do**
21 $\quad\quad\quad\quad$ Compute new subtasks (line 4-7)

three cases.

(Case 1 – Real-time Screw Servoing Tracking Control): Line 14, where $\underline{\boldsymbol{x}}_{\ell,c}$ lies inside the region $\mathcal{T}_{\ell,a:f}^{\mathcal{N}}$, that is, between the adaptive keypoint (that defines, for instance, the pre-grasping pose) and the final pose. In this case, full attention is given to redefining the remaining trajectory $\mathcal{T}_{\ell,c:f}^{\mathcal{N}}$ with a spatial displacement given by δ_c as defined in Line 11. Additionally, a feed-forward term is computed based on the screw-geodesic direction from the δ_c from $\underline{\boldsymbol{x}}_{\ell,c}$.

(Case 2 – Real-Time User-Guided Blending): Instead, this case considers when $\underline{\boldsymbol{x}}_{\ell,c}$ lies inside the current subtask execution yet before the adaptive pose $\underline{\boldsymbol{x}}_{\ell,a}^{\mathcal{N}}$. In other words, [$\underline{\boldsymbol{x}}_{\ell,o}^{\mathcal{N}}$, $\underline{\boldsymbol{x}}_{\ell,\alpha}^{\mathcal{N}}$] within the geometric-constrained window of subtask ℓ. This situation is given in Lines 16. In this condition, the final subtask goal (Line 17) is updated, and a novel trajectory is blended with (8) from the current pose (Line 18).

Special Case: Dynamic Recovering: Similar to Case 1, but with the subtask condition changing significantly such that $\underline{\boldsymbol{x}}_{\ell,c}$ falls outside the modified geometrically constrained window – moving to Case 2. Continuing the task as in Case 1 could lead to failure. In this case, the robot stops the tracking and following Lines 16-28 designs a new path ensuring demonstrated constraints are satisfied.

(Case 3 - Online Replanning for Long-Target Horizon): If the changed subtask has not been executed, that is the change of the subtask will not affect the current movement of the robot, we only need to replan future subtask executions using (Lines 19-21).

V. Constrained Reactive HQP-Based Controller

To cope with dynamic obstacles, joint limits, and singularities, all of which can cause tasks to fail during real-time motion generation, we introduce a constrained reactive HQP.

In contrast to our previous work [1], [11], our proposed controller makes direct use of the probabilistic information embedded into multiple task demonstrations, see Subsection III-A. The stochasticity of the demonstrations, that is, its confidence interval, informs our framework on the geometric flexibilities and user preferences – which are herein exploited as set-based constraints. This enlarges the robot's search space, which enables them to adjust, for instance, for better manipulability and joint-limit avoidance.

To account for the task flexibilities, herein defined either by Cartesian coordinates (3 DoFs), screw geodesic lines (6 DoFs), orientation bounds (3 DoFs), or line-to-line distances (1 DoFs), we augment the dimension of the input variable by n_s, according to the dimensionality of the task flexibility – defined by geometric primitives. See [33], [34] for further primitives. Therefore, we augment the kinematic control variable as follows

$$\boldsymbol{u} = \begin{bmatrix} \dot{\boldsymbol{q}}^T & \boldsymbol{s}^T \end{bmatrix}^T, \quad \boldsymbol{u} \in \mathbb{R}^{(n_r+n_s)\times 1}, \tag{15}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^{n_r}$ is the joint velocity, and $\boldsymbol{s} \in \mathbb{R}^{n_s}$ depicts the corresponding geometric primitive flexibility mapped to vector space – herein addressed as the optimization slack variable. The primary task problem is thus formulated as

$$\begin{aligned} \min_{\boldsymbol{u}} \; & \left\{ \kappa_1 \|\mathcal{J}_{\mathcal{T}}\|^2 + \kappa_2 \|\mathcal{J}_\sigma\| + \kappa_3 \|\mathcal{J}_q\|^2 + \kappa_4 \|\mathcal{J}_s\|^2 \right\} \\ s.t. \quad & \mathcal{A}_{\mathbb{S}} \cdot \Delta t \cdot \boldsymbol{u} \leq \boldsymbol{b}_{\mathbb{S}} \qquad \mathcal{LB}_p \leq \boldsymbol{u} \leq \mathcal{UB}_p \\ & \mathcal{LB}_v \leq \boldsymbol{u} \leq \mathcal{UB}_v \quad \mathcal{LB}_a \leq \boldsymbol{u} \leq \mathcal{UB}_a, \end{aligned} \tag{16}$$

where $\kappa 1$, $\kappa 2$, $\kappa 3$, $\kappa 4$, are positive control gains , $\mathcal{J}_{\mathcal{T}}$ addresses the collision avoidance pipeline, and $\mathcal{J}_\sigma$, $\mathcal{J}_q$, and $\mathcal{J}_s$ refer to the manipulability, joint-limit and slack variable optimization, respectively. The constraints are designed to maintain the robot's operation within limited bounds. While the $\mathcal{LB}_p, \mathcal{LB}_v, \mathcal{LB}_a$ and $\mathcal{UB}_p, \mathcal{UB}_v, \mathcal{UB}_a$ respectively defines the lower and upper bounds for the joint vector position, velocities and accelerations, computed from first-order differentiation. Notice that these are block vectors with n_s-zero elements in the lower rows.

The matrices $\mathcal{A}_{\mathbb{S}}$ and $b_{\mathbb{S}}$ are defined as

$$\mathcal{A}_{\mathbb{S}} = \begin{bmatrix} \boldsymbol{J}_{n_s\times n_r} & \boldsymbol{0}_{n_s\times n_s} \\ -\boldsymbol{J}_{n_s\times n_r} & \boldsymbol{0}_{n_s\times n_s} \\ \boldsymbol{0}_{n_s\times n_r} & \boldsymbol{I}_{n_s\times n_s} \end{bmatrix}, \boldsymbol{b}_{\mathbb{S}} = \begin{bmatrix} \gamma(\underline{\boldsymbol{x}}_{\ell,i}) - \epsilon_{\ell,i} \\ -\gamma(\underline{\boldsymbol{x}}_{\ell,i}) + \epsilon_{\ell,i} \\ \epsilon_{\ell,i} \end{bmatrix},$$

where $\boldsymbol{J}_{n_s\times n_r}$ is the geometric primitive Jacobian, see [33], [34], and $\gamma(\underline{\boldsymbol{x}}_{\ell,i})$ is the geometric primitive, e.g., Cartesian position, screw line, angle, among others, corresponding to the pose $\underline{\boldsymbol{x}}_i$ at step i of the subtask ℓ. Similarly, $\epsilon_{\ell,i}$ is the corresponding slack variable, that is, the confidence interval acquired from the variance in Subsection III-A.

In addition to the task-flexibility bounds, our HPQ also tries to minimize the slack variables to avoid pushing the solution always to the boundaries, that is, striving to keep it as close as possible to the average – without the same priority as other costs. This is achieved through

$$\mathcal{J}_s = \operatorname{diag}(\boldsymbol{0}_{n_r\times n_r}, \quad \boldsymbol{I}_{n_s\times n_s}) \cdot \boldsymbol{u}, \tag{17}$$

where $\operatorname{diag}(\cdot)$ is a block diagonal matrix.

The minimum singular value optimization is given by $\max \dot{\sigma}_{\min}$ where $\sigma_{\min}$ is the minimum singular value from the Geometric Jacobian. This desired behaviour can be obtained by minimizing

$$\mathcal{J}_\sigma = -\lambda \bar{J}_\sigma \boldsymbol{u}, \qquad \bar{J}_\sigma = [\boldsymbol{J}_\sigma, \boldsymbol{0}_{1\times n_s}], \tag{18}$$

where $\boldsymbol{J}_\sigma$ is the minimum singular value Jacobian with respect to robot joint velocities.

To improve the robot posture, and avoid joint-limits, we additionally include a joint-space goal towards the midpoint of the joint limits $\boldsymbol{q}_{mid} = (\boldsymbol{q}_{max} + \boldsymbol{q}_{min})/2$. The following cost actively strives to move the current joint position towards $\boldsymbol{q}_{mid}$.

$\mathcal{J}_q = \operatorname{diag}(\boldsymbol{I}_{n_r\times n_r}, \boldsymbol{0}_{n_s\times n_s}) \cdot \Delta t \boldsymbol{u} + [\,(\boldsymbol{q}_c{}^T - \boldsymbol{q}_{mid}^T) \; \boldsymbol{0}_{1\times n_s}\,]^T$.

Finally, to navigate around dynamic obstacles, we utilize an Artificial Potential Field (APF) method that generates a repulsive force from obstacles on the end effector, updating the guiding points $\underline{\boldsymbol{x}}_{gp}$ with this force to equip them with obstacle-avoidance capabilities. The updated $\underline{\boldsymbol{x}}_{gp}$ is

$$\underline{\boldsymbol{x}}'_{gp} = r(\underline{\boldsymbol{x}}_{gp}) + \frac{1}{2}\varepsilon(p(\underline{\boldsymbol{x}}_{gp}) + \text{Tran}(\mathcal{F}_r^{EE}))r(\underline{\boldsymbol{x}}_{gp}), \quad (19)$$

$$\mathcal{F}_r^{EE} = \mathcal{F}_{att}^{EE} + \mathcal{F}_{rep}^{EE}, \quad (20)$$

$$\mathcal{F}_{att}^{EE} = k_a \overrightarrow{x_c x_{gp}} \quad (21)$$

$$\mathcal{F}_{rep}^{EE} = k_r\,(1/Dis(x_c, x_o) - 1/\varpi)\,(1/Dis(x_c, x_o))^2 \overrightarrow{x_o x_c} \quad (22)$$

where $r(\cdot)$ extracts the rotation component of a dual quaternion, $p(\cdot)$ isolates the translation part, and Tran$(\cdot)$ converts a three-dimensional vector into a quaternion. $\overrightarrow{x_c x_{gp}}$ is the vector between guiding point position $x_d \in \mathbb{R}^3$ to robot current position $x_c \in \mathbb{R}^3$. $\overrightarrow{x_o x_c}$ is the vector from robot current position x_r to obstacle position x_o. ϖ is the threshold, and if $Dis(x_c, x_o) > \varpi$, $\mathcal{F}_{rep}^{EE} = 0$. k_a and k_r are the gain. After that the cost function $\mathcal{J}_\mathcal{T}$ is defined as

$$\mathcal{J}_\mathcal{T} = \bar{J}\boldsymbol{u} - \text{vec}(\underline{\boldsymbol{x}}'_{gp}), \quad \bar{J} = [\boldsymbol{J}, \boldsymbol{0}_{6\times n_s}], \quad (23)$$

where $\boldsymbol{J}$ is the Jacobian matrix and vec$(\cdot)$ maps the dual quaternion input to a vector space. It is also worth highlighting that in situations where the desired displacement for collision avoidance is larger than the $\epsilon_{\ell,i}$, this confidence interval will be increased until the obstacle has been removed. This is simply to ensure consistency within the control strategy while addressing dynamic obstacles.

1) Secondary task of HQP: The design of the secondary task leverages the robotic arm's redundant degrees of freedom, enabling obstacle avoidance for other joints and preventing self-collision, all without compromising the completion of the primary task. In our framework, the secondary task aims to prevent self-collision and whole-body avoidance, while holding the original objectives.

$$\begin{aligned} \min_{\dot{\boldsymbol{q}}} & \sum_{i=1}^{N_F} \left\| \boldsymbol{J}_i^f \dot{\boldsymbol{q}} - \mathcal{F}_i \right\|^2 \\ s.t. \quad & \boldsymbol{J}_a \dot{\boldsymbol{q}} = \boldsymbol{J}\boldsymbol{u}_1, \end{aligned} \quad (24)$$

where $\boldsymbol{u}_1$ is the control signal computed in (16), and N_F is the number of force vectors acting within the whole body of the manipulator. The force field $\mathcal{F}_i$ includes the repulsion of the whole body provided by obstacles $\mathcal{F}_i^{WB}$ and a self-collision repulsive force $\mathcal{F}_i^{SC}$. When the distance between two links (obstacle and link) is less than a given threshold ϖ_{self} (ϖ), $\mathcal{F}_i^{SC}$ $(\mathcal{F}_i^{WB})$ will act, which is considered to prevent self-collision. A link in a robotic arm may be subjected to multiple forces, herein we consider only the largest one. It's worth noting that due to the restrictions imposed by joint limits, adjacent links of a robotic arm will not experience self-collision. Therefore, when performing calculations, it is necessary to consider every other link. $\boldsymbol{J}_i^f$ represents the Jacobian point-contact force Jacobian (transpose of the geometric Jacobian at the contact-point).

VI. Experimental Results

This section explores quantitative and experimental aspects concerning the performance and evaluation of our proposed framework. The planning algorithms were implemented using DQ_Robotics [35] library and deployed in a Franka Emika Research 2 manipulator controlled in hardware-in-the-loop through a CoppeliaSim simulator[2] with MuJoCo[3] physical engine. The use of a simulator facilitated the online interaction with multiple dynamic obstacles affecting the whole body of the manipulator and, most importantly, the interface towards dynamic shifts in task conditions via moving targets. The primary objective was to simplify the evaluation process, and mainly, enhance the repeatability and precision of the proposed planning scenarios, alongside allowing fair comparison with other methods.

To this aim, we devised two different sequential tasks:

- A simple picking ($\mathcal{T}_1^\mathcal{D}$) and placing ($\mathcal{T}_2^\mathcal{D}$) of a block, following vertical guidance that allows for block stacking without further demonstrations.
- Grasping a cup with a user-preferred orientation ($\mathcal{T}_1^\mathcal{D}$), pouring water in the bowl, with a prescribed angle adjustment along a designed axis of rotation ($\mathcal{T}_2^\mathcal{D}$), and placing on a cup coaster ($\mathcal{T}_3^\mathcal{D}$).

Both tasks were demonstrated $10\times$ on the real robot running a modified impedance controller. All demonstrations were deployed under different initial robot configurations as well as random object placement. From both demonstrated tasks, we devise multiple scenarios including the pouring task with different object placements (Fig. 2(a)), the same scene with dynamic obstacles, Fig. 2(b), pick-and-place of blocks under different object placements, Fig. 2(c), as well as blocking stacking with dynamic obstacles, Fig. 2(d).

A. Efficiency of the HQP-Based Controller and C^1-ScLERP

First, we compare the control performance of our proposed framework against our previous solution [1], [11], where the user-guided approach was deployed via a standard kinematic controller with trajectory tracking following a C^0-ScLERP trajectory. To allow equitable comparison, we explored a simple scenario as in Fig. 2(c) without obstacles, and with stationary targets, thereby omitting the adaptability aspect detailed in Section IV. We also explore a single trajectory tracking, instead of exploring the confidence interval with the set-based control strategy outlined in Section V. While both strategies adhere to the defined task constraints, as expected, the results depicted in Fig. 3 clearly highlight the advantages of our proposed framework which consistently achieves higher manipulability, measured by the minimum singular value. The influence of the minimum singular value is clearly depicted in the norm of the joint velocities, particularly in the red-shaded areas.

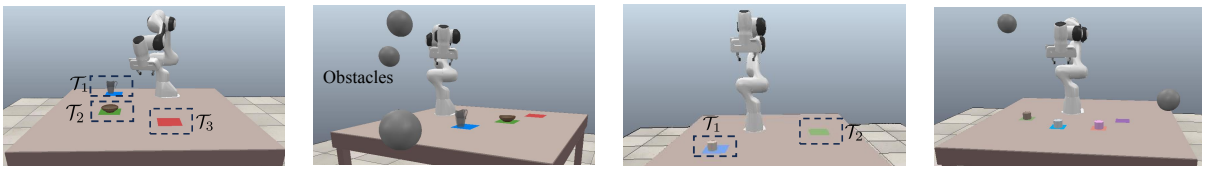
B. Exploring Multiple Demonstrations: Task Flexibilities

Secondly, we tested the capabilities of the proposed HQP framework to explore the task flexibilities computed from the multiple demonstrations, as detailed in Subsection III-A and deployed with the HQP in Section V. To this aim, we devise a similar task, as previously described, yet we also evaluate its adaptability to moving targets. Fig. 4 illustrates both cases with the 2σ range for when the multiple demonstration information is used. The red-shaded areas illustrate the original task (without target modification), whereas the blue-shaded areas depict the modified tasks. The green dotted line within the figure marks the instant at which the alteration of the goal position commences. For both cases, the end-effector Cartesian path is shown in Fig. 4 comparing the vanilla case (blue curves)[4], i.e., following exactly the average trajectory, compared to our proposed task relaxation method (red curves). It is easy to see that integrating task flexibilities leads to a different path (comparing red and blue curves) both

[2] https://coppeliarobotics.com/

[3] https://mujoco.org/

[4] Without target modification, we have the same result as a vanilla ProMP.



(a) (b) (c) (d)

Fig. 2. Task execution scenarios from both pouring and pick-and-place demonstrations. (a) Pouring with no obstacles and a dynamic scene (b); picking-and-place (c) with block-stacking and dynamic obstacles (d).

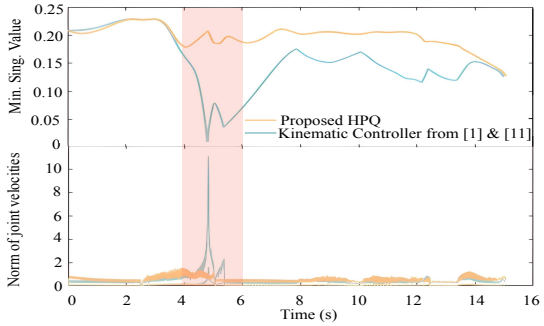


Fig. 3. The comparison between our planner and planner [1], [11].

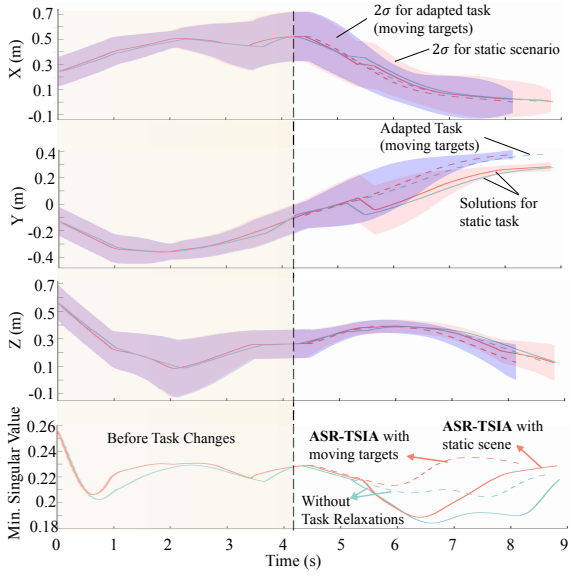


Fig. 4. Performance of the ASR-TSIA planner in both static and moving target scenarios compared to the vanilla scenario – without task flexibilities/relaxations from Subsection III-A.

for the original task (dashed lines) as well as for modified task goals (solid-lines). By exploring the task flexibilities, our framework also allows for improved manipulability, as shown in the bottom graph.

C. Real-time Dynamic Visual Servoing Planning and Control

The capability of our planning framework to adapt to changing sub-goals is illustrated in this use case. We choose a long horizon task for this example and divide it into various episodes. As shown in Fig. 5, the robot starts at an initial pose (different from the demonstration). It blends into the imitated path, which can also be regulated depending on the task. We now highlight 3 crucial events occurring sequentially: (i) Dynamic grasping (light blue frames), (ii) Dynamic recovery after a pouring task (dark blue frames), and (iii) Dynamic pouring (blue frames). It is important to note that in all of these events, employing an approach without ASR-TSIA results in a task failure. Thanks to our geometric-constrained window, the system can execute dynamic grasping maneuvers, such as securing a moving water glass—a task that presents considerable challenges within traditional visual servoing techniques. Inspecting the dynamic recovery event, we notice that our framework encodes the geometric constraints within the window in a fashion that during the pouring process if the bowl is moved too far from the cup (Case 2 in Section IV), the robot reverts to the pre-pouring state that is the beginning point of the geometric-constrained window. Lastly, achieving successful pouring into a moving bowl is accomplished (as described in Case 3 of Section IV), thereby ensuring that the necessary geometric relationships are preserved throughout the process.

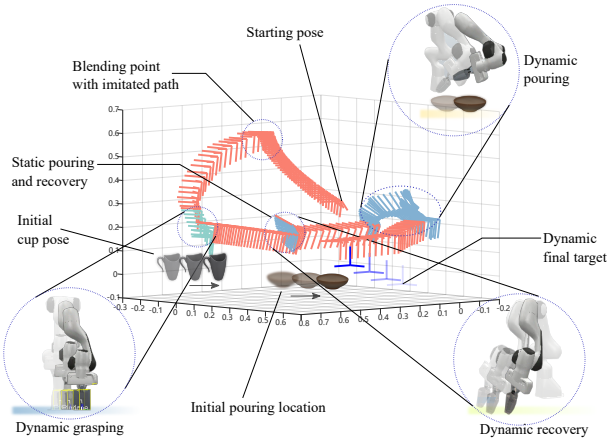


Fig. 5. End effector trajectory of the real-time dynamic experiment.

D. Planning in Clutter: Reactiveness with Adaptability

Taking the block stacking scenario, Fig. 2(d), we evaluate the performance of the proposed framework in the presence of multiple obstacles – affecting both the whole body and the end-effector of the manipulator – while tending to a dynamically changing target goal. These two often contradictory goals are challenging on their own, i.e., addressing multiple obstacles and visual servoing, yet herein we are successfully able to address both concurrently while adhering to geometric prescribed constraints. Fig. 6 illustrates this scenario, highlighting the avoidance strategy while tracking the moving target. This behaviour can be observed by the distances to different links to obstacles and the moving-goal distance (yellow curve). Our algorithm effectively ensures convergence to the target while keeping a safe distance from obstacles, as evidenced by the curves.

E. Ablation Studies

Lastly, we conducted ablation comparative studies against (i) No-Relaxation and (ii) Naive replanning. The former is a reduced version of our method yet tracking a singular trajectory, overlooking task-flexibilities. Similarly, the Re-plan approach focuses on naive replanning without considering the ASR-TSIA method. We deployed the planners in 20 different scenarios, holding the same condition for each planner. The results, shown in Table I, highlight the

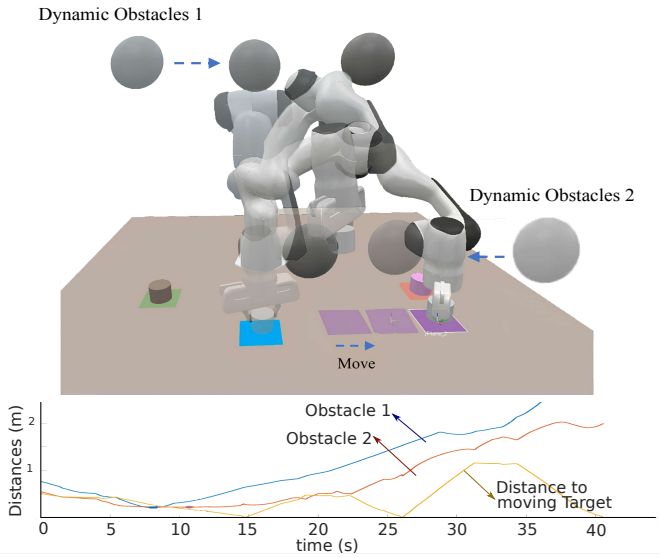


Fig. 6. Planning in clutter scenarios with multiple dynamic obstacles affecting different body links while performing the task.

TABLE I

STATISTICS OF THE THREE METHODS IN COMPARATIVE EXPERIMENTS.

	Total Time	Min. SVD (norm/min)	Dis to joint limit	Success rate
Our	1×	0.23/ 0.17	0.71	0.95
No-relaxation	2.54×	0.21/ 0.08	0.65	0.75
Re-plan	3.5×	0.20/ 0.08	0.55	0.75

relevance of the task-flexibilities as well as the ASR-TSIA algorithm in reducing the task completion time, which is more than 3.5× faster compared to purely replanning. The proposed solution also leads to considerable improvements in singularity avoidance, as expected, and a higher distance to joint limits. The comprehensive planning framework proposed herein therefore results in a much higher success rate.

VII. CONCLUSION

This work introduced a novel user-guided planning framework, tailored for high flexibility and adaptability to dynamic conditions and moving targets while satisfying original constraints. Our approach incorporates the ASR-TSIA module, capturing the inherent flexibility of geometric constraints from multiple demonstrations, and facilitating task adaptation and tracking of moving targets. Additionally, we employ whole-body collision avoidance and manipulability constraints within an HQP-based controller. We have demonstrated the versatility and effectiveness of our framework through a series of experiments. These experiments underscore the framework's ability to adapt to unforeseen scenarios and dynamically changing tasks, thereby marking a significant advancement in the field of robotic systems designed for complex task execution and real-time planning. In the future, we will conduct experiments with complex tasks on different robotic platforms, such as dual-arm robots and mobile manipulation robots, to validate the feasibility of the algorithm.

REFERENCES

[1] R. Laha, R. Sun, W. Wu, D. Mahalingam, N. Chakraborty, L. F. Figueredo, and S. Haddadin, "Coordinate invariant user-guided constrained path planning with reactive rapidly expanding plane-oriented escaping trees," in *Inter. Conf. Robotics & Autom. (ICRA)*, 2022.

[2] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug, "Interactive, collaborative robots: Challenges and opportunities." in *IJCAI*, 2018.

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Rob. Auton. Syst.*, 2009.

[4] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *IEEE Inter. Conf. Robotics & Autom. (ICRA)*, 2002.

[5] C. Chen, C. Yang, C. Zeng, N. Wang, and Z. Li, "Robot learning from multiple demonstrations with dynamic movement primitive," in *2017 2nd Int. Conf. Adv. Robot. Mechatron. (ICARM)*, 2017.

[6] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Trans. Robot.*, 2011.

[7] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, in *Adv. Neural Inf. Process. Syst.*, 2013.

[8] ——, "Probabilistic movement primitives," *Adv. Neural Inf. Process. Syst.*, 2013.

[9] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks," *Auton. Robots*, 2017.

[10] G. Maeda, M. Ewerton, R. Lioutikov, H. Ben Amor, J. Peters, and G. Neumann, "Learning interaction for collaborative tasks with probabilistic movement primitives," in *2014 IEEE-RAS Int. Conf. Humanoid Robots*, 2014.

[11] R. Laha, A. Rao, L. Figueredo, Q. Chang, S. Haddadin, and N. Chakraborty, "Point-to-point path planning based on user guidance and screw linear interpolation," in *Proc. ASME Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf. (IDETC/CIE)*, 2021.

[12] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *Int. J. Robot. Res.*, 2002.

[13] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Autom.*, 1996.

[14] E. G. Ribeiro, R. de Queiroz Mendes, and V. Grassi Jr, "Real-time deep learning approach to visual servo control and grasp detection for autonomous robotic manipulation," *Rob. Auton. Syst.*, 2021.

[15] C. De Farias, M. Adjigble, B. Tamadazte, R. Stolkin, and N. Marturi, "Dual quaternion-based visual servoing for grasping moving objects," in *2021 IEEE 17th Int. Conf. Autom. Sci. Eng. (CASE)*., 2021.

[16] L. F. C. Figueredo, "Kinematic control based on dual quaternion algebra and its application to robot manipulators," Ph.D. dissertation, University of Brasilia, Brazil, 2016.

[17] H. T. Kussaba, L. F. C. Figueredo, B. V. Adorno, and J. Y. Ishihara, "Hybrid kinematic control for rigid body pose stabilization using dual quaternions," *J. Franklin Inst.*, 2017.

[18] B. V. Adorno, "Robot Kinematic Modeling and Control Based on Dual Quaternion Algebra – Part I: Fundamentals - hal-01478225," 2017.

[19] X. Yang, H. Wu, Y. Li, S. Kang, and B. Chen, "Computationally efficient inverse dynamics of a class of six-dof parallel robots: Dual quaternion approach," *J. Intell. Robot. Syst.*, 2019.

[20] J. M. Selig, *Geometric Fundamentals of Robotics*, 2nd ed. Springer-Verlag New York Inc., 2005.

[21] S. Han and O. A. Bauchau, "On the global interpolation of motion," *Comput. Methods Appl. Mech. Eng.*, 2018.

[22] M. Žefran, V. Kumar, and C. Croke, "Metrics and connections for rigid-body kinematics," *Int. J. Robot. Res.*, 1999.

[23] J. Pardos-Gotor, *Screw theory in robotics: an illustrated and practicable introduction to modern mechanics*. CRC Press, 2021.

[24] B. Busam, T. Birdal, and N. Navab, "Camera pose filtering with local regression geodesics on the riemannian manifold of dual quaternions," in *2017 IEEE Int. Conf. Comput. Vis. (ICCV)*., 2017.

[25] F. Allmendinger, S. Charaf Eddine, and B. Corves, "Coordinate-invariant rigid-body interpolation on a parametric C1 dual quaternion curve," *Mech. Mach. Theory.*, 2018.

[26] G. E. Farin, *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002.

[27] Z. Šír and B. Jüttler, "On de casteljau-type algorithms for rational bézier curves," *J. Comput. Appl. Math.*, 2015.

[28] K. Shoemake, "Animating rotation with quaternion curves," in *Proc. 12th Annu. Conf. Comput. Graph. Interact. Tech.*, 1985.

[29] R. Laha, M. Becker, J. Vorndamme, J. Vrabel, L. F. Figueredo, M. A. Müller, and S. Haddadin, "Predictive multi-agent based planning and landing controller for reactive dual-arm manipulation," *IEEE Trans. Robot.*, 2023.

[30] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic motion planning for robots in partially unknown environments," *IFAC*, 2011.

[31] P. Praveena, D. Rakita, B. Mutlu, and M. Gleicher, "User-guided offline synthesis of robot arm motion from 6-dof paths," in *2019 Int. Conf. Robot. Autom. (ICRA)*., 2019.

[32] S. Zhou, C. Shen, F. Pang, Z. Chen, J. Gu, and S. Zhu, "Position-based visual servoing control for multi-joint hydraulic manipulator," *J. Intell. Robot. Syst.*, 2022.

[33] L. Figueredo, B. V. Adorno, J. Y. Ishihara, and G. Borges, "Switching strategy for flexible task execution using the cooperative dual task-space framework," in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014.

[34] M. M. Marinho, B. V. Adorno, K. Harada, and M. Mitsuishi, "Dynamic active constraints for surgical robots using vector-field inequalities," *IEEE Trans. Robot.*, 2019.

[35] B. V. Adorno and M. M. Marinho, "Dq robotics: A library for robot modeling and control," *IEEE Robot. Autom. Mag.*, 2020.