Skill-based Shared Control

Christopher E. Mower[1], João Moura[1,2], and Sethu Vijayakumar[1,2]
[1]School of Informatics, The University of Edinburgh, Edinburgh, UK. [2]The Alan Turing Institute, London, UK.

***Abstract*—Performing a number of motion patterns – referred to as *skills* – (e.g., wave, spiral, sweeping motions) during teleoperation is an integral part of many industrial processes such as spraying, welding, and wiping (cleaning, polishing). Maintaining these motions whilst simultaneously avoiding obstacles and traversing complex terrain requires expert operators. In this work, we propose a novel skill-based shared control framework for incorporating the notion of skill assistance to aid novice operators to sustain these motion patterns whilst adhering to environmental constraints. Our shared control method uses streaming joystick data to estimate the model parameters that provide a description of the operator's intention. We introduce a novel parametrization for state and control that combines skill and underlying trajectory models, leveraging a special type of curve known as Clothoids. This new parameterization allows for efficient computation of skill-based short term horizon plans, enabling the use of a Model Predictive Control (MPC) loop. We perform experiments on a hardware mock-up, validating the effectiveness of our method to recognize a switch of intended skill, and showing an improved quality of output motion, even under dynamically changing obstacles. See our accompanying video here: https://youtu.be/TwhsgA6fw6M.**

I. Introduction

Many industrial applications require a human to teleoperate a robotic device, typically under direct control [24], to perform a manipulation task as part of a construction or manufacturing process (e.g., Figs. (1a) and (1b)). During task execution, the operator will often employ various motion patterns to achieve different goals: in concrete spraying, an operator will switch between circular and sweeping motions to regulate the rate of concrete deposition and create a smooth lining [3]; in robot assisted welding, the operator's expertise on the desired weld determine the choice among different weave patterns [7]; in plastering, different patterns yield different moulds [4]; and in the cleaning process of the train's front panels, expert cleaners repeatedly employ spiral brushing motion patterns [21]. These motion patterns, which we will call *skills*, are crucial to the success of the tasks.

Sustaining and smoothly executing these skills over the duration of a task can be difficult, requiring extensive and expensive training regimes, yet critical to performance. For example, in concrete spraying during tunnel construction, appropriately laying concrete is critical for preventing wasted material and collapse. Such critical teleoperation tasks also put considerable cognitive load on the operator. Further, there are significant challenges to ensure the safety of the operators when they operate close to the site (e.g., concrete falls away from the wall after being sprayed, known as rebound [3]). These circumstances make the automation of such operations desirable. However, the crucial role of the operator's domain knowledge for the successful completion of such tasks motivates a human-in-the-loop *shared control* approach.

This research is supported by Costain Group Plc., The Alan Turing Institute, and EPSRC UK RAI Hub for Offshore Robotics for Certification of Assets (ORCA, EP/R026173/1).

(a) (b) (c)

Fig. 1: Industrial tasks such as (a) concrete spraying and (b) welding [10] require highly skilled operators; (c) Lab mock-up on a KUKA LWR in shared control mode; with online MPC based skill estimation and switching.

Many shared control paradigms developed in the literature can be thought of as direct control within a restricted workspace. This property benefits a novice by only allowing them to operate within safe limits – a concept that lets the operator largely dictate motion, under the principle of minimal intervention [6, 28]. While the principle achieves the crucial goal of minimally disrupting an operator, we advocate that shared control systems should, in addition, assist novice operators [9, 11]. Thus, our focus in this paper is to realise a shared control framework that captures key skills while ensuring the environment physical constraints are continuously satisfied.

Other critical issues that the shared control literature addresses are, for instance, assisted guidance [26], human intention prediction [15, 17], appropriate blending between human and autonomous policies [8], large time-delays [31], appropriate teleoperation spaces [22], obstacle avoidance [6, 23, 27], and representation and utilization of expert demonstrations [1]. However, these approaches tend to focus on cases where the intention of the operator consists of some distinct goal, e.g.

an object to grasp. To the best of our knowledge, very few approaches to shared control target the problem of *how* the robot moves with regards to some skill. In this work, the goal is to be able to reproduce skills, rather than simply getting from one place to another.

Some shared autonomous methods provide assistance to the operators by blending the current operator command with an autonomous policy responsible for the adaptation to the dynamic environment [18, 23]. However, these strategies lack any anticipatory reasoning, i.e. early adaptation to predicted future events. Consider the example of an unskilled operator inadvertently driving a robot towards an obstacle. In such an example, predicting the collision event and taking it in consideration by continuously re-planning a sequence of control commands over a receding horizon, in a Model Predictive Control (MPC) fashion, could significantly improve the level of assistance from the shared controller to the less skilled operator. Moreover, such capabilities are key to being able to produce skill-based controls, estimated from the operators actions, in the face of changing physical constraints.

In this work, we propose a novel receding horizon shared control method that continuously adapts future plans based on the estimation of the operators' skill intention. In order to plan motions that respect the aforementioned skills, and achieve computation times required for online teleoperation, we formulate parameterized representations for the system state and controls by making an important distinction between a skill model and an underlying trajectory – based on a special type of curve, Clothoids (see Sec. III for details). By integrating the approaches mentioned above, we achieve the following key contributions in this work:

- A novel **model-based framework for shared control** that combines skill and underlying trajectory models, enabling skill representation, estimation, and switching.
- Introduction of Clothoids as a suitable, **adaptive representation** for the underlying predicted skill trajectory.
- A novel **cost function** that improves the computational feasibility of the skill-based trajectory optimization whilst respecting the principle of minimal intervention from an operator intention perspective.
- An **MPC implementation** of the shared control method that respects a motion pattern whilst ensuring (changing) system **constraints** are satisfied.
- **Hardware realization** of our method on a KUKA-LWR robot arm, including a shared-control user study, quantitative comparisons, and several evaluations demonstrating the method capabilities.

II. Problem formulation

A key nuance, specific to shared control, is the need for incorporating a prediction of operator commands $\widehat{h}(t; h^r)$ over a future time period $t \in \mathbb{T}_f = [t_c, t_f]$, where t_c and t_f are the current and future time-stamp respectively, and raw human signals from an interface (e.g. joystick) are given by $h^r(t)$ for a window of previous commands $t \in \mathbb{T}_p = [t_p, t_c]$ such that t_p is a previous time-stamp. We assume the operator input commands to be drawn from a space equivalent to the control action space.

We can mathematically formalize a receding horizon shared controller as a standard optimal control problem. A sequence of the optimal control commands $u(t) \in \mathbb{R}^{n_u}$ and the corresponding sequence of system states $x(t) \in \mathbb{R}^{n_x}$ are the result from solving

$$x^*, u^* = \arg\min_{x,u} \text{cost}(x, u; \widehat{h}) \tag{1a}$$

$$\text{subject to } \dot{x} = f(x, u),\ x \in \mathbb{X}(\widetilde{e}),\ u \in \mathbb{U} \tag{1b}$$

where $f(\cdot)$ represents the equations of motion, $\mathbb{X}(\cdot)$ and $\mathbb{U}$ are the sets of feasible states and controls respectively represented by a combination of equality and inequality constraints, and $\widetilde{e}(t)$ is an environment model from sensing data.

Recent literature introduces two different approaches to solve (1) for obstacle avoidance [6, 27]. Both of these approaches minimize an objective function defined by $\text{cost}(x, u; \widehat{h}) = \int_{\mathbb{T}_f} \|x - \widehat{x}\|^2_{W_x} + \|u - \widehat{h}\|^2_{W_u}\, dt$ where $\widehat{x}$ is the solution of $\dot{x} = f(x, \widehat{h})$ such that $x(t_c) = x_c$ is known. Broad et al. [6] sample $N \gg 1$ controls $\{u\}_N \sim \mathbb{U}$, exclude samples such that $x \notin \mathbb{X}(\widetilde{e})$, and choose x, u with minimal cost. Rubagotti et al. [27] use an off-the-shelf solver to compute a local minimizer. Both works adhere to the principle of minimal intervention - evident by the choice of cost function - i.e. minimally adjust the operator input commands such that the system constraints are satisfied. Additionally, despite both approaches handling arbitrary prediction models $\widehat{h}$, both their experiments assume a simplistic model, i.e. $\widehat{h}(t) = h^r_c$ for all $t \in \mathbb{T}_f$ such that h^r_c is the current raw operator command.

The main focus of this paper is to address the problem of planning motions that respect a skill whilst satisfying system constraints. The cost functions and prediction models in the related literature fail to capture the key goal of maintaining a prescribed motion pattern; therefore, the need for an alternative approach that follows the principle of minimal intervention while being the least disruptive to the operators' intentions or skill choice.

A generalized treatment of human-policy identification or human intention detection is out-of-scope of this work – it is a challenging problem with many factors often leading to models that are computationally infeasible for online prediction [25]. However, several domains we identified as our target are conducive to some simplifying assumptions: we assume (i) a given finite set of skill models denoted $\mathcal{S}$, by an expert operator, that contain all motion patterns required for the completion of the task at hand, and (ii) the novice operator is skilled enough to enact teleoperation motions that, albeit sub-optimal, are identifiable by an off-the-shelf policy prediction method given $\mathcal{S}$.

Based on the formulation, we summarize the questions addressed in this paper as follows:

(Q1): What is an appropriate state and control representation that both captures the intended skills and allows the trajectories to bend and adapt in order to avoid obstacles?

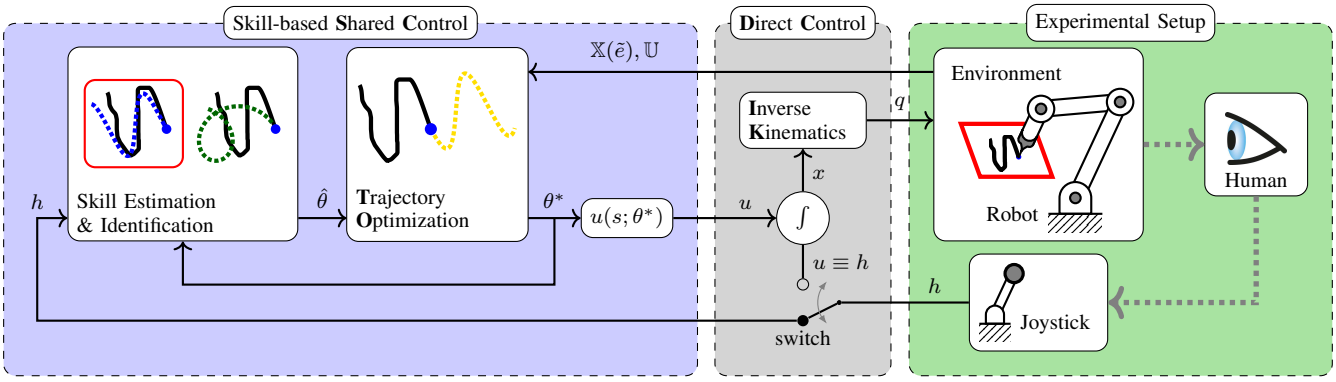

Fig. 2: Method outline. From right to left: the *Experimental Setup* illustrates an operator controlling the robot with a joystick only using visual feedback of the task; the ***Direct Control*** maps control signals to joint state targets; and the *Skill-based* ***Shared Control*** estimates skill parameters and optimizes the robot motions given continuous environment sensing.

(Q2): How to capture intentions of a human operator in the form of motion patterns?

(Q3): How to plan trajectories that respect both the estimated intended skill and the principle of minimal intervention, while avoiding obstacles and remaining computationally feasible for online teleoperation?

III. Method

In this section, we describe our proposed method for skill-based shared control. Fig. 2 illustrates and summarizes our proposed method.

A. Model representation

We start by making an observation on motion patterns that we intend to reproduce. Observe three examples of welds in Fig. 3 (upper); these show instances of patterned motions around some central axis. Fig. 3 (lower) shows two instances of patterns, a wave and cycloid, in a two-dimensional plane exhibiting the same feature (pattern around a central axis). We term this central axis (red) as the *underlying trajectory* and denote by U. Let us denote the state trajectory - the trajectory a robot will actually follow - by x (blue). In this work, we rely on a distinction between the underlying trajectory U, a skill representation S_i that produces a pattern such that subscript i denotes the ith skill from the skill set $\mathcal{S}$, and the method by which these are combined to form states x and controls u.

A key feature of the underlying trajectory is that it must sustain the intended motion pattern while having the ability to bend to avoid obstacles and satisfy changing environmental constraints. Two approaches could be considered: (1) a model-free approach where the optimization handles these requirements via constraints, or (2) a model-based approach where these requirements are inherent to the model. We have found that a model-based approach provides a suitable solution.

In order to describe our method, we introduce a change of variable – let there exist some spatial parameter s related to time by

$$\frac{ds}{dt} = v(t) \tag{2}$$

where $v(t)$ is some velocity profile. Thus, for a time period $\mathbb{T} = [t_a, t_b]$, integrating (2) such that a point, e.g. $s(t_a) = s_a$, is known leads to $s(t)$ defining $\mathbb{S} = [s_a, s_b]$. In our work, we set $v(t) = v_0$ where $v_0 \in \mathbb{R}$ is a constant parameter.

Let the underlying trajectory and skill representations be described by $U(s;\psi)$ and $S_i(s;\rho_i)$ respectively where ψ, ρ_i are model parameters. For brevity, we collect all model parameters and denote by $\theta_i = (\psi^T, \rho_i^T)^T$. We combine these models to describe the state trajectory by

$$x(s;\theta_i) = U + S_i U' \tag{3}$$

where a dash $'$ represents the derivative with respect to s. We highlight here the need for the change of variables (2): the model U' in (3) must have unit-norm so not to interfere with the skill pattern S_i. Due to a parameterized model (3), the control actions $u(\cdot)$ are given by $x'(\cdot)$, and thus defined by

$$u(s;\theta_i) = U' + S_i' U' + S_i U'' \tag{4}$$

where we have applied the product rule for differentiation.

Following assumption (i) in Sec. II, the skill set $\mathcal{S}$ is subdivided by skill models S_i, each parameterized by ρ_i, i.e. $\mathcal{S} = \cup_i \{S_i(s;\rho_i) : \rho_i \in \mathbb{R}^{m_i}\}$.

Let the skill model $S_i(s;\rho_i)$ describe some motion pattern – that is specific to a given application domain, e.g. [7] exemplifies various weave patterns in welding. Here, we state a particular form used to represent *wave* and *cycloid* skills. Let S_i be a composition of the two functions $\sigma_i(\cdot)$ and $\omega_i(\cdot)$:

$$S_i(s;\rho_i) = \sigma_i \mathcal{R}(\omega_i) \tag{5}$$

where $\mathcal{R}(\cdot)$ defines a two-dimensional rotation matrix about some angle, $\sigma_i(\cdot)$ can be thought of as a scaling and $\omega_i(\cdot)$ as a rotation angle. A skill model S_i can be therefore given by simply defining the scalar valued functions σ_i and ω_i.

During the development of our framework, various models for the underlying trajectory were considered. For example, an early considerations was a polynomial representation for U. However, this lead to numerical instabilities in the trajectory

optimization. We have found that a model-based approach using Clothoids as a representation for the underlying trajectory provides a suitable solution. The Clothoid representation – also known as Euler curves, has some inherent properties such as (1) linearly varying curvature, and (2) a compact representation (small number of parameters), that will allow us to realize online adaptation within our optimization framework. Clothoids have been utilized in road design [20], path [5] and attitude [13] planning, autonomous driving [19, 29], and continuum robotics [14].

The Clothoid model of the underlying trajectory is given by

$$U' = [\cos(\alpha), \sin(\alpha)]^T, \text{ such that} \tag{6a}$$
$$\alpha' = \phi_0 + \phi_1(s - s_c) \tag{6b}$$

for all $s \in \mathbb{S}_f = [s_c, s_f]$ where $\alpha(s)$ describes the orientation and $\phi = [\phi_0, \phi_1]^T \in \mathbb{R}^2$ describe the curvature of the Clothoid trajectory. Note, $\alpha(s)$ is found by integrating (6b) such that $\alpha(s_c) = \alpha_c$ is known. The underlying trajectory parameters are $\psi = [\phi_0, \phi_1, \alpha_c]^T$. Notice, for all α, U' has unit-norm.

We can find U by integrating (6a) given that $U(s_c; \psi) = U_c$ is known. If the current robot state x_c is known, then we can obtain $U_c = x_c - S_i U'$ by re-arranging (3). Note, the analytic integral of (6a) can only be found in terms of a special class of functions known as the Fresnel integrals [16]. To avoid this difficulty, we instead compute U by approximation using the multi-variable version of the Runge–Kutta 4 method. Additionally, by simply defining more curvature parameters ϕ_0, ϕ_1, multiple Clothoid segments can be concatenated to form more elaborate underlying trajectories.

B. Skill estimation

As in related literature [8, 15, 17], we make the assumption that we are able to infer operator intent conditioned on a window of their previous input – note, intent may change. In our case, the intent estimation consists of a selection of parameters $\widehat{\theta}_i$, providing a description of the operators intent within the context of skills.

We describe how to estimate $\widehat{\theta}_i$ from a window of raw interface signals $h^r(s)$ for $s \in \mathbb{S}_p = [s_p, s_c]$ that correspond to the time window $\mathbb{T}_p = [t_p, t_c]$ where t_c is the current time, and t_p is some previous time stamp such that $t_w = t_c - t_p$ is a specified window duration. Note, since our method relies on this window duration for t_w seconds at the start of a task we must control the robot in direct control - see the switch in Fig. 2. We treat θ_i as decision variables. Under assumption (ii) in Sec. II, we infer $\widehat{\theta}_i$ by solving

$$\widehat{\theta}_i = \arg\min_{\theta_i} \varepsilon_i(\theta_i) \quad \text{subject to} \quad \theta_i \in \Theta_i^{SE} \tag{7}$$

such that $\varepsilon_i(\cdot)$ is an error function defined by

$$\varepsilon_i(\theta_i) = \int_{\mathbb{S}_p} \|u(s; \theta_i) - h^r(s)\|^2 \, ds + R(\theta_i) \tag{8}$$

where $u(\cdot)$ is defined in (4), $R(\cdot)$ is a regularization term, and Θ_i^{SE} is the set of possible values for θ_i represented by a combination of inequality and equality constraints.

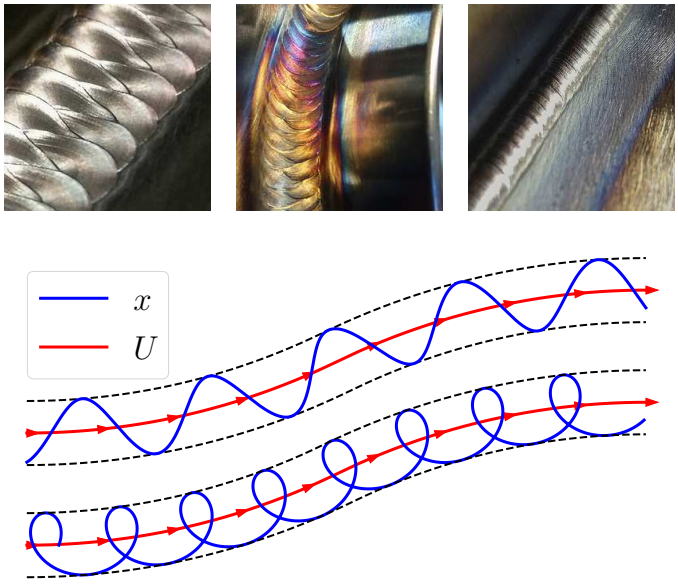

Fig. 3: Motion pattern examples. Upper: weave patterns used in welding, credit Josh Welton [30]. These highlight patterns implemented in industry to achieve different goals. Lower: model-based trajectories for a wave and cycloid skill.

Regarding the regularization term $R(\cdot)$, we can use it to bias the solution towards pre-specified goals. As an example, we may wish to reduce the rate of change in the skill parameters ρ_i, sustaining characteristics of the skill. For instance, in the case of highly variable inputs, we can define the regularization term as $R(\theta_i) = \|\rho_i - \widehat{\rho}_i^{prev}\|^2_{W_R}$ where $\widehat{\rho}_i^{prev}$ is the previous skill parameter solution and W_R is some appropriate weighting matrix. Alternatively, setting $R(\theta_i) = \|\phi\|^2_{W_R}$ penalizes ϕ, favoring low curvature trajectories. Additionally, a combination of weighted terms can address multiple goals.

C. Skill identification

Recall from Sec. III-A our skill set $\mathcal{S}$ is sub-divided by a number of skill models S_i. We identify the model S_i and corresponding values for $\widehat{\theta}_i$ by solving (7) for each model and chose the skill that results in the smallest error ε_i.

D. Trajectory optimization

We have established how to identify a skill model S_i and a corresponding parameter description $\widehat{\theta}_i$ for the intended state and control signals from a window of the operators raw interface signals. Ideally, in the next control loop step $s_n = s(t_n)$ (i.e. $t_n = t_c + \delta t$ where δt is the control loop cycle duration), we would like to execute the trajectory that most closely resembles the operator's intent, i.e. $x(s_n; \widehat{\theta}_i), u(s_n; \widehat{\theta}_i)$. However, our shared control system must also be able to assist a novice operator in avoiding collision with obstacles while simultaneously maintaining the intended skill. Since our state and control trajectories are functions of the parameters θ_i, finding optimal parameters θ_i^* by solving (1) directly leads

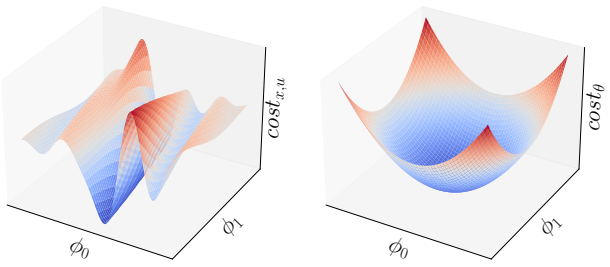

Fig. 4: Cost landscape comparison highlighting the convexity in our objective function (right) as opposed to the typical principle of minimal cost function over applying our state and trajectory models (left). Note, $\text{cost}_{x,u} = \text{cost}\big(x(\theta_i), u(\theta_i)\big)$ where cost is (1) substituting our models for x, u given by (3) and (4) respectively, and $\text{cost}_\theta = \text{cost}(\theta_i; \widehat{\theta}_i)$ is our cost function (10). Note, in both cases all variables in θ_i are fixed apart from ϕ_0, ϕ_1 for illustration purposes.

to a highly nonlinear scheme. During the development of our method, we found that constrained optimization solvers often fail to converge in this case. We resolve this issue by adapting (1) and using a modified version of the cost function, leading to the following optimization problem

$$\theta_i^* = \arg\min_{\theta_i} \text{cost}(\theta_i; \widehat{\theta}_i) \tag{9a}$$

subject to

$$x(s;\theta_i) \in \mathbb{X}(\widetilde{e}),\ u(s;\theta_i) \in \mathbb{U}, \theta_i \in \Theta_i^{TO} \tag{9b}$$

for all $s \in \mathbb{S}_f$

such that

$$\text{cost}(\theta_i; \widehat{\theta}_i) = \|\theta_i - \widehat{\theta}_i\|^2_{W_{TO}} \tag{10}$$

where $x(\cdot)$ and $u(\cdot)$ are defined by (3) and (4) respectively, Θ_i^{TO} is the set of possible values for θ_i, and $\mathbb{S}_f = [s_c, s_f]$ corresponds the time horizon $\mathbb{T}_f$ as in (1). Note that we make a differentiation between Θ_i^{SE} and Θ_i^{TO}. Depending on the application goals you can restrict the range of possible values that θ_i can take in both (7) and (9): for example, we may wish to set the underlying trajectory initial orientation α_c to the value estimated in (7) – this is enforced in (9) by setting the constraint $\alpha_c = \widehat{\alpha}_c$ where $\widehat{\alpha}_c$ is the value found by solving (7). Notice also that since we have derived explicit equations for x, u in terms of parameters θ_i, we are able to remove the equations of motion[1] as an equality constraint.

The cost function proposed here follows the criteria of the principle of minimal intervention; however, our cost function attempts to maintain the operators intentions as described by the estimated parameters. Additionally, our cost function design has the numerical benefit of being strictly convex with respect to parameters θ_i, as schematically illustrated in Fig. 4. However, note the constraints are non-convex, meaning (9) is still a nonlinear optimization problem - only convex in its objective function (10).

[1]Recall the equations of motion are represented by $f(\cdot)$ in (1b)

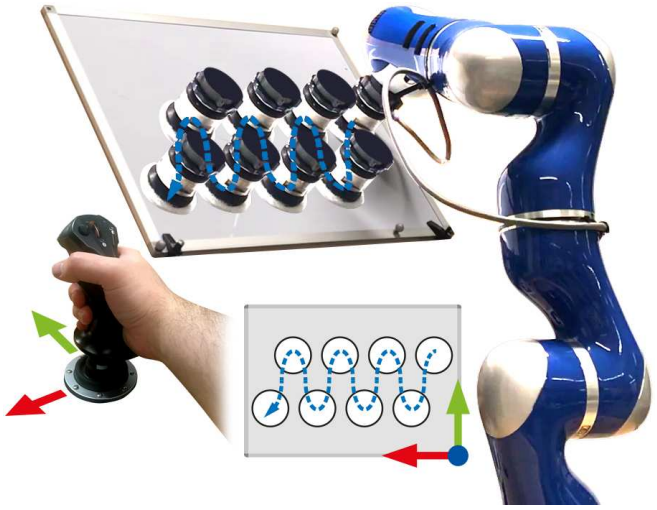

Fig. 5: Wiping mock-up task where a user teleoperates a robot to perform a wave motion pattern from *right to left*.

IV. Experiments

We now describe an experimental setup for skill-based shared control, specifically for a wiping task involving two different skills defined explicitly. Note the conceptual similarity between wiping, welding, and spraying, where the two-dimensional task can directly map to a three-dimensional workspace.

The goal of the experimental mock-up task is for a robot to sweep a given area of a surface, e.g. see Fig. 5. We assume the operator enacts one of two skills: (i) a wave skill where the robot should travel in a wave back-and-forth across the underlying trajectory, and (ii) a cycloid skill where the robot should travel in circles around the underlying trajectory. Let the states $x \in \mathbb{R}^2$ define the position of the end-effector on the wiping surface and the control input $u \in \mathbb{R}^2$ as the 2D input planar velocity. The operator gives two-dimensional inputs $h^r \in \mathbb{R}^2$ via a joystick.

Equation (5) defines a particular version of the skill representation in terms of the scaling and rotational functions $\sigma_i(\cdot)$ and $\omega_i(\cdot)$ respectively. Table I defines the wave, and cycloid skills explicitly in terms of these two functions.

TABLE I: Skill representations.

S_i	Skill	m_i	$\sigma_i(s;\rho_i)$	$\omega_i(s;\rho_i)$
S_1	Wave	3	$\rho_1^{(0)} \sin\left(2\pi\rho_1^{(1)}\left(s+\rho_1^{(2)}\right)\right)$	$\pi/2$
S_2	Cycloid	3	$\rho_2^{(0)}$	$2\pi\rho_2^{(1)}\left(s+\rho_2^{(2)}\right)$

For the skill estimation stage, the regularization function is defined by $R(\theta_i) = \|\theta_i - \widehat{\theta}_i^{prev}\|^2_{W_R}$ where $\widehat{\theta}_i^{prev}$ is the previous solution, and the constraints that define Θ_i^{SE} are as follows: $\theta_i^{min} \leq \theta_i \leq \theta_i^{max}$ bound the values of θ_i within lower θ_i^{min} and upper θ_i^{max} limits, and $\alpha_c = \alpha^*_{prev}$ where

α^*_{prev} is the previous solution from the trajectory optimization retrieved via a feedback loop as in Fig. 2. Note that on the very first iteration of our method that $\widehat{\theta}_i^{prev}$ and α^*_{prev} is undefined, and so the regularization term R and this particular constraint is relaxed for a single iteration.

In both the skill estimation and trajectory optimization, the solvers are seeded with the previous solution. In the very first iteration, a guess is given to the skill-estimation whereas the trajectory optimization uses the solution from the skill estimation $\widehat{\theta}_i$ as the initial seed.

A. System description

We implemented our method within the CasADi framework [2], and leverage the plugin for SNOPT [12] to solve (7) and (9). Our method runs in an MPC loop: given a window of previous operator inputs, we estimate $\widehat{\theta}_i$ for all skills i in separate processing threads, identify the intended skill by comparing the cost of fittings (8), compute an optimal θ_i^* by solving (9) – resulting in a trajectory of states $x(\theta_i^*)$ and controls $u(\theta_i^*)$, from which we execute the first step.

The pose of the whiteboard, used as the wiping surface for the robot, is tracked using a Vicon motion capture system. Participants interface with the system using a Thrustmaster T-Flight HOTAS X joystick. The two main axes of the joystick are mapped corresponding to the two dimensions of the whiteboard, see Fig. 5. Our experiments were executed on a PC running 64-bit Ubuntu 20.04 with a 16-core Intel Core i9-9900KF CPU at 3.60GHz. Data was collected using a 7-DoF KUKA LWR Arm.

B. Switching intended skill

We start by demonstrating our methods capability to identify a skill and a change in the operators' intention solely from streaming joystick data. An operator starts by performing a wave skill, and then switches to a cycloid roughly 20 seconds after the method starts. Fig. 6 (right) shows the trajectory resulting from applying control commands using our method. Fig. 6 (upper-left) shows the velocity profile of the commanded versus applied control as well as the instantaneous deviation between the two. Fig. 6 (lower-left) demonstrates the evolution of the skill fitting costs and instance of the skill switch identification.

C. Static obstacle avoidance and solve duration comparison

Next we demonstrate our method's ability to continuously adhere to environmental constraints while maintaining the features of the estimated skill. We now introduce a boundary, represented in (9b) as an inequality constraint given by $x_{min} \leq x(s;\theta_i) \leq x_{max}$ for all $s \in \mathbb{S}_f$. Fig. 7 (upper) shows the trajectory taken by the robot and its corresponding underlying trajectory. It also shows the trajectory that results from applying the same operator commands in a direct control mode, leading to a collision – highlighted by the black dotted lines.

An important consideration of any teleoperation system is that the overall computational time should remain below a certain threshold to allow for reasonable sampling frequencies. Fig. 7 (lower) shows the skill estimation and trajectory optimization solver duration for each MPC loop cycle for this obstacle avoidance experiment. The average number of iterations was 16.9 ± 5.1 and 2.4 ± 3.3 and the average CPU time was $4.9 \pm 1.1\text{ms}$ and $1.4 \pm 0.2\text{ms}$ for the skill estimation and trajectory optimization, respectively. The average total CPU time was $6.3 \pm 1.2\text{ms}$, well under the 20ms threshold for a 50Hz operation.

D. Obstacle avoidance in a dynamic environment

A key requirement for the adoption of shared control systems in industry, such as the construction sector, is the ability to modify control inputs in the face of dynamic (changing) environmental constraints. Due to the MPC-nature of our method we are able to adapt online to changes in the environment whilst maintaining the skill features.

We tracked a straight edge using a Vicon motion capture system (see Fig. 8) and incorporated this in our trajectory optimization step by including $d(x,\nu) \geq 0$ for all $s \in \mathbb{S}_f$ as a dynamic constraint in (9b). Here, $d(\cdot)$ is a signed distance function, which computes the distance to the line for feasible states (negative otherwise) while ν is the pose of a Vicon tracking marker – updated at each control loop cycle.

An experimenter moved the straight edge within the workspace of the robot using a motion unknown to the operator. Fig. 8 demonstrates that our method was able to avoid the obstacle whilst maintaining the required skill.

E. User Study

In this section, we describe our user study. The goal of this experiment is to analyze the relation between the quality of the inputs, provided by the participants, and the quality of the resulting motion patterns for our method in comparison to direct control.

We conducted a within-subjects experiment, evaluating the performance of 11 participants (9 male, 2 female). In our previous work we observed that certain habits (i.e. playing computer games) can effect task performance. Using the same criteria as in [22], we classified four participants as highly familiar with computer games.

1) Participant protocol: Participants were tasked with controlling the robot to perform a wiping task straight along the whiteboard, right to left, for two different skills (wave and cycloid) and in two different modes: direct control (DC) and shared control (SC). After consent was taken, the participant was given a practice run to familiarize themselves with the joystick.

We presented the skills to the user in a fixed order: wave then cycloid. The order in which we presented the modes DC and SC were randomized. We informed the participants they would teleoperate under an assistive (i.e. shared control) and non-assistive (i.e. direct control) modes of control without specifying the sequence. At the start of each skill block, a demonstration of the skill was played on the robot so they could easily envisage the task they were required to

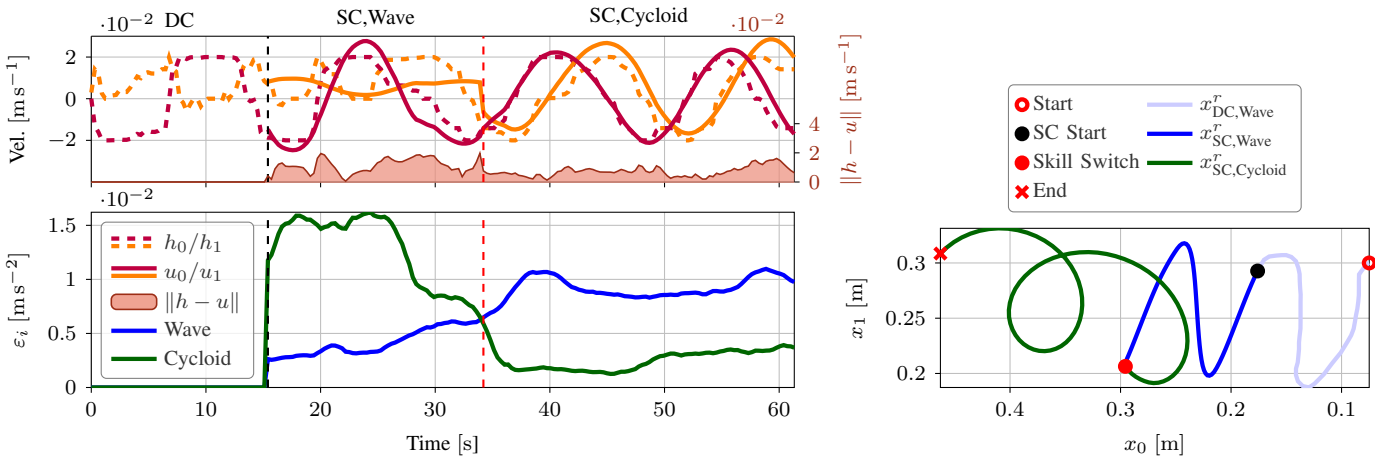

Fig. 6: Skill switch experiment. Upper-left: input commands in the velocity space, the resulting velocity controls, and the difference between the two signals. Lower-left: corresponding fitting error $\varepsilon_i(\cdot)$ for both skills indicating (by the vertical dashed red line) the moment the skill switches. Right: robot trajectory (note that the robot moves from *right to left*).

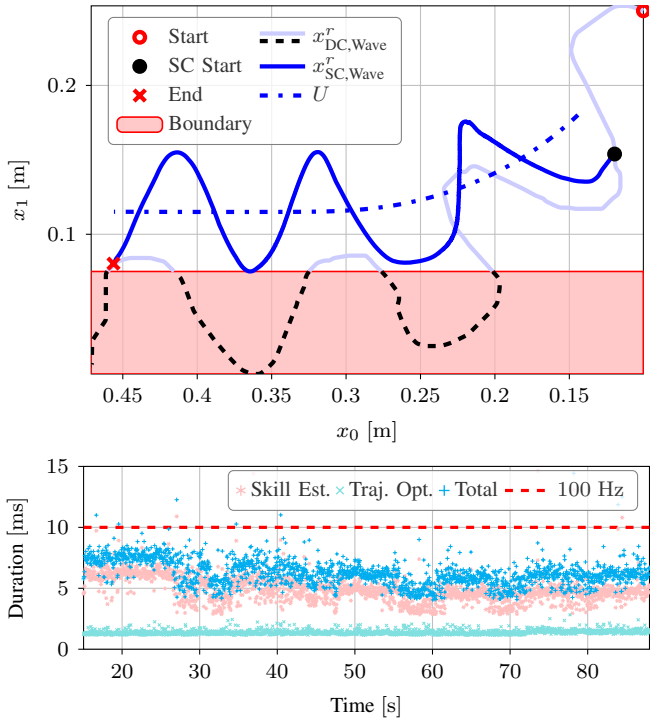

Fig. 7: Static obstacle avoidance experiment. Upper: shows the trajectory produced by direct control where the operator inadvertently leads the robot into constraint violation and also the trajectory produced by shared control - the robot moves from *right to left*. Lower: reports the computation times for the skill estimation, the trajectory optimization, and the total time.

perform. For each of the four trials, the user was asked to perform one unrecorded practice run, then another recorded trial. Participants were able to visually observe the current state of the robot without being given any visual feedback about the robots trail across the board, and thus were unable to monitor the resulting trajectory.

2) Measures and analysis: During each trial, we collected joystick signals h^r, and target positions x^r (in the whiteboard coordinate frame) at a sampling frequency of 50Hz. Note that superscript r refers to the fact that these values are raw signals. Recall from Sec. III-B, our method relies on a direct control initialization that lasts for a certain window duration t_w before shared control is activated. The data corresponding to this initialization was removed for both the direct control and shared control results so that the comparison between the two modes was fair.

We measure the quality of the joystick and applied control signals by fitting an ideal signal using our model (4), i.e. for each trial we compute

$$F_h = \frac{1}{T} \min_{\theta_i} \sum_{k=0}^{N} \left\| u\big(s(t_k);\theta_i\big) - h^r_k \right\|^2 \quad \text{subject to } \theta_i \in \Theta_i \tag{11}$$

where T is the duration of the trial used here to normalize the fitting, t_k are time stamps, N is the number of data points collected. Similarly, we measure the quality of the target position signals x^r by computing

$$F_x = \frac{1}{T} \min_{\theta_i} \sum_{k=0}^{N} \left\| x\big(s(t_k);\theta_i\big) - x^r_k \right\|^2 \quad \text{subject to } \theta_i \in \Theta_i \tag{12}$$

where $x(\cdot)$ is our model defined by (3). In both (11) and (12) we set the parameter $s_c = 0$ and constrain the variables for the Clothoid curvature ϕ to zero, since the task was to perform a skill around an underlying trajectory with no curvature. We leave all other parameters in θ_i unconstrained.

For both F_h and F_x, smaller values indicate more accurate fitting and thus higher performance. Note that even though our shared control method produces trajectories based on the models we fit here, we do *not* expect zero fitting error for our

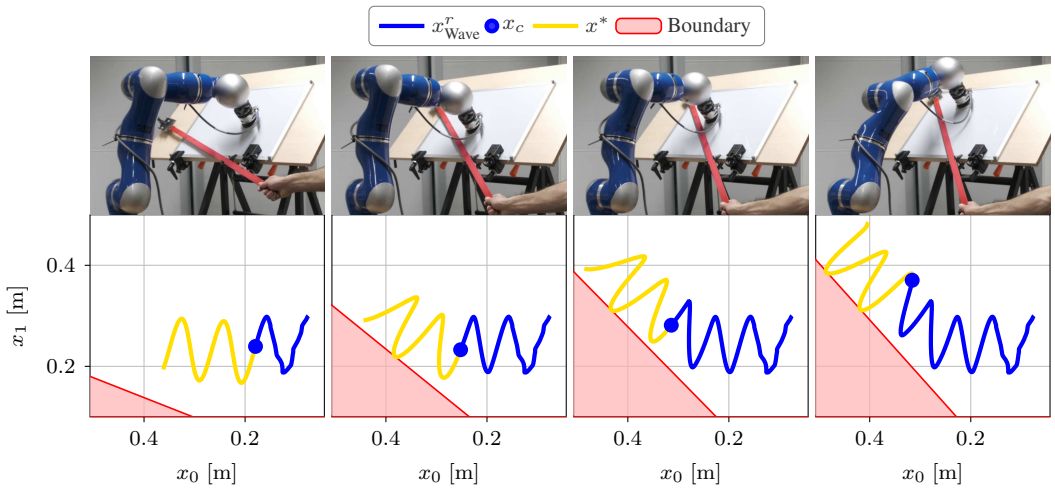

Fig. 8: Dynamic obstacle avoidance experiments at four snapshots in time. The obstacle (red boundary) is sensed using the Vicon Motion Capture system, and represented as parameterized constraints in (9). Note, the robot moves from *right to left*.

method since we allow the method to adapt to the input from the human (i.e. $\widehat{\theta}_i$ is modified online).

An example from two participants for each skill are shown in Fig. 9. We observe that, visually, the quality of the direct control signals is far poorer than the shared control.

The results of the computed fittings are shown in Fig. 9. For the wave skill, we observe that shared control leads to similar average performance for F_x, whereas the results for the cycloid skill clearly lead to a reduction in the fitting error. For the shared control mode, both the mean and variance of F_h are higher than that of the direct control signals, i.e., the operators joystick signals become more varied in the shared control mode as opposed to the direct control. This *might* be an indication that our shared control method leads to a reduction in the participants cognitive load - however, we acknowledge that further study would be required to prove or disprove this hypothesis. Despite more varied joystick signals, our method is able to produce more accurate signals for the cycloid skill. However, there is little change when comparing the average fitting F_x for the wave. This could be an indication that our method has more corrective interventions in scenarios with more complex skills.

V. Discussion

Next we discuss a few interesting observations in depth and sketch future directions. In the user study (Sec. IV-E), it was reported verbally by some participants that the speed of the robot changed at certain points of the pattern. Setting the speed of the underlying trajectory in (2) as a constant, i.e. $v(t) = v_0$, enables a spatial-temporal coupling between the underlying trajectory and state trajectory that leads to increases and decreases in speed at certain stages of the pattern (e.g. for the wave, at peaks/troughs the speed through the pattern is slow, whereas the speed at the mid points is faster). Whilst this parameter adapts to the users commands, the change in speed through the state trajectory is still perceivable and we hypothesize this may adversely affect user experience. To counter this effect, future work will explore richer models for $v(t)$ that removes this coupling.

Our skill switching experiment (Sec. IV-B) demonstrates the ability of the shared controller to react to an operator's change of intention. Fig. 6 (right) highlights this change with a red dot. While conducting the experiment, we observed that there is a slight delay from when the user starts to produce a cycloid to when the controller identifies this switch. The question of whether this leads to user dissatisfaction remains open. We could potentially minimize this misalignment by accelerating the skill-identification process. However, our focus in this work was to introduce a novel conceptual framework for skill-based shared control, and thus we relegate this analysis and extension to future work.

Our method, in its current formulation, is restricted to a certain window duration t_w that must be enforced for all skill models. This could potentially lead to difficulties when tuning parameters, especially in the face of, for example, a line skill (defined by $\sigma_i = 0$, $\omega_i = 0$) - we may see a bias in the skill identification step towards a particular skill as opposed to any other without an adaptive window size. We plan to investigate the effect of additional skills on the skill estimation and identification steps.

Fig. 7 (upper) shows how our skill-based shared control method adapts a trajectory to avoid a boundary. The trajectory curves smoothly whilst ensuring the wave skill. Visually, the wavelength of the resulting trajectory seems slightly smaller. This might be due to the regularization term of the skill identification, added in order to avoid over-fitting the operator's commands which have a fairly high variance (Fig. 6).

Fig. 7 (lower) shows the solver duration's for the skill identification and trajectory optimization steps. For the majority of cases ($\sim$99%), the total duration stays under 10ms, compatible for online teleoperation. This particular experiment considers

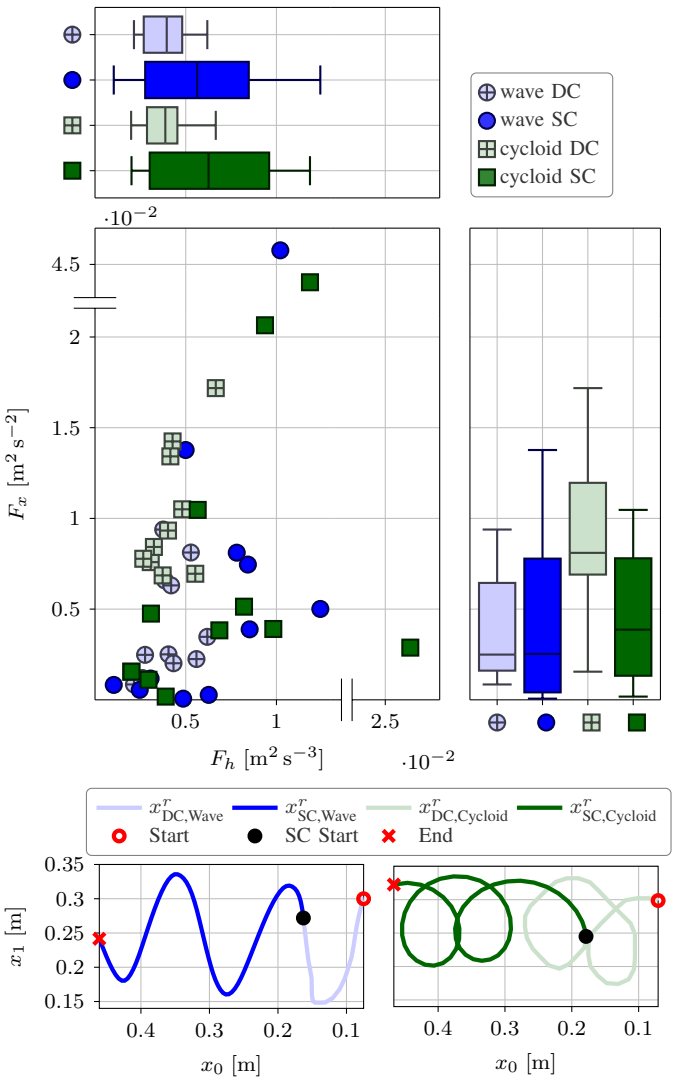

Fig. 9: User study experiment. Upper: scatter plot comparing fitting error in the input commands and resulting trajectories with the corresponding box plots. Note, the box plots share the same axes as the scatter plot. Lower: participant shared control examples where the robot moves from *right to left*.

only a single skill. Considering more skills will increase the total duration of the skill identification, if performed sequentially. However, the skill identification step for multiple skills are mutually exclusive, and so we can avoid this issue by exploiting multi-threading or GPU hardware to perform these computations in parallel.

VI. Conclusions

In this paper we have introduced a novel framework for skill-based shared control. We demonstrate that our system is able to identify different skills, react to changes in operator intentions, and re-plan whilst accounting for changing environmental constraints. Our framework follows from a novel representation for state and control that exploits a well-established geometrical primitive, i.e. Clothoids, and a parametric model of skills. We show that our shared control method is computationally efficient for online teleoperation within a Model Predictive Control framework, that enables its adaptability to dynamically changing constraints. Furthermore, we validated our method in a lab mock-up on a KUKA LWR and showed its capability to identify skill switches. Our user study provides some early evidence that this approach to shared control leads to improved quality of output motions, while potentially reducing the operators cognitive load. In future work, we intend to investigate (i) the incorporation of learned skill representations based on expert demonstrations, rather than pre-specified analytical models; (ii) the modelling of the velocity profile of the state trajectory, as we observed that such component might significantly impact the user experience; and (iii) the use of probabilistic approaches for improving the inference of the operator's intended skill.

References

[1] Firas Abi-Farraj, Takayuki Osa, Nicoló Pedemonte Jan Peters, Gerhard Neumann, and Paolo Robuffo Giordano. A learning-based shared control architecture for interactive task execution. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017. doi:10.1109/ICRA.2017.7989042.

[2] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi:10.1007/s12532-018-0139-4.

[3] Michael Ballou. Shotcrete rebound – how much is enough? *Shotcrete magazine, American Shotcrete Association*, 2003.

[4] Joshua D. Bard, David Blackwood, Nidhi Sekhar, and Brian Smith. Decorative robotic plastering - a case study of real-time human machine - collaboration in high-skill domains. *Proceedings of the 33rd eCAADe Conference - Volume 2, Vienna University of Technology, Vienna, Austria*, 2015.

[5] Mišel Brezak and Ivan Petrović. Real-time approximation of clothoids with bounded error for path planning applications. *IEEE Transactions on Robotics*, 30(2):507–515, 2014. doi:10.1109/TRO.2013.2283928.

[6] Alexander Broad, Todd Murphey, and Brenna Argall. Highly parallelized data-driven mpc for minimal intervention shared control. In *Proceedings of Robotics: Science and Systems (RSS)*, Freiburg im Breisgau, Germany, 2019. doi:10.15607/RSS.2019.XV.008.

[7] Heitor Abdias da Silva Pereira, Marcelo Cavalcanti Rodrigues, and João Vitor Lira de Carvalho Firmino. Implementation of weave patterns by path parameterization in the simulation of welding processes by the finite element method. *The International Journal of Advanced Manufacturing Technology*, 104(1):477–487, 2019. doi:10.1007/s00170-019-03861-5.

[8] Anca D. Dragan and Siddhartha S. Srinivasa. A policy-blending formalism for shared control. *The International Journal of Robotics Research (IJRR)*, 32(7):790–805, 2013. doi:10.1177/0278364913490324.

[9] Nima Enayati, Giancarlo Ferrigno, and Elena De Momi. Skill-based human–robot cooperation in tele-operated path tracking. *Autonomous Robots*, 42(5):997–1009, 2018. doi:10.1007/s10514-017-9675-4.

[10] Mustafa Suphi Erden. Manual welding with robotic assistance compared to conventional manual welding. In *IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 570–573, 2018. doi:10.1109/COASE.2018.8560489.

[11] Mustafa Suphi Erden and Aude Billard. Hand impedance measurements during interactive manual welding with a robot. *IEEE Transactions on Robotics (T-RO)*, 31(1): 168–179, 2015. doi:10.1109/TRO.2014.2385212.

[12] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review. A Publication of the Society for Industrial and Applied Mathematics*, 47:99–131, 2005. doi:10.1137/S1052623499350013.

[13] Vicent Girbés, Gloria Vanegas, and Leopoldo Armesto. Clothoid-based three-dimensional curve for attitude planning. *Journal of Guidance, Control, and Dynamics*, 42 (8):1886–1898, 2019. doi:10.2514/1.G003551.

[14] Phanideep S. Gonthina, Apoorva D. Kapadia, Isuru S. Godage, and Ian. D. Walker. Modeling variable curvature parallel continuum robots using euler curves. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEE, 2019. doi:10.1109/ICRA.2019.8794238.

[15] Kris Hauser. Recognition, prediction, and planning for assisted teleoperation of freeform tasks. *Autonomous Robots*, 35(4):241–254, 2013. doi:10.1007/s10514-013-9350-3.

[16] Mark A. Heald. Rational approximations for the fresnel integrals. *Mathematics of Computation*, 44(170):459–461, 1985. doi:10.1090/S0025-5718-1985-0777277-6.

[17] 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 (IJRR)*, 37(7):717–742, 2018. doi:10.1177/0278364918776060.

[18] Sina Nia Kosari, Fredrik Rydén, Thomas S. Lendvay, Blake Hannaford, and Howard Jay Chizeck. Forbidden region virtual fixtures from streaming point clouds. *Advanced Robotics*, 28(22):1507–1518, 2014. doi:10.1080/01691864.2014.962613.

[19] Pedro F. Lima, Marco Trincavelli, Jonas Martensson, and Bo Wahlberg. Clothoid-based model predictive control for autonomous driving. In *European Control Conference (ECC)*. IEEE, 2015. doi:10.1109/ecc.2015.7330991.

[20] Hormoz Marzbani, Reza N. Jazar, and M. Fard. Better road design using clothoids. In Ingemar Denbratt, Aleksandar Subic, and Jörg Wellnitz, editors, *Sustainable Automotive Technologies 2014*, pages 25–40. Springer International Publishing, 2015. doi:10.1007/978-3-319-17999-5_3.

[21] João Moura, William Mccoll, Gerard Taykaldiranian, Tetsuo Tomiyama, and Mustafa Suphi Erden. Automation of train cab front cleaning with a robot manipulator. *IEEE Robotics and Automation Letters (RA-L)*, 3(4): 3058–3065, 2018. doi:10.1109/LRA.2018.2849591.

[22] Christopher E. Mower, Wolfgang Merkt, Aled Davies, and Sethu Vijayakumar. Comparing alternate modes of teleoperation for constrained tasks. In *IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1497–1504, 2019. doi:10.1109/COASE.2019.8843265.

[23] Christopher E. Mower, João Moura, Aled Davies, and Sethu Vijayakumar. Modulating human input for shared autonomy in dynamic environments. In *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019. doi:10.1109/RO-MAN46459.2019.8956304.

[24] Günter Niemeyer, Carsten Preusche, and Gerd Hirzinger. *Telerobotics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-30301-5_32.

[25] Ozgur S. Oguz, Zhehua Zhou, and Dirk Wollherr. A hybrid framework for understanding and predicting human reaching motions. *Frontiers in Robotics and AI*, 5:27, 2018. doi:10.3389/frobt.2018.00027.

[26] Louis B. Rosenberg. Virtual fixtures: Perceptual tools for telerobotic manipulation. In *IEEE Virtual Reality Annual International Symposium*, pages 76–82, Washington, DC, USA, 1993. IEEE Computer Society. doi:10.1109/VRAIS.1993.380795.

[27] Matteo Rubagotti, Tasbolat Taunyazov, Bukeikhan Omarali, and Almas Shintemirov. Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control. *IEEE Robotics and Automation Letters (RA-L)*, 4(3):2746–2753, July 2019. doi:10.1109/LRA.2019.2917707.

[28] Wilko Schwarting, Javier Alonso-Mora, Liam Pauli, Sertac Karaman, and Daniela Rus. Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017. doi:10.1109/icra.2017.7989224.

[29] Júnior A. R. Silva and Valdir Grass. Clothoid-based global path planning for autonomous vehicles in urban scenarios. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEE, 2018. doi:10.1109/ICRA.2018.8461201.

[30] Josh Welton. How do you get those weaved welds? thefabricator.com/thewelder, 2014.

[31] Zhang Ya-kun, Li Hai-yang, Huang Rui-xue, and Liu Jiang-hui. Shared control on lunar spacecraft teleoperation rendezvous operations with large time delay. *Acta Astronautica*, 137:312 – 319, 2017. doi:10.1016/j.actaastro.2017.04.014.