Feasibility Checking and Constraint Refinement for Shared Control in Assistive Robotics

Samuel Bustamante, Ismael Rodríguez, Gabriel Quere, Peter Lehner, Maged Iskandar, Daniel Leidner, Andreas Dömel, Alin Albu-Schäffer, Jörn Vogel, Freek Stulp

***Abstract*—Shared control enables users with motor impairments to control high-dimensional assistive robots with low-dimensional user interfaces. The challenge is to simultaneously 1) provide support for completing daily living tasks 2) enable sufficient freedom of movement to foster user empowerment 3) ensure that shared-control support precludes the robot from running into kinematic limitations such as obstacles, unreachable areas, or loss of manipulability due to joint limits.**

In this letter, we propose a framework that performs feasibility checks before executing a shared-control task. We activate shared control only if a task is deemed feasible, and refine the task regions by excluding paths that are infeasible, e.g. due to obstacles or kinematic limitations. This reduces task failures, whilst still ensuring freedom of movement. We evaluate our framework on a set of daily living tasks with our wheelchair-based mobile manipulator EDAN.

***Index Terms*—Telerobotics and Teleoperation; Physically Assistive Devices.**

I. Introduction

USING a low-dimensional input device to control an assistive robot with many degrees of freedom – for instance our wheelchair-based mobile manipulator EDAN [1] – can be challenging, at times even frustrating. Shared control facilitates the use of such robots for activities of daily living.

As users with motor impairments express satisfaction when being empowered to control the robot [2], [3] – rather than the robot executing the task autonomously – they should have as much freedom of movement as possible. For instance, shared control should keep a bottle tip over a glass while pouring to avoid spilling. But it should not predetermine the approach direction or the amount of water that is poured by following a predetermined path.

Manuscript received: October 23, 2023; Last revised May 6, 2024; Accepted June 19, 2024. This paper was recommended for publication by Editor Jee-Hwan Ryu upon evaluation of the Associate Editor and Reviewers' comments.

All authors are with the German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchner Str. 20, 82234 Weßling, Germany. Authors SB & AAS are also with the Technical University of Munich (TUM), TUM School of Computation, Information and Technology, Garching, Germany. Contact: `first.last@dlr.de`.

This work was partly supported by the German Research Foundation (DFG) within the Collaborative Research Center EASE (SFB 1320) and the Bavarian Ministry of Economic Affairs, Regional Development and Energy (StMWi) by means of the project SMiLE2gether (LABAY102).

The authors thank João Silverio and Arne Sachtler for their help with formalization; Antonin Raffin for code reviews; Annette Hagengruber and Milena Eisemann for continual support and motivating discussions; and the paper reviewers for their valuable feedback.

Digital Object Identifier (DOI): 10.1109/LRA.2024.3430710

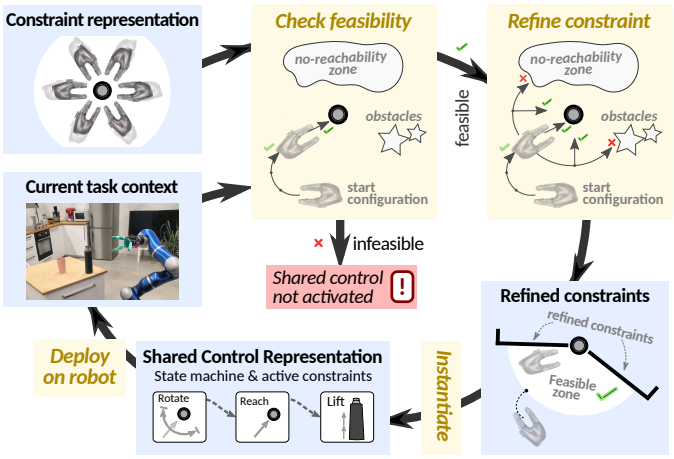


Fig. 1: Summary of our approach. Shared control is only activated after its feasibility with respect to obstacles and kinematic constraints has been confirmed. If it is feasible, TSR constraints are refined, and shared control is instantiated with the refined TSR constraints.

Shared control should, on the other hand, assist the user by excluding motions that would lead to task failure, including collisions and kinematic limitations. To do so, previous works have proposed methods that apply constraints locally *during* the movement [4], [5], [6], [7]. In this article, we propose a framework that performs feasibility checks that detect kinematic limitations *before* shared control for the task is activated. If a task is deemed infeasible – i.e. not one path was found which achieves the task – shared control is not activated for the task, as illustrated in Fig. 1. The user is thus averted from having to ascertain the task's infeasibility during its execution.

In principle, motion planning could compute one feasible path that solves the task, and guide the user along this predetermined path [8]. However, this would not satisfy the freedom-of-movement requirement for user empowerment. We therefore use *Task Space Regions* (TSRs) [9] to represent all feasible paths. In our approach, TSRs are refined such that obstacle collisions and unreachable zones are excluded from the region. The resulting TSR is then used to instantiate a Shared Control Template [10], which aims at providing freedom of movement whilst still respecting kinematic limitations and safety considerations.

The main contribution of this paper is to propose a shared-control framework that: 1) checks the feasibility of conducting

a task with shared control *before* it is executed; 2) does not activate shared control if the task is deemed infeasible (the user is thus kept from starting the execution of tasks that are predicted to fail eventually); 3) refines the shared-control constraints for feasible tasks, if necessary in the current task context (this further supports the user in avoiding kinematic limitations *during* the assisted task).

This paper is structured as follows: After presenting related work and SCTs in the next two sections, we describe how our shared-control framework has been extended with TSRs in Section IV. We describe the procedures for feasibility checks and constraint refinement in Section V. After presenting and discussing the experimental evaluation in Section VI, we conclude with Section VII.

II. Related Work

In the context of teleoperation, the term *kinematic limitations* has been used to describe a variety of movement limitations that arise during teleoperation due to obstacles, virtual walls, joint limits, singularities, or reduced manipulability [11].

Our approach aims at checking the kinematic feasibility of a task *before* it is executed (which is common in autonomous robots) rather than *during* the task (which is common in shared control). These two approaches are complementary, and determine the structure of this section.

a) Avoiding kinematic limitations during shared control: A common and effective method to signal proximity to kinematic limitations during shared control is haptic cues from force feedback [4], [5], [6]. As this modality is not available on our assistive system – it is controlled through electromyography [1] or with a joystick so as to tailor it to the needs of the target group – we do not include this modality in the work described in this paper.

An alternative is to provide the user with constraints, either with a static environment (e.g. created using perception [12]) or with reactive approaches during task execution [4], [5], [6], [7]. As an example within our application domain, Iregui et al. [7] prevent the robot from kinematic limitations by reactively adapting task-specific constraints while the robot is controlled with a low-dimensional interface (based on a framework for reactive constraint adaption [13]).

b) Feasibility checks before task execution: In our approach, we check kinematic limitations *before* the shared-control task starts. It is thus complementary to the approaches above; checking beforehand does not preclude also checking during task execution. This is indeed exemplified by our work, in which we do feasibility checks beforehand, but also use a whole-body Cartesian impedance controller to reactively extend the workspace [14]. Furthermore, reactive checking during the task does not guarantee the existence of task solutions, e.g. the user may still guide the robot EE into configurations that preclude task completion, for instance due to competing constraints.

In assistive robotics, a common aproach is to generate a policy that takes kinematic limitations into account, for instance with motion planning, and to blend user commands with policy commands [15], [16], [17].

Our main inspiration for the feasibility checks comes from autonomous robotic planning in domains such as space missions [18], industrial manufacturing [19] and daily living environments [20]. These approaches check the feasibility of an action before it is incorporated in a task plan.

As our approach checks kinematic feasibility before task execution, it does not take dynamic obstacles into account. To do so, our approach could be combined with methods that do [13], [4], [5], [6], [7].

c) Affordance-based task-space representations: Our approach includes a representation of the solution space of tasks, based on TSRs [9]. Apart from TSRs and manifold representations such as [13], our work is also inspired by the manipulation or grasp-planning communities, in concepts such as shape-templates [22] or grasp manifolds [23]. Our work explores the EE yaw angle symmetry while grasping cylindrical objects, as also explored in [24], [25].

III. Background: Shared Control Templates

This section summarizes previous work on Shared Control Templates (SCTs) [10]. An SCT is a template for providing shared control for a specific task (e.g. pouring water from *a* bottle into *a* container) that is instantiated with specific objects in a given context, e.g. pouring water from *the* bottle in the robot's hand (with id=238) into *the* red cup to the left of it (with id=301). The name derives from shared control and Action Templates for autonomous robots [26]. As this paper builds on the SCT approach, we now describe them in more technical detail.

SCTs extend *Active Constraints* (ACs), which are virtual fixtures that block the end-effector from going into certain regions [8]. In SCTs, ACs are represented as linear constraints or volumetric primitive constraints. Proxy-based methods are used to enforce these constraints [8], by projecting an end-effector (EE) pose that violates a constraint to the nearest point that does not. For example, an AC may set height limits to the EE whilst grasping. If a pose is higher than a threshold, it is projected to the threshold itself, thus respecting the height boundary. Another example is the orientation support to grasp: if the EE yaw angle with respect to the object is larger than a threshold, it is projected back to the boundary as well. Therefore, in practice the EE always points towards the object to be grasped[1].

The pose after applying the ACs is provided to a spherical linear interpolator [27], which ensures EE constraints are resolved smoothly and with velocity limits for safety. Finally, the interpolated poses are given to the robot Cartesian whole-body impedance controller [14].

Tasks usually consist of multiple phases, each requiring different constraints. SCTs for a task are therefore *Finite State Machines*, in which each phase is a state, and each state applies

[1]For examples of SCTs for everyday tasks, we refer to the video recording of the winning entry (https://youtu.be/EoER_5vYZsU) of the Assistance Robot Race (https://cybathlon.ethz.ch/en/event/disciplines/rob) of the 2023 CYBATHLON Challenges [21].

ACs relevant to that phase. Transitions between states are based on events such as distance thresholds (e.g. between the EE and an object frame) or interaction forces.

The *activation of SCTs* is determined by the distance of the end-effector to objects for which SCTs have been defined. In the default control mode, the user switches between controlling either EE translations or EE rotations with *mode switches* [28]. If the distance drops below a threshold, the first state of the SCT is activated. Our world representation continually updates a list of symbolic representations of perceived objects along with their geometric location, see [1] for details. Hierarchies of object classes are stored in the Object Database [26], and the perception pipeline is based on [29].

Advantages of SCTs include: **1)** automating transitions between task phases, which *avoids 'mode switching'* between simultaneous control of all translations and of all rotations, a key problem for users of assistive robotic arms in terms of cognitive load and time spent [28]. **2)** providing users with assistance while they remain *empowered*. For instance, users determine the speed of movement [2], [3], and tasks are never completed autonomously without an explicit request by the user to do so. **3)** being object-centric and time-independent means SCTs can be readily applied to different objects and robots [30], [31].

Limitations of SCTs include: **1)** Not taking into account obstacles and robot kinematics whilst enforcing ACs. The user can thus collide into objects or run into the joint limits or singularities of the robotic arm, reducing manipulability. **2)** SCTs are activated based on distance thresholds alone, and do not take obstacles or kinematic limitations into account which may preclude interaction with objects in practice. **3)** Representing the constraints as projection functions means the constraints are not represented explicitly.

This paper addresses limitations 1 and 2 by taking obstacles into account by running feasibility checks (Section V), and limitation 3 by using TSRs as an intermediate representation for the ACs in a SCT, as described in the next section.

IV. Task Space Regions for Shared Control

In this section, we explain the steps depicted in Fig. 1. The ultimate aim is to refine the constraints in the SCT, so that it precludes users from colliding with obstacles or run into kinematic issues, and avoids the activation of SCTs altogether if no collision-free paths are available.

A. Background: Task Space Regions

TSRs is a well-established constraint representation for motion planning [9]. A TSR $\mathcal{S}$ describes an end-effector constraint (subset of $\mathrm{SE}(3)$) given by the allowed motion of a coordinate frame w' in the TSR origin w. Formally, it consists of three parts,

$$\mathcal{S} = \{\ \mathbf{T}^0_w,\ \mathbf{B}^w\ ,\mathbf{T}^{w'}_e\}, \tag{1}$$

where $\mathbf{T}^0_w$ is the fixed transform from the world origin 0 to w, $\mathbf{T}^{w'}_e$ is a fixed frame (called an *end-effector offset*) in the coordinates of w', and $\mathbf{B}^w$ is a 6×2 matrix of bounds for the allowed movement of w' in the coordinates of w, using a Roll-Pitch-Yaw Euler angle convention:

$$\mathbf{B}^w = \begin{bmatrix} x_{\min} & y_{\min} & z_{\min} & \psi_{\min} & \theta_{\min} & \phi_{\min} \\ x_{\max} & y_{\max} & z_{\max} & \psi_{\max} & \theta_{\max} & \phi_{\max} \end{bmatrix}^\intercal \tag{2}$$

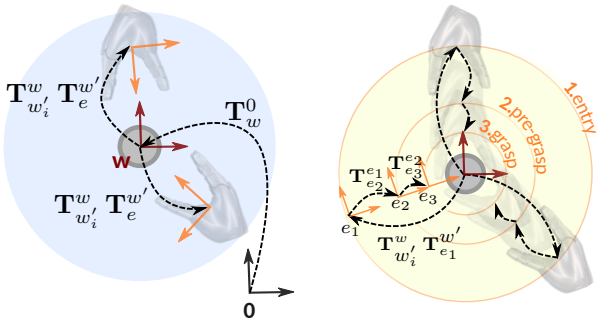
$\mathbf{B}^w$ defines a region of constrained poses $\in \mathrm{SE}(3)$. We can take samples from the region by obtaining values from the ranges of $\mathbf{B}^w$. A sample is a vector

$$\boldsymbol{b}'_i = \begin{bmatrix} x_\mathrm{i} & y_\mathrm{i} & z_\mathrm{i} & \psi_\mathrm{i} & \theta_\mathrm{i} & \phi_\mathrm{i} \end{bmatrix}^\intercal \tag{3}$$

which can be converted to a transform $\mathbf{T}^w_{w'_i} = f(\boldsymbol{b}'_i)$, as described in [9]. Then the corresponding EE pose sample $\mathbf{T}^0_{e_i}$ is obtained with

$$\mathbf{T}^0_{e_i} = \mathbf{T}^0_w\ \mathbf{T}^w_{w'_i}\ \mathbf{T}^{w'}_e \text{ with } \mathbf{T}^w_{w'_i} = f(\boldsymbol{b}'_i) \quad \text{compare Eq. (1)} \tag{4}$$

where $\mathbf{T}^0_w$ and $\mathbf{T}^{w'}_e$ are independent of the samples taken from $\mathbf{B}^w$. Fig. 2a illustrates the relevant coordinate frames of a TSR, and two samples taken from the TSR with Eq. (4).



(a) Illustration of two samples of the TSR for grasping a cylindrical container (top view). Exploiting the vertical symmetry, the frame w' (not shown) can rotate around the vertical axis of w.

(b) SCTs may have multiple subsequent states, for instance from the entry to the pregrasp, and from the pregrasp to the grasp. This is represented by a chain of transformations in the TSR.

Fig. 2: TSRs and chains of TSRs.

B. Representing Active Constraints in SCTs with TSRs

As the SCTs represent active constraints implicitly as projection functions, they do not provide a suitable (explicit) search space within which to conduct feasibility checks (Section III). Therefore, we propose to represent these ACs explicitly with TSRs, as one of the main motivations behind TSRs is to provide such an explicit search space [9].

As an example, when approaching a cylindrical container, $\mathbf{T}^0_w$ centers the frame of references on the container, and $\mathbf{T}^w_e$ projects from the bottle to the end-effector[2]. $\mathbf{B}^w$ would then represent that (*a*) the container can be approached from any yaw angle, i.e. $\phi \in [-2\pi, 2\pi]$; (*b*) the relative roll and pitch

[2]These example tasks are used in the evaluation, are shown in the video attachment, and will be illustrated in Section VI.

angle of the end-effector to the bottle is fixed, i.e. $\psi = \theta = 0$; and (c) the appropriate height of the end-effector depends on the height of the bottle $z \in [z_{\text{min}}, z_{\text{max}}]$. $\mathbf{B}^w$ would thus be

$$\mathbf{B}^w = \begin{bmatrix} 0 & 0 & z_{\text{min}} & 0 & 0 & -2\pi \\ 0 & 0 & z_{\text{max}} & 0 & 0 & 2\pi \end{bmatrix}^\intercal. \tag{5}$$

As another example, when grasping a drawer with a long horizontal handle, the end-effector can be placed anywhere along the handle, but the x and z coordinates are fixed, as well as the orientation:

$$\mathbf{B}^w = \begin{bmatrix} 0 & y_{\text{min}} & 0 & 0 & 0 & 0 \\ 0 & y_{\text{max}} & 0 & 0 & 0 & 0 \end{bmatrix}^\intercal. \tag{6}$$

SCTs consist of multiple states with different active constraints. For the running example for instance, a pre-grasp and a grasp are also represented and traversed in sequence in the different states, see Fig. 2b. To represent this with a TSR, the standard TSR representation in Eq. (1) is extended by adding $n \geq 0$ transformations to the TSR as subsequent EE offsets, i.e.

$$\mathcal{S}^* = \{\ \mathbf{T}^0_w,\ \mathbf{B}^w\ , \{\mathbf{T}^{w'}_{e^1}, \mathbf{T}^{e^1}_{e^2}, \ldots, \mathbf{T}^{e^{n-1}}_{e^n}\}\}. \tag{7}$$

Given a sample $\boldsymbol{b}'_i$ from $\mathbf{B}^w$ which is converted to $\mathbf{T}^w_{w'_i}$, the first pose in the chain is computed with Eq. (4), i.e. $\mathbf{T}^0_{e^1_i} = \mathbf{T}^0_w \mathbf{T}^w_{w'_i} \mathbf{T}^{w'}_{e^1}$, and all subsequent poses in the chain are computed with:

$$\mathbf{T}^0_{e^j_i} = \mathbf{T}^0_{e^1_i} \prod_{k=2}^{j} \mathbf{T}^{e^{k-1}}_{e^k}. \tag{8}$$

In the above, $\mathbf{T}^0_{e^j_i}$ refers to the transformation from the origin to the j^{th} viapoint and i^{th} random sample from $\mathbf{B}^w$, as illustrated in Fig. 2b for 3 viapoints and 2 samples.

C. Instantiating an SCT from a (refined) TSR

In our approach, an SCT is a template in which the active constraints are instantiated from a TSR. The procedure explained in this section applies to unrefined and refined TSRs alike, which we will leverage to instantiate refined active constraints in Section V-B.

The bounds $\mathbf{B}^w$ of the TSR define a region in end-effector space. This region is converted to a projection function that transforms poses *outside* of the region to poses *on* the region boundary, a common approach for active constraints [8]. This is straight-forward, as the boundaries of the region are linear, e.g. $z > z_{\text{min}}$, see Fig. 3 for an illustration.

Note that the first state is also responsible for guiding the end-effector from the activation pose to $\mathbf{T}^{w'}_{e^1}$, see the dashed line. From $\mathbf{T}^{w'}_{e^1}$ onwards, the user moves freely within the AC. The subsequent frames of reference $\{\mathbf{T}^{e^1}_{e^2}, \mathbf{T}^{e^2}_{e^3}, \ldots\}$ define ACs in the subsequent SCT states.

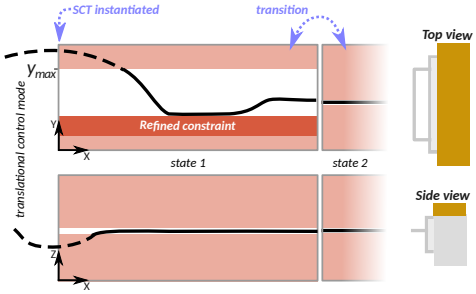


Fig. 3: Schematic of an instantiated SCT for opening a drawer. The active constraints bring the robot from the activation pose to y_{max}, and enable the user to move freely in the xy plane (top view), but fix the EE height (side view). xy not drawn to scale. A refined constraint (dark red) added from the refined TSR blocks paths that were determined to be infeasible during the feasibility checks. Once the user is close enough to the handle (defined by a distance to $\mathbf{T}^{e^1}_{e^2}$) the transition to state 2 happens. Further states complete the grasp and open the drawer, as explained in detail in Section VI.

V. Feasibility Checking and Constraint Refinement for Shared Control

TSRs are object-centric. Thus, if the object is moved, the coordinate frame w moves as well, and thus all derived frames. However, due to being only object-centric, TSRs as described so far do not take kinematic limitations into account. Thus, not all sampled end-effector poses $\mathbf{T}^0_{e^j_i}$ (for a sample $\boldsymbol{b}'_i$) may be reachable in practice, for instance due to obstacles or joint limits. The SCT instantiated from such a TSR would thus have ACs that do not preclude the user from violating kinematic limitations, which is to be avoided.

We therefore *refine* the matrix $\mathbf{B}^w_{\text{ref}} \subseteq \mathbf{B}^w$, such that samples taken from $\boldsymbol{b}'_i \sim \mathbf{B}^w_{\text{ref}}$ result in paths (through all viapoints $\mathbf{T}^0_{e^{j=1\ldots n}_i}$) that do not violate kinematic limitations given the current task context. This section describes the algorithm for acquiring $\mathbf{B}^w_{\text{ref}}$ from $\mathbf{B}^w$, which results in a refined TSR:

$$\mathcal{S}^*_{\text{ref}} = \{\ \mathbf{T}^0_w,\ \mathbf{B}^w_{\text{ref}} \subseteq \mathbf{B}^w,\ \{\mathbf{T}^{w'}_{e^1}, \mathbf{T}^{e^1}_{e^2}, \ldots, \mathbf{T}^{e^{n-1}}_{e^n}\}\}. \tag{9}$$

This algorithm is run *before* activating shared control, and consists of two main parts (illustrated in Fig. 4): A) Finding one path that does not violate kinematic limitations. If there is no such path, shared control is not activated. B) If there is one path, find further paths that do not violate kinematic limitations. Use them to compute $\mathbf{B}^w_{\text{ref}}$ for the refined TSR $\mathcal{S}^*_{\text{ref}}$. Then instantiate an SCT with the refined TSR $\mathcal{S}^*_{\text{ref}}$, and activate the refined SCT to provide shared-control support. These two parts are described in subsections A and B of this section, respectively.

A. Feasibility Checking

The aim of this part of the algorithm is to find one path (in simulation) that does not violate kinematic limitations. To describe its sub-steps, we use the abbreviation $e^1_i \ldots e^n_i$ to refer

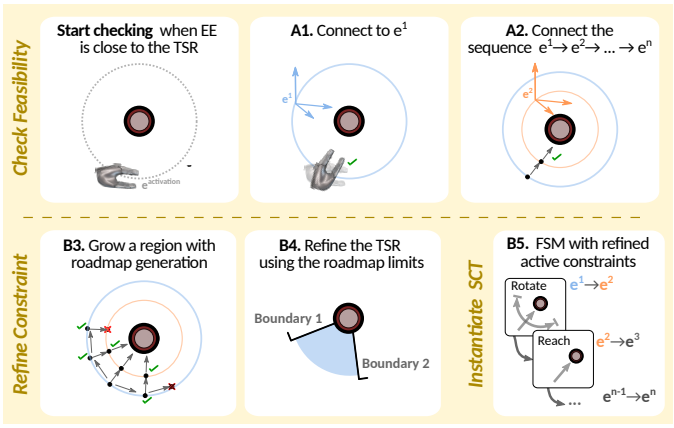


Fig. 4: Illustration of the algorithm for feasibility checking, and constraint refinement. This algorithm runs in simulation before an SCT is activated.

to the EE poses $\mathbf{T}^0_{e^1_i} \dots \mathbf{T}^0_{e^n_i}$ at the different viapoints, for a given sample i.

Initialization: As a default control mode (see Section III) the user controls either the EE translations or the EE rotations. Whenever the EE is close within a certain distance of the object (e.g. in the vicinity of a TSR first offset sample e^1_i), the feasibility checks algorithm is started. It is initialized with the current EE pose, i.e., not with that TSR sample. Thus we denote this *activation pose* as $e^{\text{activation}}_{\text{entry}}$.

Step A1: First, $e^{\text{activation}}_{\text{entry}}$(the activation pose) and e^1_{entry} (the closest pose respecting the constraints in the TSR $\mathcal{S}^*$) are connected. In the running example for instance, the hand will align towards the object; see "A1" in Fig. 4. In this paper, "connecting" means finding paths between any e^j_i and e^{j+1}_i in both EE and joint space, see the explanation after step A2. If these paths violate kinematic limitations (collisions, out-of-reach, etc.) the feasibility check fails.

Step A2: This step asserts that there is *one* solution to the task, e.g. a path from the aligned hand to the object; see "A2" in Fig. 4. We use Eq. (8) to compute all poses $e^2_{\text{entry}} \dots e^n_{\text{entry}}$ from e^1_{entry}. Then, we connect subsequent poses, first e^1_{entry} to e^2_{entry}, then e^j_{entry} to e^{j+1}_{entry} until e^n_{entry}.

Connecting and checking: The EE path is generated with the spherical linear interpolator approach used in SCTs [10], [27], as mentioned in Section III. This is to ensure consistency of the paths generated on the robot *during* shared control, and those generated *beforehand* in simulation for the feasibility checks.

We then perform a feasibility check on the corresponding path in joint space, which is computed from the EE pose path by solving the inverse kinematics locally using the Jacobian pseudo-inverse. We use Damped Least Squares [32] to prevent numerical instability near Jacobian singularities, and add joint limit avoidance as a secondary target. We check if a resulting joint-space path exists, or if it leads to joint limits or collisions. For the latter we use the FCL library [33] and a 3D model of the robot with all its links (not just the EE). FCL supports obstacles in the form of primitive shapes (e.g., a box), a 3D mesh or an Octomap [34] volume representation. We also model expected collisions (e.g. between the fingers and an object being grasped).

Infeasible tasks: We define a task as infeasible if either Step 1 or 2 fail. If this happens, the SCT is not activated for the user, and Step B3, B4, B5 are not executed.

Illustration: Feasibility checks are illustrated in the video attachment, snapshots of which are included in Fig. 5. In the starting position depicted, the previous approach would activate shared control, as the EE is close to the bottle. This is not intuitive however, as the sideboard of the shelf constitutes an obstacle between the bottle and the shelf. The feasibility checks identify this, and do not active shared control.

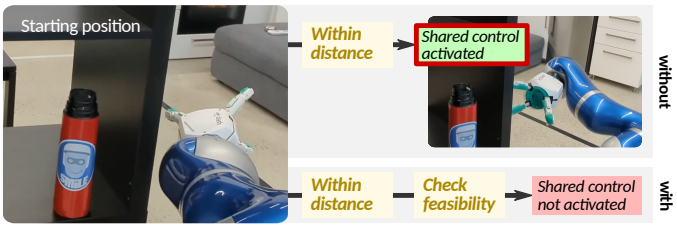


Fig. 5: Illustration of the difference between shared-control activation without (top) and with (bottom) feasibility checking.

B. Constraint Refinement and SCT Instantiation

If the task is deemed feasible in Part A of the algorithm, its result is one path that connects $e^{\text{activation}}_{\text{entry}}$and e^n_{entry} whilst respecting kinematic violations. As the constraints in TSRs and SCTs are regions rather than paths, the next step is to determine the appropriate ranges for the region in the TSR, by exploring further feasible paths from the one feasible path determined in Part A of the algorithm. In Step B5 of the algorithm, the refined regions in the TSR are converted to refined constraints in the SCT. Hence we refer to this process as 'constraint refinement', rather than 'region refinement'.

Step B3: We grow a roadmap from the initial path, see step B3 in Fig. 4. We first retrieve the vector $\boldsymbol{b}'_{\text{entry}}$, containing the position and Euler-angle orientation of the entry pose e^1_{entry} expressed in the coordinates of w (see Eq. (3)). Then, we obtain a vector $\boldsymbol{b}'_i$ in the vicinity of $\boldsymbol{b}'_{\text{entry}}$, within the bounds of $\mathbf{B}^w$. We perform two types of connections: first, we connect the first offset in $\boldsymbol{b}'_{\text{entry}}$ (i.e. e^1_{entry}) to the first offset in $\boldsymbol{b}'_i$ (i.e, e^1_i). Second, we connect all offsets $e^1_i \dots e^2_n$, akin to Step A2. If both types of connections are successful, we mark vector $\boldsymbol{b}'_i$ as feasible, and infeasible otherwise.

If $\boldsymbol{b}'_i$ is feasible, we then keep growing the roadmap by obtaining a new vector in its vicinity and repeating the connections, continuing the procedure iteratively until finding kinematic issues, obstacles, or heuristics, such as a maximum TSR size.

Step B4: This step refines $\mathbf{B}^w$ into $\mathbf{B}^w_{\text{ref}}$ using the roadmap. $\mathbf{B}^w_{\text{ref}}$ is the matrix containing the largest limits (min and max) that surrounds the set of feasible vectors $\boldsymbol{b}'_i$ in the roadmap, but such that it does not contain any infeasible vector.

Step B5: In the final step, an SCT is instantiated with the refined TSR with the procedure defined in Section IV-C. The ACs of the instantiated SCT are refined, i.e. the constraint does not include the simulated paths that were not feasible.

Illustration: Constraint refinement is illustrated in the video attachment, snapshots of which are included in Fig. 6.

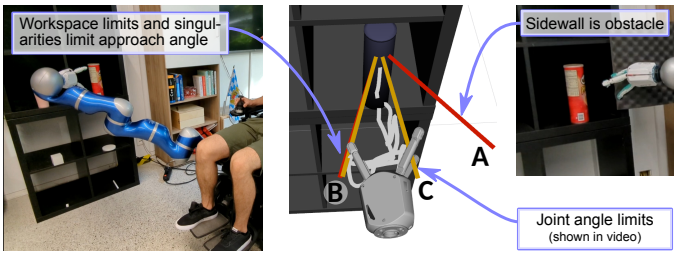


Fig. 6: The photos left and right show kinematic limitations that the user may run into without refined constraints. The refined constraints that preclude this are highlighted by the straight lines projecting from the cylindrical object in the middle. The constraint labeled A arises from the failure of paths colliding with the sidewall (right photo). B arises from workspace constraints and singularities (left photo). C arise from joint limits, highlighted only in the video. That these constraints ensure task completion is also shown in the video attachment. The white lines between the constraints B and C show EE trajectories executed by the user, highlighting that there is freedom of movement between the constraints.

VI. Experimental Evaluation

In our experiments, we used the EDAN robot [1]. It consists of a 8 degree-of-freedom robot arm with a three-fingered hand, attached to the base of a wheelchair. A set of dynamic and reactive virtual walls prevent the robot from getting close to the user. Thus, the collision models we used include both the robot and those virtual walls. As noted on each experiment, we used either the real robot or a full dynamic simulation including the controller [35].

Interface: in shared control, a 3D joystick interface was used to control the robot EE, which was commanded by one of the authors. When noted in some of the experiments below, and to prevent bias, the EE was commanded instead by an Automaton algorithm that automatically completes an SCT task by issuing joystick commands [30].

Evaluation Tasks: Three tasks – highlighted in Fig. 7 – were used in the evaluation: `Container`: grasping a tall, cylindrical container; `Mug`: grasping a mug, i.e. a short and cylindrical container; `Drawer`: grasping a drawer's handle and opening it. To test the feasibility checking, we intentionally added sources of kinematic limitations, often leading to task failure.

The TSRs for all tasks had EE offsets for *entry* ($\mathbf{T}_{e^1}^{w'}$), *pre-grasping* ($\mathbf{T}_{e^2}^{e^1}$) and *grasping* ($\mathbf{T}_{e^3}^{e^2}$). The container tasks (`Mug` and `Container`) had also a *post-grasp* transform above the object ($\mathbf{T}_{e^4}^{e^3}$). The task `Drawer` had two variations: opening a drawer with a horizontal handle (`Drawer-H`), containing an EE offsets where the drawer *is fully open and the hand releases* ($\mathbf{T}_{e^4}^{e^3}$), and another *post-release* one near the handle ($\mathbf{T}_{e^5}^{e^4}$); and a more general variant for pulling a vertical bar (`Drawer-V`) with only a *post-grasp* offset, where the object is pulled ($\mathbf{T}_{e^4}^{e^3}$).

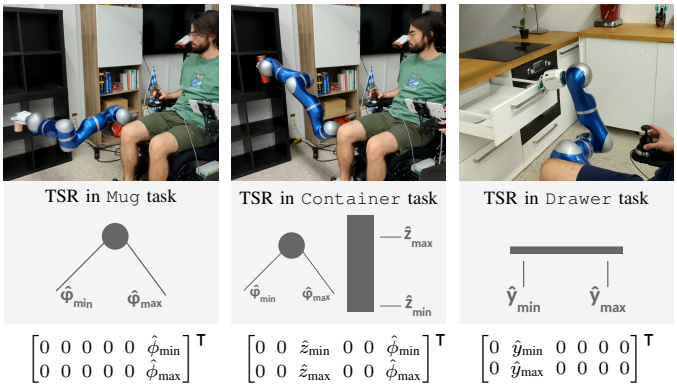


Fig. 7: The three TSRs used in the evaluation. For each task we show a schematic of the TSR, the matrix $\mathbf{B}^w$, and a snapshot of a completed task in the real robot, and The execution of all tasks is shown in the video submission.

A. Appropriate SCT Activation (simulation)

To what extent can our feasibility checks preclude activation of SCTs that are infeasible? We ran 447 trials of `Drawer-V` in the dynamic simulation, using a vertical bar with no mass nor collision meshes as a dummy target. The initial position of the bar varied in the yz plane with fixed x values (we aggregated results for $x = 0$m & $x = -0.15$m).

On each trial, an SCT was activated with an EE position sampled uniformly in the bounds of the TSR first offset (TSR max length: 0.3m), and an EE orientation was sampled uniformly from a fixed mean with pre-coded tolerances (± 0.4 roll (r), ± 1.15 pitch (p) ± 1.5 yaw (y) rad;); a corresponding robot configuration was sampled using conventional inverse kinematics. Ground-truth trials were completed by the Automaton [30]. A trial would be marked as *failed* if the robot would have a self-collision or if the EE could not reach the transitions frames projected from the active constraints using pre-coded tolerances (± 0.02m, 0.15rad), or *successful* otherwise.

In parallel to SCT activation, the feasibility checking algorithm (Step A1, A2) was run to predict whether the task would succeed or not. Fig. 8 and Table I summarize the prediction metrics of the feasibility checks as a function of the bar location. We report **accuracy** as the ratio between correct feasibility predictions and total feasibility predictions; **recall** as the ratio between the number of trials correctly predicted to fail and the total trials that failed; and **precision** as the ratio between trials correctly predicted to fail and the total trials predicted to fail. We emphasize these metrics correspond to feasibility prediction, not task success.

Sources of kinematic limitations: low manipulability in the selected workspace due to (i) proximity of the EE to the robot origin and virtual walls (e.g. in $y = 0.05\text{m}, z = 0.4\text{m}$) and (ii) reachability out of the workspace (e.g. $y = 0.65\text{m}$ or $z = 1\text{m}$); additionally: aleatory proximity to joint limits due to randomly sampled start configurations.

B. Appropriate SCT Activation (real robot)

We ran 20 trials of the `Mug` task on the real robot. The robot was initialized on an initial fixed seed configuration. The mug position was fixed. The trials were started by the experimenter driving the EE to an SCT activation pose nearby the mug, with varying EE positions (but not orientations), and were completed by the Automaton [30]. As ground truth, a trial would be marked as *successful* if the robot would grasp and lift the mug, regardless of any minor self-collision, and *failed* otherwise. In parallel to SCT activation, the feasibility checking algorithm (Step A1, A2) was run to predict task failure. Fig. 9 illustrates the ground truth and predicted failures and successes. Table I summarizes these results.

Sources of kinematic limitations: 10/20 trials were seeded from a configuration between a joint limit and the virtual walls, yielding low manipulability in the task direction. The other 10 were from a configuration with high manipulability in the task direction.

		Feasibility checks prediction		
Experiment	Trials(failed)	Accuracy	Recall	Precision
Simulation	447(280)	0.76	0.89	0.76
Real robot	20(9)	0.80	0.89	0.73

TABLE I: Effectiveness of infeasibility prediction.

C. Computation Times Benchmark

To benchmark the computation time for the feasibility checks and constraint refinement, we ran Steps A1-B5 for different robot configurations, 20 in which the task was infeasible, and 20 in which it was feasible. An EE position was sampled uniformly in the bounds of the TSR (using the first offset), and an EE orientation was sampled uniformly from pre-coded boundaries (`Mug` & `Container`: r$\pm$0.8 p$\pm$0.2 y$\pm$0.3rad, `Drawer-H`: r$\pm$0.3 p$\pm$1.58 y0.3$\pm$rad); a corresponding robot configuration was obtained using conventional inverse kinematics. We note that no SCT was instantiated for infeasible tasks. The experiment was conducted with a 3.70GHz CPU. Table II summarizes the computation times for the three tasks.

	`Mug`	`Container`	`Drawer-H`
Steps A1-A2 ($\Rightarrow$infeas.)	0.43 ± 0.16	0.63 ± 0.45	0.48 ± 0.31
Steps A1-B4 ($\Rightarrow$feas.)	1.55 ± 0.40	2.86 ± 1.04	2.17 ± 0.55
Step B5 ($\Rightarrow$feas.)	0.07 ± 0.03	0.07 ± 0.04	0.08 ± 0.03

TABLE II: Computation (s), $\mu \pm \sigma$ over 20 trials.

D. Discussion and limitations

The results in Fig. 9 and Table I show that our algorithm has high recall (0.89 in Table I), thus it can successfully prevent activation and instantiation of infeasible SCTs. In the real robot experiment, all but one of the overall failed trials (8 of 9) were successfully flagged by our feasibility checks as infeasible. This means the user is averted from ascertaining the infeasibility of these tasks through trial-and-error.

Our method does suffer from a number of false positives in some regions of the workspace, i.e. deeming a trial infeasible, even thought it is not. This results in relatively low aggregated values of precision $(0.73, 0.76)$ and accuracy $(0.76, 0.8)$ in Table I. In Fig. 9, for instance, 3 out of 20 trials were predicted to fail when they did not. In the simulated experiment this is most pronounced in the area of high task success $y = 0.2, z = 0.5$ (right graph in Fig. 8), where the robot has large reachability. We believe that this is due to differences between our kinematic model for connecting via-points and the dynamic model of the robot, since the latter can dynamically exploit the robot nullspace, and finds task solutions more often.

In future work, we will aim to increase our algorithm accuracy by using both dynamic and kinematic models in the connection step. This is akin to our previous work on assembly tasks [19], where feasibility checks are run first on a fast kinematic layer and only if necessary on a more expensive dynamic one.

Our connection algorithm explores the task region locally from the activation point. This may lead to suboptimal search spaces for the task, yielding small refined TSRs, or none. We could obtain wider solution spaces if we would search globally, e.g., bidirectionally between the goal and the start. However, this contrasts with the affordance-based nature of constraints in shared control in the related work [10], [13], [17], [12]. In future work, we want to address fallback options with global solutions in case a task is unfeasible, akin to backtracking in task and motion planning [26].

We show in Fig. 6 how the refined constraints algorithm incorporates different kinematic limitations due to obstacles (the shelf sideboard), singularities, and joint limits. While a skilled user may not need constraints to avoid the shelf, it may be useful for novel users and for users with noisy interfaces such as electromyography [1]. Also, in situations such as Fig. 9, task failure is not immediately apparent, because joint limits and virtual walls are invisible to the user. Thus, our feasibility checks can help skilled users as well.

Finally, from Table II, we see that determining if a task is infeasible took between 0.43-0.63 seconds on average. Infeasible tasks can thus be excluded quickly. Instantiating the SCT was also very fast, i.e. less than 100ms. The constraint refinement in B3/B4 takes longest, as multiple paths must be computed. In future work, we intend to decrease these times by exploiting reachability maps. We will also consider the initialization of shared control with the single path computed in Step A1-A2, which is then replaced with the entire region soon afterwards.

VII. Conclusion and Future work

In this letter we have proposed a novel shared-control framework that performs feasibility checks before task execution, and that computes refined constraints that take

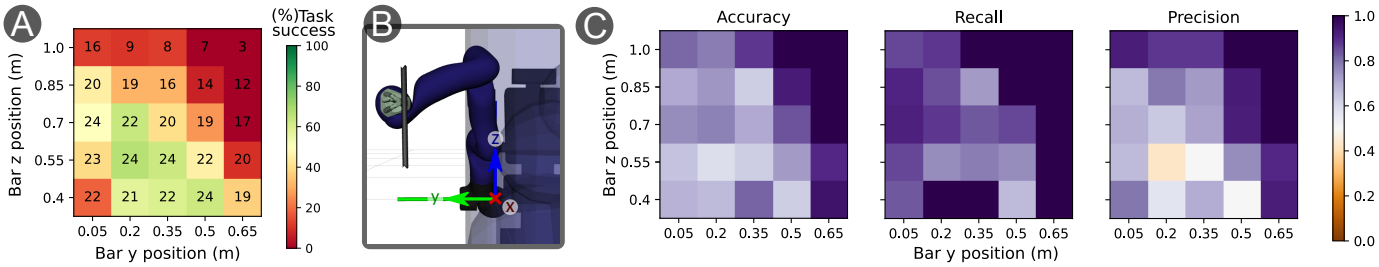


Fig. 8: **A.** Ground truth task success. The numbers in boxes indicate the amount of trials ran at each bar position in yz. The color signifies task success (%). **B.** A snapshot of the robot simulation and the dummy target in a task. **C.** Evaluation of feasibility predictions as a function of the bar position in yz with respect to the arm origin (coordinate frame in **B**).

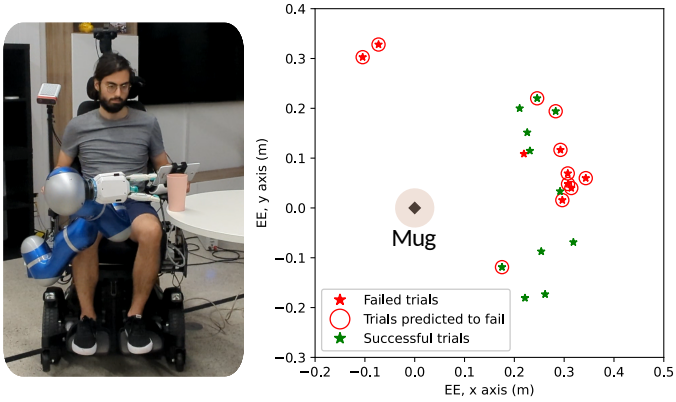


Fig. 9: Summary of the results on the real robot. Left: snapshot of the experimental setup. Right: success as a function of EE activation positions.

kinematic limitations into account. We show our framework in a set of experiments with the assistive robot EDAN, demonstrating that infeasible tasks can be identified accurately, averting users from having to discover the infeasibility through trial-and-error.

In future work we will consider dynamic obstacles, and improve explainability. We expect that explaining failed feasibility checks to the user is especially useful in situations where the kinematic limitations are not immediately obvious to the user, such as loss of manipulability due to joint limits.

References

[1] J. Vogel, A. Hagengruber, M. Iskandar, G. Quere, U. Leipscher, S. Bustamante, A. Dietrich, H. Hoeppner, D. Leidner, and A. Albu-Schäffer, "Edan - an emg-controlled daily assistant to help people with physical disabilities," in *2020 IEEE/RSJ IROS*, 2020.

[2] D.-J. Kim, R. Hazlett-Knudsen, H. Culver-Godfrey, G. Rucks, T. Cunningham, D. Portee, J. Bricout, Z. Wang, and A. Behal, "How Autonomy Impacts Performance and Satisfaction: Results From a Study With Spinal Cord Injured Subjects Using an Assistive Robot," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, no. 1, pp. 2–14, Jan. 2012.

[3] T. Bhattacharjee, E. K. Gordon, R. Scalise, M. E. Cabrera, A. Caspi, M. Cakmak, and S. S. Srinivasa, "Is More Autonomy Always Better?: Exploring Preferences of Users with Mobility Impairments in Robot-assisted Feeding," in *Proceedings of the 2020 ACM/IEEE Int'l Conference on Human-Robot Interaction*. Cambridge United Kingdom: ACM, Mar. 2020, pp. 181–190.

[4] F. Abi-Farraj, N. Pedemonte, and P. Robuffo Giordano, "A visual-based shared control architecture for remote telemanipulation," in *IEEE/RSJ IROS 2016*, 2016, pp. 4266–4273.

[5] M. Selvaggio, F. Abi-Farraj, C. Pacchierotti, P. R. Giordano, and B. Siciliano, "Haptic-based shared-control methods for a dual-arm system," *IEEE Robotics and Automation Letters*, 2018.

[6] M. Selvaggio, P. Robuffo Giordano, F. Ficuciello, and B. Siciliano, "Passive task-prioritized shared-control teleoperation with haptic guidance," in *IEEE ICRA 2019*, 2019, pp. 430–436.

[7] S. Iregui, J. De Schutter, E. Aertbeliën, "Reconfigurable constraint-based reactive framework for assistive robotics with adaptable levels of autonomy," *IEEE Robotics and Automation Letters*, 2021.

[8] S. A. Bowyer, B. L. Davies, and F. R. y Baena, "Active constraints/virtual fixtures: A survey," *Transactions on Robotics*, vol. 30, no. 1, pp. 138–157, 2013.

[9] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The Int'l Journal of Robotics Research*, vol. 30, no. 12, 2011.

[10] G. Quere, A. Hagengruber, M. Iskandar, S. Bustamante, D. Leidner, F. Stulp, and J. Vogel, "Shared Control Templates for Assistive Robotics," in *2020 IEEE ICRA*, Paris, France, 2020, p. 7.

[11] A. Campeau-Lecours and C. Gosselin, "An anticipative kinematic limitation avoidance algorithm for collaborative robots: Two-dimensional case," in *IROS 2016*, 2016, pp. 4232–4237.

[12] M. Selvaggio, G. Notomista, F. Chen, B. Gao, F. Trapani, and D. Caldwell, "Enhancing bilateral teleoperation using camera-based online virtual fixtures generation," in *IEEE/RSJ IROS 2016*, 2016.

[13] C. Vergara, S. Iregui, J. De Schutter, and E. Aertbeliën, "Generating reactive approach motions towards allowable manifolds using generalized trajectories from demonstrations," in *IEEE/RSJ IROS 2020*, 2020, pp. 9697–9704.

[14] M. Iskandar, G. Quere, A. Hagengruber, A. Dietrich, and J. Vogel, "Employing Whole-Body Control in Assistive Robotics," in *2019 IEEE/RSJ IROS*. Macau, China: IEEE, Nov. 2019, pp. 5643–5650.

[15] A. Dragan and S Srinivasa, "A policy-blending formalism for shared control," *The Int'l Journal of Robotics Research*, 2013.

[16] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The Int'l Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, Jun. 2018.

[17] K. Muelling et. al., "Autonomy infused teleoperation with application to brain computer interface controlled manipulation," *Autonomous Robots*, vol. 41, no. 6, pp. 1401–1422, Aug. 2017.

[18] I. Rodríguez, A. S. Bauer, K. Nottensteiner, D. Leidner, G. Grunwald, and M. A. Roa, "Autonomous robot planning system for in-space assembly of reconfigurable structures," in *2021 IEEE Aerospace Conference*, 2021, pp. 1–17.

[19] I. Rodríguez et. al., "Iteratively Refined Feasibility Checks in Robotic Assembly Sequence Planning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1416–1423, Apr. 2019.

[20] M. Beetz, G. Kazhoyan, and D. Vernon, "The CRAM cognitive architecture for robot manipulation in everyday activities," arXiv, 2023.

[21] L. Jaeger, R. d. S. Baptista, C. Basla, P. Capsi-Morales, Y. K. Kim, S. Nakajima, C. Piazza, M. Sommerhalder, L. Tonin, G. Valle, R. Riener, and R. Sigrist, "How the cybathlon competition has advanced assistive technologies," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 6, no. 1, pp. 447–476, 2023, assistive Robots Race: https://cybathlon.ethz.ch/en/event/disciplines/rob.

[22] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal, "Learning of grasp selection based on shape-templates," *Autonomous Robots*, vol. 36, no. 1-2, pp. 51–65, Jan. 2014.

[23] J. Hager, R. Bauer, M. Toussaint, and J. Mainprice, "Graspme - grasp manifold estimator," in *IEEE Int'l Conference on Robot & Human Interactive Communication (RO-MAN)*, 2021, p. 626–632.
[24] J. Claassens and Y. Demiris, "Exploiting affordance symmetries for task reproduction planning," in *2012 12th IEEE-RAS Humanoids 2012*, 2012.
[25] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *IEEE IROS*, 2009.
[26] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *12th IEEE-RAS (Humanoids 2012)*. IEEE, 2012.
[27] R. Weitschat, A. Dietrich, and J. Vogel, "Online motion generation for mirroring human arm motion," in *2016 IEEE ICRA*, 2016.
[28] L. V. Herlant, R. M. Holladay, and S. S. Srinivasa, "Assistive teleoperation of robot arms via automatic time-optimal mode switching," in *ACM/IEEE Int'l Conference on Human-Robot Interaction*. 2016.
[29] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel, "Augmented autoencoders: Implicit 3d orientation learning for 6d object detection," *Int'l Journal of Computer Vision*, 2020.
[30] S. Bustamante, G. Quere, K. Hagmann, X. Wu, P. Schmaus, J. Vogel, F. Stulp, and D. Leidner, "Toward seamless transitions between shared control and supervised autonomy in robotic assistance," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3833–3840, 2021.
[31] A. Padalkar, G. Quere, F. Steinmetz, A. Raffin, M. Nieuwenhuisen, J. Silvério, and F. Stulp, "Guiding reinforcement learning with shared control templates," in *IEEE ICRA 2023*. IEEE, July 2023.
[32] S. Buss, "Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods," University of California, San Diego, Tech. Rep., 2009.
[33] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE ICRA*, 2012.
[34] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, 2013.
[35] J. Vogel, J. Bayer, and P. van der Smagt, "Continuous robot control using surface electromyography of atrophic muscles," in *2013 IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, 2013.