Robotics: Science and Systems 2024
Delft, Netherlands, July 15-July 19, 2024

Task Adaptation in Industrial Human-Robot Interaction: Leveraging Riemannian Motion Policies

Mike Allenspach, Michael Pantic, Rik Girod, Lionel Ott and Roland Siegwart
All authors are with the Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland.
{amike, mpantic, brik, lott, rsiegwart}@ethz.ch

***Abstract*—In real-world industrial environments, modern robots often rely on human operators for crucial decision-making and mission synthesis from individual tasks. Effective and safe collaboration between humans and robots requires systems that can adjust their motion based on human intentions, enabling dynamic task planning and adaptation. Addressing the needs of industrial applications, we propose a motion control framework that (i) removes the need for manual control of the robot's movement; (ii) facilitates the formulation and combination of complex tasks; and (iii) allows the seamless integration of human intent recognition and robot motion planning. For this purpose, we leverage a modular and purely reactive approach for task parametrization and motion generation, embodied by Riemannian Motion Policies. The effectiveness of our method is demonstrated, evaluated, and compared to a representative state-of-the-art approach in experimental scenarios inspired by realistic industrial Human-Robot Interaction settings.**

I. Introduction

With continuous research in artificial intelligence and automation, recent works have demonstrated the potential of autonomous mobile robots in various industrial applications, including infrastructure inspection [1, 2], construction [3, 4], and physical interaction [5, 6]. However, modern robots often still lack the decisional autonomy required for deployment in real-world scenarios with uncertain operating conditions. This encompasses challenges such as disturbed or unknown task hierarchies (e.g., determining which object to pick up first) and addressing unexpected situations (e.g., handling failed measurements at inspection targets).

Therefore, operational safety is typically guaranteed by a human operator that provides situational awareness, reasoning, and problem-solving skills. Apart from safety considerations, combining the complementary capabilities of humans and robotic systems is beneficial to the overall task performance and completion time [7].

Augmenting a (partially) autonomous system with real-time human inputs can be realized in different ways, ranging from low-level manual control of the robot's movements [8, 9] to high-level control by indicating the current mission objective [10]. Although the former offers complete motion control, the latter tends to be less strenuous for the operator during repetitive and precision-demanding tasks [11]. Additionally, recent investigations in [9, 12] indicate that manual control of robots is inefficient, especially for systems involving a high number of controllable degrees of freedoms (DoFs).

As such, our objective is to develop an exclusively autonomous motion control framework tailored to industrial applications. The goal is to reduce the operator's workload by eliminating manual robot motion generation while still preserving control authority. Inspired by recent advances in robot motion generation, we use Riemannian Motion Policies (RMPs) [13] as a purely local reactive motion generation framework that provides a simple yet powerful, mathematically and geometrically consistent [14] task specification. A motivating example that showcases the desired functionality of the framework is illustrated in Figure 1.

Fig. 1: A motivating application for this work is obstacle grasping with unknown task hierarchy. The robot possesses motion policies for handling both objects but initially prioritizes the orange one for pickup. However, the human operator envisions a different order and issues corrective commands via a gamepad to convey their desired direction of motion. The task is dynamically adapted to pick up the blue object instead.

A. Related Work

The alignment of robot motions with human intentions has gained increasing interest in the research community, particularly in the context of safe deployment of autonomous systems in industrial settings [5, 7, 11]. Throughout the extensive literature, a consistent assumption is that the human operator has a particular reward function in mind, representing their preferences for mission execution. In this context, it is crucial to note that the scope of industrial HRI significantly differs from more human-centered research fields, such as household or rehabilitation robotics. There, the emphasis lies in optimizing execution details for human comfort [15], considering factors like distance to humans and obstacles [16, 17], based on expert demonstrations of a specific task [18–20]. Con-

versely, industrial HRI focuses on simplifying ad-hoc dynamic task planning and adaptability. While the constituting tasks of industrial applications are commonly predefined [13], their specific order may be unknown or evolve in real-time based on changing operational requirements. Two sets of issues are tackled by inferring the unknown reward from human actions; compliance of the robot's motion to the human input [21], and active reconfiguration of the task hierarchy.

The unknown human reward function is commonly formulated as a linear combination of individual task features [22]. As such, inferring the reward function simplifies to determining the appropriate relative weights. Corresponding data-driven methods based on Neural Networks [23–25], as well as offline [26] and online Reinforcement Learning (RL) [27, 28] have been derived. Despite their success in accomplishing complex tasks, such data-driven approaches generally require a (potentially large) amount of expert demonstrations and are highly sensitive to changes in the mission setup, robotic system, and environment.

Alternatively, model-based methods remove the need for prior data collection, instead only observing the real-time user input. Initial works concentrated on grasping scenarios, where task features corresponded to the objects to be picked up [29–31]. Based on the user input history, the likelihood of each object is continuously updated until the system is confident about the next object to be picked. In addition to being rather slow, the proposed methods often require approximations and simplification for tractability. Instead, [32] uses dynamical systems for task adaptation, enabling the parametrization of generic tasks and guaranteeing stability.

Many of the existing approaches have limitations regarding their applicability for generic industrial applications. Firstly, the manual control they typically employ is inefficient [9, 12], inspiring an *exclusively autonomous* motion control framework. Secondly, the presented methodologies lack the ability to infer independent task features, emphasizing the need for a more *modular* approach. Thirdly, formulating all task features in a single coordinate frame or manifold is often challenging, motivating a *purely local* task parametrization.

B. Contributions

To address the identified limitations of existing task adaptation approaches, we propose an *exclusively autonomous* robot motion control framework that relies on *highly modular* and *purely local* reactive motion policies. A state-of-the-art implementation is provided by RMPs [13], which offer high-computational efficiency [33], inherent system agnosticism and robustness [34], as well as a well-defined geometric-mathematical structure [14].

As such, the main contributions of this work are:

- Design of an exclusively autonomous motion control framework for dynamic task planning and adaptation;
- Demonstration and evaluation of the proposed approach's effectiveness;
- Discussion and comparison of the proposed approach to a representative state-of-the-art implementation [32].

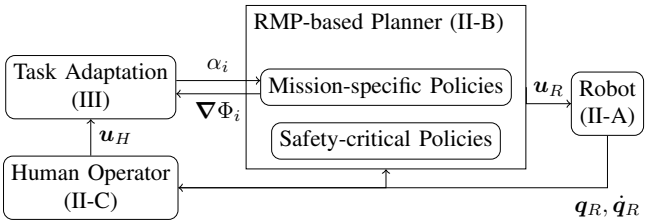

Fig. 2: Motion control framework for dynamic task adaptation: The human operator observes the robot's motion $\dot{\boldsymbol{q}}_R$ and issues corrective commands $\boldsymbol{u}_H$, initiating the adaptation of the current task and corresponding adjustment of the planned motion in the RMP planner.

In the following sections, we present the main building blocks of our system (Section II), describe how task adaption is realized through the inferred re-weighting of individual motion policies (Section III), introduce our experimental setup (Section IV), and perform a quantitative evaluation of the proposed approach (Section V).

II. System Overview

In this section, we introduce the main elements of the system and provide a succinct problem definition. The main components are depicted in Figure 2, comprising a i) *robot* whose motion is guided by a ii) *RMP-based planner*, and a iii) *human operator* whose input influences certain elements of the planner.

A. Robot

Our method is robot-agnostic, however in this study we use a robotic arm with full pose controllability at the end-effector. Therefore, without loss of generality (WLOG), we consider the robot's end-effector configuration space manifold $\mathcal{Q}$ to be $\mathsf{SE}(3)$, where pose and twist are denoted by $\boldsymbol{q}_R \in \mathbb{R}^6$ and $\dot{\boldsymbol{q}}_R \in \mathbb{R}^6$, respectively. We treat the closed-loop dynamics as a double integrator $\ddot{\boldsymbol{q}}_R = \boldsymbol{u}_R$, where the virtual input $\boldsymbol{u}_R \in \mathbb{R}^6$ is calculated by the motion planner and represents the instantaneous reference acceleration to be executed by the robot.

Remark. *The choice of generalized coordinates and configuration space inherently arises from the type of robot and control scheme. However, the RMP motion planner facilitates straightforward transformations between different spatial manifolds (see Section II-B) and handles joint or actuator limits naturally [13]. Consequently, the system model is agnostic to specific robot configurations, including for example underactuated, joint- or pose-controlled platforms.*

B. Riemannian Motion Policy (RMP)-based Planner

RMPs decompose complex robot motion generation problems into several independent policies. Each policy operates within a designated manifold, where the associated problem is easiest to solve. Below is a concise overview of key attributes and operators as outlined in [13, 14].

1) Policy Definition: The essential building block, a motion policy, is defined by two elements: a state-dependent acceleration $\boldsymbol{f}$ and an associated Riemannian metric $\boldsymbol{A}$, both defined on a dedicated manifold $\mathcal{X}$. Formally, we define the manifold $\mathcal{X}$ of dimensionality $d_{\mathcal{X}}$ with generalized coordinates $\boldsymbol{x} : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{X}}}$ and $\dot{\boldsymbol{x}} \in \mathbb{R}^{d_{\mathcal{X}}}$. A policy on this manifold is then denoted by the tuple $\left(\boldsymbol{f}^{\mathcal{X}}, \boldsymbol{A}^{\mathcal{X}}\right)$, with the state-dependent acceleration $\boldsymbol{f}^{\mathcal{X}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) : \mathbb{R}^{d_{\mathcal{X}}} \times \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}^{d_{\mathcal{X}}}$ and the smoothly varying, positive semi-definite Riemannian metric $\boldsymbol{A}^{\mathcal{X}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) : \mathbb{R}^{d_{\mathcal{X}}} \times \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}^{d_{\mathcal{X}} \times d_{\mathcal{X}}}_{\geq 0}$.

While the specific form of $\boldsymbol{f}^{\mathcal{X}}$ varies between policies, stability considerations necessitate a structured expression:

$$\boldsymbol{f}^{\mathcal{X}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = -\boldsymbol{\nabla}\Phi^{\mathcal{X}}(\boldsymbol{x}) - \beta\dot{\boldsymbol{x}}, \tag{1}$$

where $\beta \in \mathbb{R}_{\geq 0}$ is a damping coefficient. This formulation guarantees asymptotically stable minimization of the underlying smooth potential function $\Phi^{\mathcal{X}} : \mathbb{R}^{d_{\mathcal{X}}} \to \mathbb{R}_{\geq 0}$ [13].

Remark. *Despite the seemingly restrictive nature of this formulation, Φ enables the expression of intricate motions [35].*

The form of the metric $\boldsymbol{A}^{\mathcal{X}}$ is not standardized. Rather, it permits the weighting of individual policies relative to others, directionally or axis-wise.

Remark. *As outlined in [13], directional weighting can, for example, improve obstacle avoidance. Compared to the superposition of collision controllers or potential fields, considering directionality ensures that the robot smoothly navigates around obstacles, preventing sudden stops and unintuitive behaviors.*

2) Manifold Mapping: To locally map policies from an arbitrary manifold $\mathcal{X}$ to the configuration space $\mathcal{Q}$, we define the map $\boldsymbol{\psi}^{\mathcal{Q}}_{\mathcal{X}} : \mathbb{R}^6 \to \mathbb{R}^{d_{\mathcal{X}}}$ and the corresponding analytic Jacobian $\boldsymbol{J}^{\mathcal{Q}}_{\mathcal{X}} \in \mathbb{R}^{d_{\mathcal{X}} \times 6}$ as:

$$\boldsymbol{x} = \boldsymbol{\psi}^{\mathcal{Q}}_{\mathcal{X}}(\boldsymbol{q}) \qquad\qquad \boldsymbol{J}^{\mathcal{Q}}_{\mathcal{X}} = \left.\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{q}}\right|_{\boldsymbol{q}_R} \tag{2}$$

The original policy can then be transformed into an equivalent configuration space policy using the *pull-back* operator defined in [13]:

$$\begin{aligned}\left(\boldsymbol{f}^{\mathcal{Q}}, \boldsymbol{A}^{\mathcal{Q}}\right) &= pull_{\boldsymbol{\psi}}\left(\left(\boldsymbol{f}^{\mathcal{X}}, \boldsymbol{A}^{\mathcal{X}}\right)\right) \\ &:= \left(\left(\boldsymbol{J}^{\mathcal{Q}\top}_{\mathcal{X}} \boldsymbol{A}^{\mathcal{X}} \boldsymbol{J}^{\mathcal{Q}}_{\mathcal{X}}\right)^{\dagger} \boldsymbol{J}^{\mathcal{Q}\top}_{\mathcal{X}} \boldsymbol{A}^{\mathcal{X}} \boldsymbol{f}^{\mathcal{X}}, \boldsymbol{J}^{\mathcal{Q}\top}_{\mathcal{X}} \boldsymbol{A}^{\mathcal{X}} \boldsymbol{J}^{\mathcal{Q}}_{\mathcal{X}}\right),\end{aligned} \tag{3}$$

where † is the matrix pseudo-inverse.

3) Policy Combination: Given a set of policies $\left(\boldsymbol{f}^{\mathcal{Q}}_i, \boldsymbol{A}^{\mathcal{Q}}_i\right)$, $i = 1, \ldots, N$ in configuration space, we define the RMP addition operator $\boxplus$ to compute the overall motion policy $\left(\boldsymbol{f}^{\mathcal{Q}}_{tot}, \boldsymbol{A}^{\mathcal{Q}}_{tot}\right)$ as a metric-weighted average [13]:

$$\begin{aligned}\left(\boldsymbol{f}^{\mathcal{Q}}_{tot}, \boldsymbol{A}^{\mathcal{Q}}_{tot}\right) &= \boxplus_i \left(\boldsymbol{f}^{\mathcal{Q}}_i, \boldsymbol{A}^{\mathcal{Q}}_i\right) \\ &:= \left(\left(\sum_i \boldsymbol{A}^{\mathcal{Q}}_i\right)^{\dagger} \sum_i \boldsymbol{A}^{\mathcal{Q}}_i \boldsymbol{f}^{\mathcal{Q}}_i, \sum_i \boldsymbol{A}^{\mathcal{Q}}_i\right).\end{aligned} \tag{4}$$

C. Human Operator

We model the human operator input as a vector on the manifold of admissible human inputs $\mathcal{H}$. Formally, $\mathcal{H}$ is of dimensionality $d_{\mathcal{H}}$ and admits generalized coordinates $\boldsymbol{h} : \mathcal{H} \to \mathbb{R}^{d_{\mathcal{H}}}$. Note that we permit $\mathcal{H}$ to have arbitrary topology and dimensionality, maximizing the utilization of application geometry for an intuitive user interface (see Section IV).

The human input signal $\boldsymbol{u}_H \in \mathbb{R}^{d_{\mathcal{H}}}$ serves as a corrective action aimed at achieving the optimal robot motion. In other words, the operator observes the robot state $\boldsymbol{h}_R = \boldsymbol{\psi}^{\mathcal{Q}}_{\mathcal{H}}(\boldsymbol{q}_R) \in \mathbb{R}^{d_{\mathcal{H}}}$ and motion $\dot{\boldsymbol{h}}_R = \boldsymbol{J}^{\mathcal{Q}}_{\mathcal{H}} \dot{\boldsymbol{q}}_R \in \mathbb{R}^{d_{\mathcal{H}}}$ and indicates the direction and magnitude adjustments required to achieve their desired task.

D. Problem Formulation

1) Task Parametrization: Generic industrial applications can be decomposed into multiple tasks, each defined by a set of inherent features to optimize. In this work, we parametrize tasks as a linear combination of RMPs, each dedicated to optimizing one specific task feature. Mathematically, this is expressed by modeling all features $i = 1, \ldots, N$ as potential functions $\Phi^{\mathcal{X}_i}_i$ in a designated manifold $\mathcal{X}_i$. Optimizing these task features using RMPs then becomes straightforward through the definition of a set of policies $\left(\boldsymbol{f}^{\mathcal{X}_i}_i, \boldsymbol{A}^{\mathcal{X}_i}_i\right)$ with accelerations following Equation (1). We categorize policies as either (i) *safety*-critical (e.g., collision avoidance, workspace limits, joint constraints), with metrics $\boldsymbol{A}^{\mathcal{X}_i}_i$ designed to dominate all other policies in critical conditions (see [13, 14]), or (ii) *mission*-specific (e.g., targets to inspect, objects to pick up), allowing for dynamic adjustments in priorities as the task adapts by gradual modifications of the relevant metrics $\boldsymbol{A}^{\mathcal{X}_i}_i$.

Remark. *While motion policies must be tailored to specific mission conditions, simple attractor and repulsor RMPs often suffice for describing even complex behaviors [34]. These policies, characterized by well-defined potential functions, can be repurposed by simply adjusting the extremum point, e.g. based on task feature and obstacle locations. Independent of the reuse of policies, the proposed adaptation methodology (see Section III) remains application-agnostic, requiring RMPs to adhere only to the structured expression in Equation* (1).

For dynamic task adaptation, we allow the humans to control the system by indirectly scaling the metric of the N_{mis} mission-specific policies through their inputs, while the $N_{saf} = N - N_{mis}$ safety-critical ones remain unaffected. Restricting human intervention solely to mission aspects not only simplifies task adaptation but also prevents accidental overwriting of safety features [36]. Considering this, the robot motion reference $\boldsymbol{u}_R$ is given in configuration space $\mathcal{Q}$ as:

$$(\boldsymbol{u}_R, \cdot) = \boxplus_{i=1}^{N_{mis}} \left(\boldsymbol{f}^{\mathcal{Q}}_i, \alpha_i \boldsymbol{A}^{\mathcal{Q}}_i\right) + \boxplus_{i=N_{mis}+1}^{N} \left(\boldsymbol{f}^{\mathcal{Q}}_i, \boldsymbol{A}^{\mathcal{Q}}_i\right), \tag{5}$$

where the scaling values $\alpha_i \in [0, 1]$ specify the human intended task and are inferred based on their corrective action $\boldsymbol{u}_H$ by the proposed task adaptation (see Section III). Here, all policies are pulled to $\mathcal{Q}$ according to Equation (3), evaluated

at the robot's current pose $\boldsymbol{q}_R$ and twist $\dot{\boldsymbol{q}}_R$, and finally aggregated following Equation (4).

Remark. *In line with the fully autonomous setup motivated in Section I, it is crucial to note the absence of direct manual control over robot motion by the user. Instead, user influence is indirect via metric scale adjustments, eliminating the need for designing suitable arbitration laws between human and planner [29]. Additionally, robot motion is more deterministic, driven by globally defined and provably stable motion policies.*

2) Human Reward: Given our choice of task parametrization (see Section II-D1), each motion policy in Equation (5) fundamentally optimizes the associated task feature. Therefore, scaling the former's metrics by α_i is equivalent to adjusting the latter's relative importance, enabling the modeling of the human objective $r^{\mathcal{H}} : \mathbb{R}^{d_{\mathcal{H}}} \to \mathbb{R}_{\leq 0}$ in the input manifold $\mathcal{H}$ as a linear combination of all task feature potentials [22]:

$$r^{\mathcal{H}}(\boldsymbol{h}_R) = -\sum_{i=1}^{N_{mis}} \alpha_i \Phi_i^{\mathcal{H}}(\boldsymbol{h}_R) =: -\Phi_{des}^{\mathcal{H}}(\boldsymbol{h}_R) \tag{6}$$

Following Section II-C, the operator issues corrective actions $\boldsymbol{u}_H$ to optimize the robot configuration with respect to this reward function. In practical terms, this involves achieving a robot velocity $\dot{\boldsymbol{h}}_R$ aligned with the gradient of $r^{\mathcal{H}}$, i.e., along the negative gradient of the desired potential $-\boldsymbol{\nabla}\Phi_{des}^{\mathcal{H}}(\boldsymbol{h}_R)$. This implies that the human effectively indicates the desired potential function gradient, which is a crucial postulate for the task adaptation discussed in Section III.

Remark. *Users could adjust the policy scaling directly to convey their desired objective, but such explicit communication is often ineffective [37, 38]. This motivates the adoption of a more implicit form of information exchange, like the natural indication of desired robot motion pursued here.*

III. Task Adaptation

The individual mission-specific policies are weighted and combined based on the currently desired task, which is dynamically inferred from the user's input. Our proposed adaptation approach involves two steps: i) determining policy likelihood and ii) scaling policy importance. We determine the *policy likelihood* of all mission-specific RMPs by evaluating how well they match the observed human input. The *policy scaling* step then prioritizes policies with likely and non-conflicting features. In the following, both steps are presented in detail. A conceptual 2D example is shown in Figure 3.

A. Policy Likelihood

1) Human-desired Motion and Likelihood Formulation: As outlined in Section II-D, the human corrective input $\boldsymbol{u}_H$ provides insights into their desired potential gradient $\boldsymbol{\nabla}\Phi_{des}^{\mathcal{H}}$. Adopting a modeling approach for human-interactive trajectory deformation [16, 39], this relationship is formalized as:

$$-\boldsymbol{\nabla}\Phi_{des}^{\mathcal{H}} = \dot{\boldsymbol{h}}_R + \boldsymbol{K}\boldsymbol{u}_H, \tag{7}$$

where $\boldsymbol{K} \in \mathbb{R}_{>0}^{d_{\mathcal{H}} \times d_{\mathcal{H}}}$ is a scaling matrix.

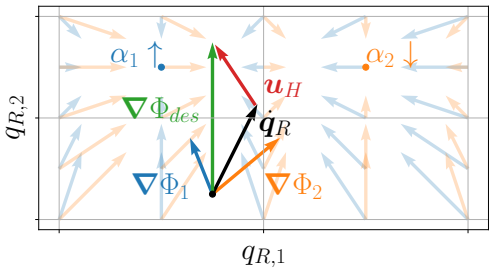

Fig. 3: A 2D task adaptation example, resembling a top-down view of the scene in Figure 1: The human-desired direction of motion $\boldsymbol{\nabla}\Phi_{des}$ aligns better with the first policy $\boldsymbol{\nabla}\Phi_1$. The task adaptation prioritizes it $\alpha_1 \uparrow$ over the second one $\alpha_2 \downarrow$.

Remark. *If $\boldsymbol{u}_H$ is normalized, the matrix $\boldsymbol{K}$ represents the maximum permissible human velocity adjustment. This intuitive interpretation simplifies the tuning process.*

We then define the likelihood $P_i \in [0, 1]$ of each mission-specific RMP $i = \{1, \ldots, N_{mis}\}$ as:

$$P_i := P\left(\boldsymbol{\nabla}\Phi_{des}^{\mathcal{H}} \middle| \left(\boldsymbol{f}_i^{\mathcal{H}}, \boldsymbol{A}_i^{\mathcal{H}}\right)\right) \cdot P\left(\left(\boldsymbol{f}_i^{\mathcal{H}}, \boldsymbol{A}_i^{\mathcal{H}}\right)\right), \tag{8}$$

where the first factor represents the conditional likelihood of achieving the human-desired gradient with this particular policy, and the second term is the policy prior.

2) Conditional Likelihood $P\left(\boldsymbol{\nabla}\Phi_{des}^{\mathcal{H}} \middle| \left(\boldsymbol{f}_i^{\mathcal{H}}, \boldsymbol{A}_i^{\mathcal{H}}\right)\right)$: The conditional likelihood quantifies the disparity between the human-desired and policy induced potential gradient. Assuming that the human primarily indicates the desired direction of motion rather than the actual speed (see Section II-D), the conditional likelihood is modeled based on the alignment angle between these two gradients. We therefore measure the cosine similarity which lies in $[-1, 1]$, utilizing the inner product and norm induced by the policy metric, and project it onto $[0, 1]$:

$$\text{cos\_sim} = \frac{< \boldsymbol{\nabla}\Phi_{des}^{\mathcal{H}}, \boldsymbol{\nabla}\Phi_i^{\mathcal{H}} >_{\boldsymbol{A}_i^{\mathcal{H}}}}{\left|\left|\boldsymbol{\nabla}\Phi_{des}^{\mathcal{H}}\right|\right|_{\boldsymbol{A}_i^{\mathcal{H}}} \cdot \left|\left|\boldsymbol{\nabla}\Phi_i^{\mathcal{H}}\right|\right|_{\boldsymbol{A}_i^{\mathcal{H}}}} \tag{9}$$

$$P\left(\boldsymbol{\nabla}\Phi_{des}^{\mathcal{H}} \middle| \left(\boldsymbol{f}_i^{\mathcal{H}}, \boldsymbol{A}_i^{\mathcal{H}}\right)\right) = \frac{1}{2}\left(1 + \text{cos\_sim}\right) \tag{10}$$

3) Policy Prior $P\left(\left(\boldsymbol{f}_i^{\mathcal{H}}, \boldsymbol{A}_i^{\mathcal{H}}\right)\right)$: Unlike the conditional likelihood, the policy prior is not influenced by the human-desired direction of motion. Instead, it incorporates more intrinsic factors associated with the policy. We take into account two aspects: i) the optimality of the robot's current state with respect to the policy's underlying task feature, and ii) the policy's metric which encodes its relative importance compared to all other polices.

We denote $\boldsymbol{h}_{R,i}^{\star}$ as the optimal state of each task feature expressed in the human manifold $\mathcal{H}$, i.e., $\boldsymbol{h}_{R,i}^{\star} = \text{argmin}_{\boldsymbol{h}_R} \Phi_i^{\mathcal{H}}(\boldsymbol{h}_R)$. Inspired by the goal detection in [29], we formulate the policy prior by considering the distance vector $\boldsymbol{d}_i = \boldsymbol{h}_R - \boldsymbol{h}_{R,i}^{\star} \in \mathbb{R}^{d_{\mathcal{H}}}$ between the current and optimal state. This essentially creates an artificial region of attraction for the different policies.

The policy metric should account for the directional and axis-specific weighting properties (see Section II-B). To do

so, we use the spectral decomposition of $\boldsymbol{A}_i^{\mathcal{H}}$ to independently consider each principal axis and its associated weight in the calculation of $P\left(\left(\boldsymbol{f}_i^{\mathcal{H}}, \boldsymbol{A}_i^{\mathcal{H}}\right)\right) \in [0,1]$:

$$d_{i,j} = (1 - ||\boldsymbol{u}_H||) < \boldsymbol{v}_{i,j}^{\mathcal{H}}, \boldsymbol{d}_i > \tag{11}$$

$$P\left(\left(\boldsymbol{f}_i^{\mathcal{H}}, \boldsymbol{A}_i^{\mathcal{H}}\right)\right) = \sum_{j=1}^{d_{\mathcal{H}}} \exp\left(-\frac{d_{i,j}^2}{2\lambda_{i,j}}\right) \cdot \frac{\lambda_{i,j}^{\mathcal{H}}}{\sum_k \lambda_{i,k}^{\mathcal{H}}}, \tag{12}$$

where $\boldsymbol{v}_{i,j}^{\mathcal{H}} \in \mathbb{R}^3$ and $\lambda_{i,j}^{\mathcal{H}} \in \mathbb{R}_{\geq 0}$ are the j-th eigenvector and corresponding eigenvalue of $\boldsymbol{A}_i^{\mathcal{H}}$, respectively. The exponential term in Equation (12) maps the projected distance $d_{i,j} \in [0, \infty)$ into a likelihood in the range $[0,1]$, following the well established *maximum entropy* methodology [22]. As such, Equation (12) is a weighted average over the main policy directions, exhibiting increased sensitivity towards dominant directions characterized by high metric eigenvalues.

The projected distance $d_{i,j} \in [0, \infty)$ in Equation (11) is adjusted based on the amount of human input $\boldsymbol{u}_H$. This ensures that the operator maintains full control over the system's motion, preventing the robot from becoming trapped in a policy's region of attraction.

Remark. *The heightened sensitivity, as well as dividing the distance $d_{i,j}$ by the eigenvalue in Equation* (12) *are deliberate design decisions. Firstly, directions emphasized in the construction of the metric $\boldsymbol{A}_i^{\mathcal{H}}$ tend to have a more significant impact. Secondly, high-value metrics generally increase the policy prior due to smaller $d_{i,j}$ compared to low-value ones. The former ensures handling of intra-policy directional weighting, while the latter considers inter-policy relative weighting.*

B. Policy Scaling

Interpreting the proposed task parametrization as a linear combination of RMPs, the current task is uniquely defined by the metric scaling values α_i in Equation (5). These values directly govern the influence of each RMP and its associated task feature on the robot's trajectory. At its core, task adaptation thus involves the adjustment of these coefficients.

The approach, outlined in Algorithm 1, processes policies in decreasing order of likelihood, i.e. giving preference to likely policies. Policies controlling task features that are independent of the current summed-up policy increase in importance. Conversely, policies aligned with the current overall policy are reduced in importance, as other higher-likelihood policies are already controlling the same task features. The individual steps outlined in the algorithm are explained in detail next.

Remark. *Our approach differs from existing literature which executes only the most likely policy [29, 30, 32]. This limits the ability to track independent/orthogonal task features (e.g., maintaining end-effector orientation while moving towards a goal position). As our approach leverages RMPs, we can naturally combine multiple policies into a single coherent one.*

1) Alignment Computation: We use cosine similarity between policy gradients to assess if their task features conflict. Specifically, as indicated in Line 4, we use the cosine similarity between the iterator policy's objective gradient $\nabla\Phi_j^{\mathcal{H}}$ and that of all higher-likelihood policies $\nabla\Phi_{P_i>P_j}^{\mathcal{H}}$ (see Section III-B3). Similar to Equation (9), we utilize the inner product induced by the associated metric $\boldsymbol{A}_{P_i>P_j}^{\mathcal{H}}$, adhering to the principles of Riemannian geometry [13].

2) Scale Update: Depending on the similarity returned by the alignment computation, the metric scales α_i are adjusted. Intuitively, task feature gradients are declared non-conflicting when the alignment angle γ is exactly $90°$ and conflicting otherwise. The actual adjustment of the scales is done gradually to ensure continuity, as required by stability guarantees for RMPs [14]. Fine-tuning the adjustment step α_Δ affects the dynamics of task adaptation. Large values enhance responsiveness to human inputs, while smaller values reduce sensitivity to erroneous operator commands [16], resulting in overall smoother transitions between policies.

3) Cumulative Policy Update: The cumulative policy $\left(\boldsymbol{f}_{P_i>P_j}^{\mathcal{H}}, \boldsymbol{A}_{P_i>P_j}^{\mathcal{H}}\right)$ combines all higher-likelihood polices with scaled metrics. In this context, the $+=$ operator in Line 10 implements the addition of two RMPs according to Equation (4).

Algorithm 1: Policy Scaling

Input: N_{mis} mission-specific policies $\left(\boldsymbol{f}_i^{\mathcal{H}}, \boldsymbol{A}_i^{\mathcal{H}}\right)$ and corresponding likelihoods P_i

Output: policy scales α_i

1 Sort policies in decreasing order of likelihood;

2 Initialize cumulative policy $\left(\boldsymbol{f}_{P_i>P_j}^{\mathcal{H}}, \boldsymbol{A}_{P_i>P_j}^{\mathcal{H}}\right) = (\boldsymbol{0}, \boldsymbol{0})$;

3 **foreach** *policy* $\left(\boldsymbol{f}_j^{\mathcal{H}}, \boldsymbol{A}_j^{\mathcal{H}}\right)$ *in sorted list* **do**

`// 1.) alignment computation`

4 $\gamma = \frac{<\nabla\Phi_{P_i>P_j}^{\mathcal{H}}, \nabla\Phi_j^{\mathcal{H}}>_{\boldsymbol{A}_{P_i>P_j}^{\mathcal{H}}}}{\left|\left|\nabla\Phi_{P_i>P_j}^{\mathcal{H}}\right|\right|_{\boldsymbol{A}_{P_i>P_j}^{\mathcal{H}}} \cdot \left|\left|\nabla\Phi_j^{\mathcal{H}}\right|\right|_{\boldsymbol{A}_{P_i>P_j}^{\mathcal{H}}}}$;

`// 2.) scale update`

5 **if** $\gamma = 0$ **then**

6 $\alpha_j \mathrel{+}= \alpha_\Delta$;

7 **else**

8 $\alpha_j \mathrel{-}= \alpha_\Delta$;

9 Clamp α_j to be in $[0,1]$;

`// 3.) cumulative policy update`

10 $\left(\boldsymbol{f}_{P_i>P_j}^{\mathcal{H}}, \boldsymbol{A}_{P_i>P_j}^{\mathcal{H}}\right) \mathrel{+}= \left(\boldsymbol{f}_j^{\mathcal{H}}, \alpha_j \boldsymbol{A}_j^{\mathcal{H}}\right)$;

C. Stability and Robustness Considerations

In this section, we briefly comment on several stability and robustness-related aspects of the developed methodology.

Firstly, the motion control framework is bounded-input-bounded-output stable since it is driven by asymptotically stable motion policies, and no energy is introduced from human manual control. Secondly, if the human perfectly demonstrates the desired task, the task adaptation converges to

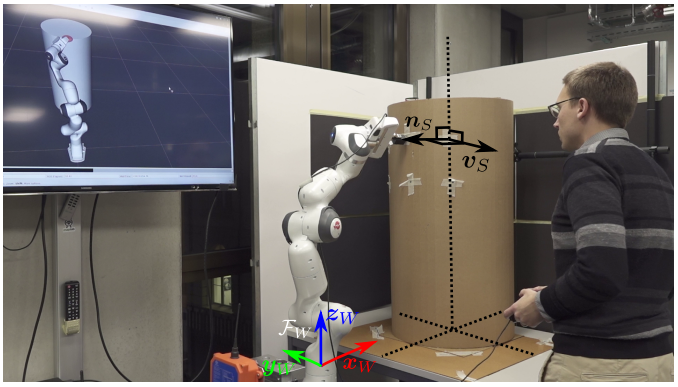

Fig. 4: Experimental setup: cylindrical surface with inspection targets for the Franka Panda arm, user input via Gamepad, and visual feedback in RViz. The inertial world frame $\mathcal{F}_W$, as well as the instantaneous surface normal $\boldsymbol{n}_S$ and tangent $\boldsymbol{v}_S$ are indicated.

the correct scaling values α_i. Finally, once the task adaptation has converged, the robot motion $\boldsymbol{u}_R$ optimizes the human reward $r^{\mathcal{H}}$ in a globally asymptotically stable manner. Together, these properties guarantee the stability and functionality of the overall system. The detailed mathematical analysis of these properties is provided in the Appendix.

In uncertain environments, discrepancies between the safety-critical/mission-specific RMP definitions and real-world conditions may arise. However, task adaptation maintains the stated stability and convergence properties by relying solely on the instantaneous potential gradient of predefined task features, regardless of the precision and accuracy of these features.

Remark. *Failure to align task features with the real-world environment can still hinder mission success, despite robust task adaptation. While we leave specific investigations into robustifying features for future work, the modular nature of our methodology inherently permits real-time adjustment or addition of individual motion policies based on evolving environmental understanding, without modifying the motion planning and task adaptation framework.*

IV. Experimental Setup

In this section, we outline the experimental setup and evaluation scenario for the quantitative analysis presented in Section V. The supplementary video material[1] showcases the setting, as well as demonstrating additional applications.

A. General Mission Description

Our objective is to replicate a non-destructive testing application [40], using a 7DoF Franka Panda arm equipped with a prism-like end-effector imitating the sensor. Inspired by applications in oil tank and wind turbine inspections, the experimental setup features a cylindrical object with six inspection targets on its surface (see Figure 4).

To ensure accurate measurements, the sensor must be positioned in close proximity to an inspection point-of-interest, while maintaining alignment with the surface normal $\boldsymbol{n}_S$.

[1]https://youtu.be/E5SlSKx7pQk

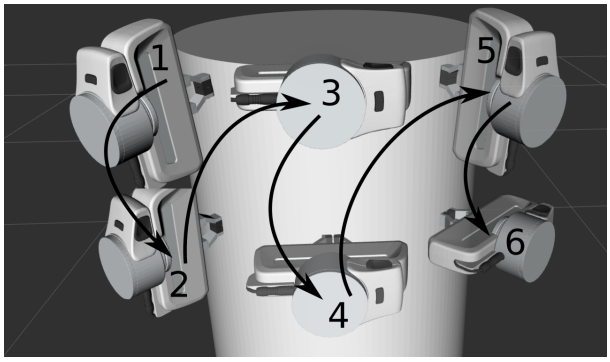

Fig. 5: Desired sequence of inspection targets and sensor rotations.

Additionally, a minimum distance from the object should be ensured for safe operation when not recording data. To interact with the robot, user commands $\boldsymbol{u}_H$ are entered using a Gamepad (Logitech F310), providing 4DoF inputs with continuous joysticks. As the application is intricately tied to the object surface, we define the human input manifold to encompass 2D motion on the surface along $\boldsymbol{z}_W$ and $\boldsymbol{v}_S$, and end-effector rotation around $\boldsymbol{n}_S$. Situational awareness is ensured by allowing the user to maintain line-of-sight with the physical robot or through a digital twin of the setup via a 3D rendered visualization, as depicted in Figure 4.

B. Task Feature Formulation

For this application, we define the following motion policies and matching manifolds to capture relevant task features:

Normal-Keeping RMP *Safety-critical* policy that ensures sensor alignment with $\boldsymbol{n}_S$. Acting on the $\mathcal{X}_1 = \mathsf{SO}(3)$ manifold of end-effector orientations.

Distance-Keeping RMP *Safety-critical* policy that ensures safe distance. Acting on the $\mathcal{X}_2 = \mathbb{R}$ manifold along $\boldsymbol{n}_S$.

Inspection Position RMPs *Mission-specific* policies $\boldsymbol{f}_{pos,i}$, $i \in \{1, \cdots, 6\}$ that pull the end-effector towards the respective inspection target. Acting on the 3-dimensional manifold $\mathcal{X}_3 = \mathbb{R} \times \mathcal{S}^1 \times \mathbb{R}$ encompassing the object surface and distance to it.

Inspection Rotation RMPs *Mission-specific* policies $\boldsymbol{f}_{rot,j}$, $j \in \{1, 2\}$ that rotate the sensor around $\boldsymbol{n}_S$ into a horizontal or vertical orientation (see below). Acting on the $\mathcal{X}_1$ manifold of end-effector orientations.

The specific formulation of these policies is omitted here for brevity but follows the generic target attractor outlined in [13].

C. Evaluation Scenario

The operator receives visual instructions through the 3D visualization, indicating the next position to inspect and a desired tool orientation according to Figure 5. The task is to move the end-effector within $\pm 5\,\mathrm{cm}$ and $\pm 3^\circ$ of the indicated pose, as fast as possible.

We evaluate our methodology both independently and in comparison with the state-of-the-art task adaptation method

in [32], hereafter referred to as *Sota*. This work is deemed representative, as it addresses industrial HRI, allows parametrization of generic tasks, and features a comparable task adaptation setup. By additionally assessing our method against two different forms of purely manual robot control we gain insight into how our exclusively autonomous framework compares in input efficiency. To ensure optimal performance and fairness in comparison, all experiments are conducted by a group of $N = 7$ robotic researchers, generally experienced in operating robotic systems but not familiar or trained on this experimental setup. All policies and adaptation laws run at $100\,\mathrm{Hz}$.

The different methodologies are briefly discussed below:

1) Ours: This method adopts the dynamic task adaptation and motion control framework proposed in this paper. Following Section IV-A, the human input manifold allows motion on the object surface in vertical and tangential directions, as well as end-effector rotation around the surface normal. The robot motion is driven by combining RMPs to maintain normal orientation, ensure safety distance, and move towards inspection targets.

2) 2D Manual Control: This method adopts standard rate control teleoperation, where the joystick controls the robot's velocity. The human input is mapped to the curved surface of the cylinder, which relieves the operators from exact control along $\boldsymbol{n}_S$. Unlike *Ours* however, the robot only moves in response to non-zero user inputs and does not track inspection poses. The task adaptation in Section III is executed in the background for evaluation purposes. Normal- and Distance-Keeping is still ensured by the corresponding RMPs.

3) 3D Manual Control: Similar to the *2D* approach, this method employs a rate control setup to manually control the robot's velocity. However, the operator is now responsible for also controlling the distance to the surface along $\boldsymbol{n}_S$. Task adaptation data is again recorded for evaluation only. Robot autonomy is now limited to orientation of the end-effector in order to remain normal to the surface.

4) Sota: The methodology outlined in [32] represents tasks as 1st-order dynamical systems. This involves the definition of a function that maps the robot's current pose into a velocity that optimizes all corresponding features. The task adaptation mechanism in [32] compares the human-adjusted robot velocity with each task, updating the respective probability depending on how closely they match. Note that robot motion control, task formulation, human interaction, and task adaptation are all designed in the same manifold, namely the robot configuration space $\mathsf{SE}(3)$.

V. Quantitative Evaluation

Based on the setup presented in Section IV, we conduct a quantitative assessment of the proposed methodology. The results and discussion address critical performance aspects, considering the methodology both independently and in comparison with the state-of-the-art task adaptation method in [32].

A. Results

We focus the evaluation of our method and comparison with *Sota* on how quickly the respective adaptation converges to the human-desired task and the amount of user input required to achieve this convergence. To emphasize, convergence refers to the time required for task adaptation (Section III for *Ours*, *2D*, *3D*; or [32] for *Sota*) to identify the correct target, and not completion of the task itself. The convergence times for inspection position and rotation of each task are reported in Figure 6, visualizing the distribution across the $N = 7$ operators for each methodology. In essence, this represents the minimum interaction time required before the robot can autonomously complete the task, which correlates with the expressiveness of human input for the different methodologies.

In terms of user input, statistics for the total human effort needed to complete the mission are provided in Table I. Figure 7 illustrates the accumulated operator commands along the individual joystick axes during the transition from task 2 to 3 (see Figure 5). This transition offers detailed insights into fundamental differences between the four methods and, specifically, between *Ours* and *Sota*.

For completeness, we also report statistics on the task completion time and accuracy, represented as the minimum translational and rotational error, in Table I. Although these metrics are subject to the influence of the robot and tuning of the individual RMPs or dynamic systems, the table offers a high-level comprehensive comparison of autonomous (*Ours*/*Sota*) and manual (*2D*/*3D*) methods. Indeed, the results confirm that pure manual control is generally unsuitable, resulting in increased user effort, longer task completion time and degraded accuracy. More fundamental insights into our methodology are discussed below.

Method	Ours	2D	3D	Sota
$\frac{1}{T}\int_0^T \lVert\boldsymbol{u}_H(t)\rVert^2\,dt$	$\boldsymbol{\mu = 41.15}$	$\mu = 64.84$	$\mu = 49.91$	$\mu = 54.09$
	$\sigma = 11.35$	$\sigma = 8.74$	$\boldsymbol{\sigma = 8.33}$	$\sigma = 15.71$
Task completion time	$\boldsymbol{\mu = 11.19}$	$\mu = 19.98$	$\mu = 27.94$	$\mu = 13.19$
$[s]$	$\boldsymbol{\sigma = 5.24}$	$\sigma = 10.28$	$\sigma = 14.95$	$\sigma = 6.75$
Translation error	$\mu = 0.89$	$\mu = 6.74$	$\mu = 9.87$	$\boldsymbol{\mu = 0.57}$
$[mm]$	$\boldsymbol{\sigma = 0.30}$	$\sigma = 3.88$	$\sigma = 4.45$	$\sigma = 0.40$
Rotation error	$\boldsymbol{\mu = 0.19}$	$\mu = 2.74$	$\mu = 2.56$	$\mu = 0.66$
$[^\circ]$	$\boldsymbol{\sigma = 0.12}$	$\sigma = 1.21$	$\sigma = 1.34$	$\sigma = 0.36$

TABLE I: Mean μ and standard deviation σ of different metrics across all $N = 7$ operators.

B. Discussion

A thorough investigation of the task adaptation convergence time and user inputs allows us to draw several conclusions.

As illustrated in Figure 6, dynamic task adaptation demonstrates significantly faster convergence to the correct task when no manual control is involved. This aligns with observations from Figure 7, indicating that operator commands in *Ours* are highly task oriented compared to *2D* and *3D*. In our approach, all joystick commands are concentrated in the first quadrant, aligning with the optimal task direction and facilitating the rapid identification of the human-desired task. Contrastingly, inputs using the two manual control methods are generally less informative for task adaptation. This discrepancy likely arises from the precision required to maneuver the robot under

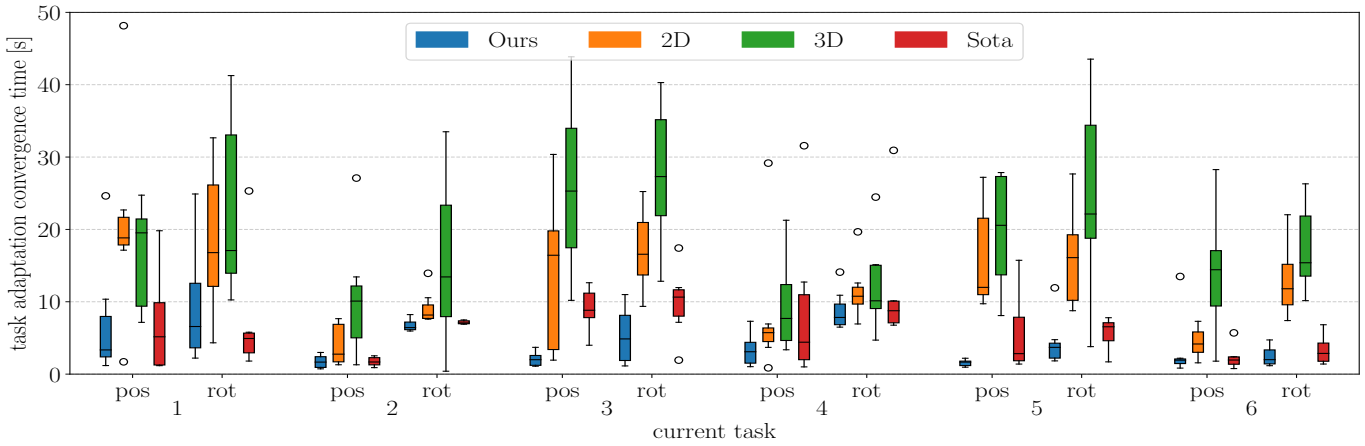

Fig. 6: Task adaptation convergence time for inspection position (pos) and rotation (rot) part of each task across the $N = 7$ operators. Note that task adaptation is non-blocking, running in real-time at the same rate as robot motion control. This enables the robot to remain in motion and move towards the most likely target even prior to convergence.

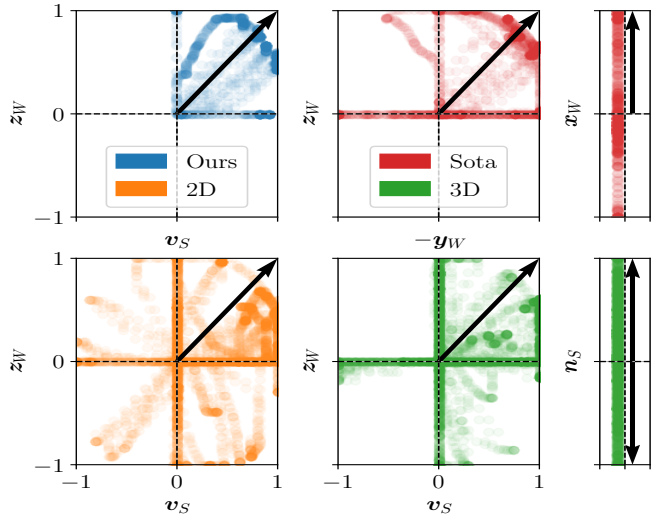

Fig. 7: Translational joystick inputs for transition from task 2 to 3, accumulated over all $N = 7$ operators. Dots correspond to inputs commanded by the users, with opacity corresponding to the occurrence frequency. Labels indicate the mapped robot motion axis while black arrows represent the optimal commands for convergence of the respective task adaptation methodology. Notably, the input manifolds $\mathcal{H}$ for *Ours* and *2D* feature two translational control inputs, while the ones for *3D* and *Sota* feature three.

manual control, leading to numerous small-scale direction-changing corrective inputs. This conclusion strongly supports the use of an *exclusively autonomous* motion control framework.

When comparing with *Sota*, their task adaptation achieves similar convergence rates as ours. However, the required human input is much more intricate. As the human input manifold is the cartesian configuration space $\mathsf{SE}(3)$, the motion needs to be carefully coordinated for the human to be able to correctly indicate the task along the curved surface. This becomes particularly evident when considering the translational control inputs during the transition from task 2 to 3. In *Ours*, where the input manifold $\mathcal{H}$ encompasses the surface, convergence is achieved by a 2D input along $\boldsymbol{v}_S$ and $\boldsymbol{z}_W$, as indicated by the black arrows in Figure 7. In *Sota*'s $\mathsf{SE}(3)$ manifold however, this necessitates coordinated motion along all three principal axes, $\boldsymbol{x}_W$, $-\boldsymbol{y}_W$ and $\boldsymbol{z}_W$. When moving along $-\boldsymbol{y}_W$, providing control input along $\boldsymbol{x}_W$, i.e., pushing the robot towards the surface, is crucial, since the task adaptation otherwise converges on task 5 or 6 due to the ambiguity in motion. Many operators found this challenging, as evidenced by the strong corrective actions along $\boldsymbol{y}_W$ in Figure 7. In fact, oral intervention from the experiment supervisor was often necessary, since the required correction along $\boldsymbol{x}_W$ was not self-evident. Consequently, this transition exhibits the largest median convergence time for *Sota* across all tasks. This observation highlights the advantages of a *purely local* motion planning framework, allowing us to leverage the inherent geometry of applications and design appropriate input manifolds.

Finally, we highlight key differences between our *modular* approach and the full body motion planning used in *Sota*. In *Sota*, each task is expressed as a single $\mathsf{SE}(3)$ dynamical system, responsible for controlling every aspects of robot motion. This means that all desired behaviors, task completion, safety measures, and system constraints have to be formalized within a single function. The design and debugging of these functions becomes notably challenging. Independent testing of individual components is often not feasible, and given that many aspects, e.g. safety considerations, are common across all tasks, redundant definitions are necessary. Indeed, experiments in [32] required prior learning-from-demonstration for dynamical systems that did not follow the simple form of linear or circular motion. This reduces overall adaptability, as even small changes in tasks or the environment necessitate redesigning the entire dynamical system.

On the contrary, modular motion planning, exemplified by RMPs, allows us to represent tasks as a combination of smaller, individually testable policies. Each component addresses a specific challenge and can be geometrically com-

bined with others to realize more complex motion. Such modular approaches provide a more versatile toolbox for various tasks, streamlining the overall design process and enhancing usability. Using the approach presented in this paper, we enable the use of such a modular framework for HRI.

C. Scalability Considerations

Comparing the scalability of our method with *Sota* [32], we focus on complexity in industrial applications arising from non-intuitive task geometries and intricate task features.

The quantitative evaluation highlights the potential for enhanced intuitiveness through shaping the human input manifold $\mathcal{H}$, resulting in superior user interaction compared to *Sota*, or at worst, similar performance when selecting $\mathcal{H}$ the same as theirs.

As task features increase in number and sophistication, designing a monolithic dynamical system for *Sota* becomes increasingly challenging. In contrast, our modular approach maintains simplicity and functional task adaptation by integrating independent and composable policies. Challenges can arise when multiple task features have closely aligned potential gradients, which diminishes the expressiveness of human input and may prolong task adaption time. Possible solutions are the inclusion of exaggerated robot motions to elicit more informative human inputs [41] or using augmented reality techniques [42].

VI. Conclusion

We introduce a motion control framework designed for industrial applications, streamlining dynamic task planning and adaptation to human operator commands. Our method eliminates the need for direct manual control, opting for exclusively autonomous motion planning without restricting human control authority over robot motion. Task parametrization and adaptation leverage purely local and modular motion policies in the form of Riemannian Motion Policies. This enables a straightforward formulation of task features in their native manifold and provides a geometrically consistent way to combine them. To do so, we contribute a novel method to infer user intent directly in the form of a re-weighting of policy metrics, effectively exploiting the mathematical structure of Riemannian Motion Policies.

Quantitative experiments show that our method enables smooth and seamless Human-Robot Interaction. Moreover, we validate the potential of exclusively autonomous frameworks, showing that human manual control tends to increase the convergence time of task adaptation. In comparison to the state-of-the-art, we emphasize the advantages of exploiting the inherent geometry of tasks, simplifying the user interface, as well as the formulation of the tasks themselves. In future work, we aim to enhance the robustness of our framework against inaccuracies in task features within uncertain environments. This entails refining RMPs in real-time based on evolving environmental understanding, as well as the identification of new task features on-the-fly.

References

[1] David Lattanzi and Gregory Miller. Review of Robotic Infrastructure Inspection Systems. *Journal of Infrastructure Systems*, 2017. doi: 10.1061/(ASCE)IS.1943-555X.0000353.

[2] P. Pfändler, K. Bodie, G. Crotta, M. Pantic, R. Siegwart, and U. Angst. Non-destructive corrosion inspection of reinforced concrete structures using an autonomous flying robot. *Automation in Construction*, 2024. doi: https://doi.org/10.1016/j.autcon.2023.105241.

[3] Pascal Egli, Dominique Gaschen, Simon Kerscher, Dominic Jud, and Marco Hutter. Soil-Adaptive Excavation Using Reinforcement Learning. *IEEE Robotics and Automation Letters*, 2022. doi: 10.1109/LRA.2022.3189834.

[4] Hilti. Hilti Jaibot, 2023.

[5] Anibal Ollero, Marco Tognon, Alejandro Suarez, Dongjun Lee, and Antonio Franchi. Past, Present, and Future of Aerial Robotic Manipulators. *IEEE Transactions on Robotics*, 2022. doi: 10.1109/TRO.2021.3084395.

[6] Giuseppe Rizzi, Jen Jen Chung, Abel Gawel, Lionel Ott, Marco Tognon, and Roland Siegwart. Robust Sampling-Based Control of Mobile Manipulators for Interaction With Articulated Objects. *IEEE Transactions on Robotics*, 2023. doi: 10.1109/TRO.2022.3233343.

[7] Canjun Yang, Yuanchao Zhu, and Yanhu Chen. A Review of Human–Machine Cooperation in the Robotics Domain. *IEEE Transactions on Human-Machine Systems*, 2021. doi: 10.1109/THMS.2021.3131684.

[8] Andre Coelho, Harsimran Singh, Konstantin Kondak, and Christian Ott. Whole-Body Bilateral Teleoperation of a Redundant Aerial Manipulator. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. doi: 10.1109/ICRA40945.2020.9197028.

[9] Mike Allenspach, Nicholas Lawrance, Marco Tognon, and Roland Siegwart. Towards 6DoF Bilateral Teleoperation of an Omnidirectional Aerial Vehicle for Aerial Physical Interaction. In *2022 International Conference on Robotics and Automation (ICRA)*, 2022. doi: 10.1109/ICRA46639.2022.9812346.

[10] Justin Storms, Kevin Chen, and Dawn Tilbury. A shared control method for obstacle avoidance with mobile robots and its interaction with communication delay. *The International Journal of Robotics Research*, 2017. doi: 10.1177/0278364917693690.

[11] Mario Selvaggio, Marco Cognetti, Stefanos Nikolaidis, Serena Ivaldi, and Bruno Siciliano. Autonomy in Physical Human-Robot Interaction: A Brief Survey. *IEEE Robotics and Automation Letters*, 2021. doi: 10.1109/LRA.2021.3100603.

[12] Michael Young, Christopher Miller, Youyi Bi, Wei Chen, and Brenna D. Argall. Formalized Task Characterization for Human-Robot Autonomy Allocation. In *2019 International Conference on Robotics and Automation*

(ICRA), 2019. doi: 10.1109/ICRA.2019.8793475.

[13] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian Motion Policies. *arXiv preprint arXiv:1801.02854*, 2018.

[14] Ching-An Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. RMPflow: A Computational Graph for Automatic Motion Policy Generation. In *Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*. Springer, 2020.

[15] Dylan P. Losey, Craig G. McDonald, Edoardo Battaglia, and Marcia K. O'Malley. A Review of Intent Detection, Arbitration, and Communication Aspects of Shared Control for Physical Human–Robot Interaction. *Applied Mechanics Reviews*, 2018. doi: 10.1115/1.4039145.

[16] Dylan P Losey, Andrea Bajcsy, Marcia K O'Malley, and Anca D Dragan. Physical Interaction as Communication: Learning Robot Objectives Online from Human Corrections. *The International Journal of Robotics Research*, 2022.

[17] Dylan P. Losey and Marcia K. O'Malley. Learning the Correct Robot Trajectory in Real-Time from Physical Human Interactions. *J. Hum.-Robot Interact.*, 2019. doi: 10.1145/3354139.

[18] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[19] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative Inverse Reinforcement Learning. *Advances in neural information processing systems*, 29, 2016.

[20] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent Advances in Robot Learning from Demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 2020.

[21] Paolo Franceschi, Nicola Pedrocchi, and Manuel Beschi. Adaptive Impedance Controller for Human-Robot Arbitration based on Cooperative Differential Game Theory. In *2022 International Conference on Robotics and Automation (ICRA)*, 2022. doi: 10.1109/ICRA46639.2022.9811853.

[22] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum Entropy Inverse Reinforcement Learning. In *Proc. AAAI*, 2008.

[23] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum Entropy Deep Inverse Reinforcement Learning. *arXiv preprint arXiv:1507.04888*, 2015.

[24] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Deep Inverse Reinforcement Learning. *arXiv preprint arXiv:1507.04888*, 2015.

[25] Yujiao Cheng, Liting Sun, Changliu Liu, and Masayoshi Tomizuka. Towards Efficient Human-Robot Collaboration With Robust Plan Recognition and Trajectory Prediction. *IEEE Robotics and Automation Letters*, 2020. doi: 10.1109/LRA.2020.2972874.

[26] Zoe Ashwood, Aditi Jha, and Jonathan W. Pillow. Dynamic Inverse Reinforcement Learning for Characterizing Animal Behavior. In *Advances in Neural Information Processing Systems*, 2022.

[27] Nicholas Rhinehart and Kris M. Kitani. First-Person Activity Forecasting with Online Inverse Reinforcement Learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. doi: 10.1109/ICCV.2017.399.

[28] Siddharth Reddy, Anca Dragan, and Sergey Levine. Shared Autonomy via Deep Reinforcement Learning. *Robotics: Science and Systems XIV*, 2018.

[29] Anca D Dragan and Siddhartha S Srinivasa. A policy-blending formalism for shared control. *The International Journal of Robotics Research*, 2013. doi: 10.1177/0278364913490324.

[30] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared Autonomy via Hindsight Optimization for Teleoperation and Teaming. *The International Journal of Robotics Research*, 2018.

[31] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared Autonomy via Hindsight Optimization. *Robotics: Science and Systems XI*, 2015.

[32] Mahdi Khormashahi and Aude Billard. A dynamical system approach to task-adaptation in physical human–robot interaction. In *Autonomous Robots*, 2019. doi: 10.1007/s10514-018-9764-z.

[33] Michael Pantic, Isar Meijer, Rik Bähnemann, Nikhilesh Alatur, Olov Andersson, Cesar Cadena, Roland Siegwart, and Lionel Ott. Obstacle avoidance using raycasting and Riemannian Motion Policies at kHz rates for MAVs. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[34] Christian Lanegger, Michael Pantic, Bähnemann Rik, Siegwart Roland, and Ott Lionel. Chasing Millimeters: Design, Navigation and State Estimation for Precise In-flight Marking on Ceilings. *Autonomous Robots*, 2023. doi: https://doi.org/10.1007/s10514-023-10141-5.

[35] M. Asif Rana, Anqi Li, Harish Ravichandar, Mustafa Mukadam, Sonia Chernova, Dieter Fox, Byron Boots, and Nathan Ratliff. Learning Reactive Motion Policies in Multiple Task Spaces from Human Demonstrations. In *Proceedings of the Conference on Robot Learning*, 2020.

[36] Przemyslaw A. Lasota, Terrence Song, and Julie A. Shah. *A Survey of Methods for Safe Human-Robot Interaction*. Now Foundations and Trends, 2017.

[37] Scott Green, Mark Billinghurst, Xiaoqi Chen, and James Chase. Human-Robot Collaboration: A Literature Review and Augmented Reality Approach In Design. *International Journal of Advanced Robotic Systems*, 2008. doi: 10.5772/5664.

[38] M.A. Goodrich and D.R. Olsen. Seven principles of efficient human robot interaction. In *Proceedings on 2003 IEEE International Conference on Systems, Man and Cybernetics.*, 2003. doi: 10.1109/ICSMC.2003.1244504.

[39] Dylan Losey and Marcia O'Malley. Trajectory Deformations From Physical Human–Robot Interaction. *IEEE Transactions on Robotics*, 2018. doi: 10.1109/TRO.2017.2765335.

[40] U. M. Angst. Challenges and opportunities in corrosion of steel in concrete. *Materials and Structures*, 2018.

[41] Ananth Jonnavittula and Dylan P. Losey. Communicating Robot Conventions through Shared Autonomy. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7423–7429, 2022. doi: 10.1109/ICRA46639.2022.9811674.

[42] James F. Mullen, Josh Mosier, Sounak Chakrabarti, Anqi Chen, Tyler White, and Dylan P. Losey. Communicating Inferred Goals With Passive Augmented Reality and Active Haptic Feedback. *IEEE Robotics and Automation Letters*, 6(4):8522–8529, 2021. doi: 10.1109/LRA.2021.3111055.