Toward Seamless Transitions Between Shared Control and Supervised Autonomy in Robotic Assistance

S. Bustamante; G. Quere; K. Hagmann; X. Wu; P. Schmaus; J. Vogel; F. Stulp; D. Leidner

Toward Seamless Transitions Between Shared Control and Supervised Autonomy in Robotic Assistance

Samuel Bustamante, Gabriel Quere, Katharina Hagmann, Xuwei Wu, Peter Schmaus,
Jörn Vogel, Freek Stulp, Daniel Leidner

***Abstract*— Assistive robots aim to help humans with impairments execute motor tasks in everyday household environments. Controlling the end-effector of such robots directly, for instance with a joystick, is often cumbersome. Shared control methods, like Shared Control Templates (SCTs) [1], have therefore been proposed to provide support for robotic control. Moreover, depending on factors such as workload, system trust or engagement, users may like to freely adjust the level of autonomy, for instance by letting the robot complete a task by itself.**

In this paper, we present a concept for adjustable autonomy in the context of robotic assistance. We extend the SCT approach with an automatic control module that allows the user to switch between Shared Control and Supervised Autonomy at any time during task execution. As both support modes use the same action representation, transitions are seamless. We show the capabilities of this approach in a set of daily living tasks with our wheelchair-mounted robot EDAN and our humanoid robot Rollin' Justin. We highlight how automatic execution benefits from SCT features, like task-related constraints and whole-body control.

I. Introduction

Assistive robots for people with motor impairments are continually being improved. For instance, the ability of research prototypes to execute tasks autonomously is steadily increasing. This enables Supervised Autonomy, where the human issues high-level control commands to the robot to autonomously complete a specified task.

Many target users prefer control authority over the robot's movements, even if this leads to higher workloads [2]. However, directly controlling end-effector poses is cumbersome, because humans do not naturally represent movements in terms of end-effector positions in Cartesian space [3]. Therefore, direct control often requires undesirable mode switches [4] and task failure is probable.

Shared Control methods have emerged as a trade-off: they efficiently map low-dimensional user input signals to multi-dimensional goal-directed end-effector movements, enabling humans to be in full control whilst the robot provides support for completing the task [5], [6], [7]. However, depending on factors such as workload, system trust or engagement at the moment, users may like to freely adjust the level of autonomy on the fly.

This work is 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).

All authors belong to the Robotics and Mechatronics Center (RMC), German Aerospace Center (DLR), Münchner Str. 20, 82234 Weßling, Germany. Contact: `samuel.bustamante` at `dlr.de`.

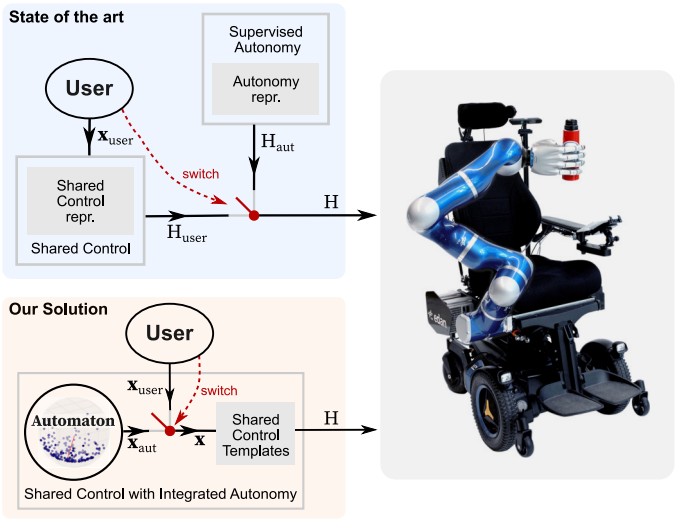

Fig. 1: We address the challenge of seamless transitioning between Shared Control and Supervised Autonomy with our robotic assistants, like EDAN (right). The State of the Art requires multiple action representations to switch between modalities (top). In contrast, our representation allows us to switch autonomy on the input commands $\mathbf{x}$, not the output (H), leading to smoother transitions (bottom).

In this paper, we extend the *spectrum of autonomy* available to users by introducing "Shared Control with Integrated Autonomy" (SCIA). We aim that SCIA allows users to seamlessly switch between Shared Control and autonomous task execution. For instance, a user may start a task in Shared Control, let the robot complete it with SCIA, and even switch back to Shared Control again later.

One challenge in achieving seamless switches between distinct modes in the autonomy spectrum is that they use different action representations, shown in Fig. 1 (top left). For instance, our implementation of Shared Control, called Shared Control Templates (SCTs [1]), is based on multi-phase geometric representations similar to those in the CARE framework [8]. In contrast, our framework for autonomous task planning and execution (Action Templates [9]) integrates geometric planning with declarative knowledge, specified in the Planning Domain Description Language [10]. Adjustable autonomy as a switch between Shared Control and Supervised Autonomy is thus a non-trivial integration problem.

To this end, SCIA extends our SCTs framework to provide an autonomous mode, and therefore both use the same underlying action representation, as seen in Fig. 1 (bottom left). This enables seamless transitions between Shared Control and Supervised Autonomy. More concretely, the module

responsible for autonomously completing the task, the so-called *automaton*, is provided with the same shared control interface as the user. In other words, SCTs are agnostic to whether the input comes from the user or the automaton.

We show how SCIA enables seamless switches to Supervised Autonomy on a set of experiments on our EMG-controlled Daily AssistaNt (EDAN), a wheelchair-robot research platform depicted in Fig. 1 (right). EDAN provides Shared Control using as input either a 3DoF joystick or surface electromyography (sEMG) signals, and SCTs. Furthermore, SCT skills reason about objects and their frames of interest, and are fully defined in task space. This enables multiple robots to use the same skill with either control mode. As a proof of concept we transfer a pouring skill developed for EDAN to our humanoid Rollin' Justin.

In summary, our contributions in this paper are: (i) providing a concept for seamless transitions between Shared Control and Supervised Autonomy in the context of robotic assistance; (ii) extending SCTs with a framework to enable these seamless transitions, as in Fig. 1; and (iii) showing how this framework enables Supervised Autonomy within an SCT skill in a set of experiments with our robots EDAN and Rollin' Justin, displaying the seamless switches.

II. Related Work

1) Shared and Traded Control: Shared Control implies that the robot control variables are jointly controlled by the human and the system, either proportionally, or split along degrees of freedom of motion. The problem of switching between Teleoperation, Shared Control and Autonomy is a recurrent topic in robotic manipulaton, and is also referred to as Traded Control.

As early as in 1989, Hayati and Venkataraman designed a robotic system with Shared and Traded Control capabilities [11]. Later, Kortenkamp et. al. [12] argued that seamless transitions between teleoperation and autonomy are difficult because the robot cannot know how the environment changes while the human is in control. Inagaki [13] suggested that control should be adapted dynamically, based on environmental factors (like safety) in different contexts. Later, Abbink et. al. [14] proposed a set of design guidelines for human-automation interaction, suggesting a system in which the human user "*always remains in control, but can experience or initiate smooth shifts between levels of automation*".

In the context of household robotic assistance, Dragan and Srinvasa [5] developed a Traded Control method as a mixture of intent inference, autonomy, and Shared Control: first, the intention of the user while on teleoperation is inferred. Then, the user input is mapped to the end-effector space, where an arbitration with the autonomy module of the robot takes place through a blending function. This formalism has been widely used, for instance by Muelling et. al. [15] on a brain-robot interface with integrated autonomy on multiple activities of daily living. Gopinath et. al. [7] integrated it in their framework for autonomy customization, which enables users to tune the autonomy capabilities of the system based on their task preferences. Javdani et. al. use a similar framework in which the robot assists an open-ended manipulation task, while it discovers the goal of the user [6]. In contrast to these works, our design of an autonomy trigger relies entirely on the human's explicit wish to trade control, and the trading of commands happens in the user input space, not in the robot configuration nor the task space (see Fig. 1).

2) Supervised autonomy: Supervised autonomy traditionally includes two elements: First, declarative knowledge, in the form of symbols, allows the robot to generate an abstract high-level plan. Second, procedural knowledge, in the form of geometrical operations, supports the robot to produce low-level motion plans and execute them. Linking these two knowledge types is non-trivial; two examples of successful applications are the Cognitive Robotics Abstract Machine [16] and the Action Templates [9].

Although Supervised Autonomy representations feature a geometric description of actions, its calculation and execution is constrained by a high-level planner. With long horizon planners like Action Templates, planning times are often large, which may lead to problems for autonomy switches. In order to have seamless switches of autonomy, we argue that the plan does not only have to be correct (i.e. it solves the task) nor geometrically smooth, but also immediate. This is a complex technical challenge.

In the context of constraint-based action representations in robotics, there has been work on generating robot autonomy behaviors. For instance Bartels et. al. [17] developed an action representation with constraints for planning, as well as a control system for autonomous task execution. Berenson et. al. [18] proposed Workspace Goal Regions, which define robot goals as intuitive volumes in the workspace, and planners to achieve them. Both of these approaches resulted in rich geometric definitions for motions, but only an agent exploited them as there is no human in the loop. Also notably, Pérez-D'Arpino and Shah [19] proposed a scheme for autonomous task execution with task-related constraints, and allowed a supervising human to make adjustments to motion plans in teleoperation.

III. Shared Control Templates

We build upon Shared Control Templates (SCTs), which we originally proposed in [1] and summarize in Fig. 2. In SCTs, a human operator controls the movements of the robot using either a 3D joystick or an sEMG-based interface [20]. Exploiting knowledge of the task and the objects in the world, the robot assists the user by mapping input commands to task-relevant robot motions. This provides an intuitive solution of the task.

A. Components

We summarize below the key aspects of this framework, and develop the building blocks of SCIA.

1) SCT input: The user input is an *input command* $\boldsymbol{x_t}$, with $\boldsymbol{x} \in \Re^3$. The rationale for using $\Re^3$ is that it allows either translational or rotational motions, and it works well with our target users using sEMG sensors (see [21] for more information).

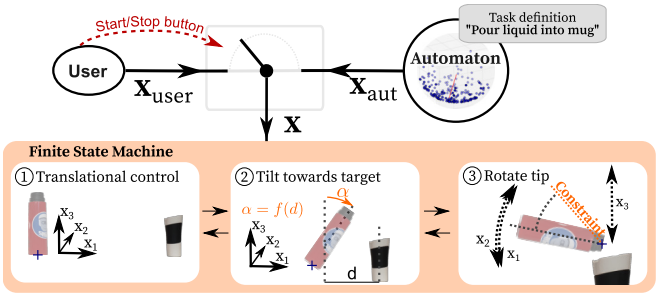

Fig. 2: A schematic of our Shared Control pipeline, and how SCIA connects to the existing infrastructure. We show the original formulation of Shared Control Templates, featuring a Finite State Machine (below) for the robot task *pour liquid* (taken from [1]). The user (top left) can issue commands $x \in \Re^3$ to move the robot. We show its components x_1, x_2, x_3 and how they are mapped to motions of the bottle, described in detail in the text. We introduce in this paper an Automaton, depicted in the top right, whose goal is to finish the task given by a task definition. The automaton is deactivated by default, but can be activated if the user presses a button, and likewise deactivated again. The automaton generates commands on the same space x as the user, which effectively means that the robot takes control over. When the automaton is active, the user command is silenced.

2) Finite State Machines (FSMs): A Shared Control Template is defined as a Finite State Machine (FSM, see Figure 2). The key elements of the framework are *states* and *transitions* between them. Each state represents a different skill phase. Transitions between states are triggered when certain pre-defined events between the objects of interest in the workspace occur. *Example:* In Figure 2, transitions depend on the distance between the mug and the bottle. The SCT thus continually monitors the distance $\boldsymbol{D}$ between the two objects, and will change the state as this distance reaches pre-defined thresholds. Distance is only one example, and we discuss more transition modalities options (like forces) in Section IV-A.1.

3) Input Mappings with Active Constraints: Each state in the FSM defines an Input Mapping (IM), which maps low-dimensional user inputs $\boldsymbol{x}$ to task-relevant end-effector motions. *Example:* During the first skill phase in Figure 2 (*translational control*), the components of $\boldsymbol{x}$ are mapped to the translation of the grasped thermos, and the user thus controls the translation of the bottle in space. In the state *rotate tip*, user inputs x_1 and x_2 map to rotations of the bottle, and x_3 to its vertical translation. Input Mappings ensure that the end-effector makes task-relevant movements, leaving the user in full control of decisions and the speed of robot motions. In many cases, it is also necessary to limit the range of movement, for instance to avoid collision or tipping the bottle over too far. Such limits are implemented as Active Constraints (AC), which are geometric limits affecting the robots end-effector pose (see [1] for details). *Example:* during the states *tilt towards target* and *rotate tip* the maximum tilt angle of the bottle is enforced as an Active Constraint, to avoid spilling liquid on the table.

IMs and ACs define the manifold of allowable end-effector poses on a given state, $M(s)$, where $M \subseteq SE(3)$ and s is a state of the FSM. In other words, each state defines a different manifold. We refer to the collection of $M(s)$ as the *Shared Control Manifold* of a task. Furthermore, the mapping of the user command is of special interest for us. We formalize this as follows: while in state s at an end-effector pose H_{t-1}, given a user command $\boldsymbol{x_t}$ the Shared Control Template outputs a new pose H_t in Cartesian space,

$$H_t = \varphi_s(\boldsymbol{x_t}, H_{t-1}), \tag{1}$$

where $\varphi : (\Re^3, SE(3)) \rightarrow SE(3)$ is the combination of IMs and ACs. Furthermore, by definition, φ fulfills the following property:

$$H_{t-1} \in M(s), \rightarrow \varphi_s(\boldsymbol{x_t}, H_{t-1}) \in M(s). \tag{2}$$

This property reflects the principle of Shared Control: through a low-dimensional user input, the robot actions stay on a Shared Control Manifold, effectively reducing the number of DoF the user has to control.

4) SCT output: The template outputs end-effector poses H_t, represented as homogeneous transformation matrices, that are sent to the robot's low-level controller. Both EDAN and Justin use a Cartesian impedance controller for the LWR arms. EDAN also includes a whole-body controller that coordinates the motion of the wheelchair (2DoF) and the modified [20] arm (8DoF) to follow the target pose H_t. More details on this approach are given in Iskandar et. al. [22].

5) Summary: An SCT supports the user in achieving a task by providing object and task-aware mappings and constraints for each state of the skill, whereby the FSM monitors progress and triggers transitions between the different phases. To ease development, an entire SCT is stored in a YAML file, which can be adapted and modified without knowledge of the underlying robot control framework (see [1] for examples).

IV. Shared Control with Integrated Autonomy

Our key idea for enabling seamless transitions between shared control and autonomous modes is to implement autonomy within an SCT. Instead of using a different action representation, we reuse the SCT, and define an *automaton* module that can provide input commands to the SCT, as illustrated in Fig. 2. During a task in SCIA the robot always stays within an SCT, and the input can be switched forth and back between the user and the automaton[1]. This means that the SCT is agnostic to whether a command $\boldsymbol{x}$ comes from either of them, and applies the same state transitions, input mappings, active constraints and whole-body control irrespectively.

A. Autonomous Execution within a State

Given a set of planned states with its different transitions, and IM/ACs in each state, the main question for autonomous execution of the task is: which sequence of input commands should be generated to pass through the different states and complete the task? Expressed from the perspective of the current state: which input commands enable the transition to the next state in the FSM? We describe next how the automaton generates these commands.

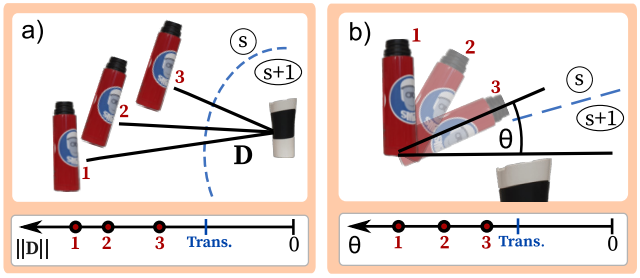

Fig. 3: Two examples of our framework's loss function. In a), the transition from state s to $s+1$ depends of the Euclidean distance $\|D\| \in \Re^1$ between the mug and the bottle. Using this metric, we can compare three different bottle positions in the real line, being bottle 3 the closest to the transition and bottle 1 the farthest. In a different example, b), the transition does not depend on the Euclidean distance, but rather on the angle θ of the bottle. Projecting the angle into the real line we see that here bottle 3 is the closest to the transition, and bottle 1 the farthest.

1) Loss function: The automaton's immediate aim is to arrive at an end-effector pose where a transition to the next state $s+1$ will be valid. We formalize this with a distance metric, $\mathcal{D} \in \Re$, that relates the current state with the event of a transition. We show two examples in Fig. 3. In a), $\mathcal{D}$ corresponds to the Euclidean distance $\|\boldsymbol{D}\|$ between the mug and the bottle. As the bottle (in hand) approaches the goal, the transition will happen when it reaches a threshold, denoted by a dashed line. Similarly, in b), $\mathcal{D}$ refers to the rotation angle θ of the thermos. More interestingly, in both cases we show how this metric allows to evaluate three different bottle poses, giving the automaton clear knowledge of when it is getting close to the next target state.

We define this metric as a loss,

$$\mathcal{L} = \mathcal{D}_s^{s+1}(H), \tag{3}$$

in order to evaluate which poses H can bring the robot closer to the goal. Note therefore that $\mathcal{L}$ will be influenced by the automaton's choice on input space $\boldsymbol{x}$, given (1).

Metric choice: Recall that this transition metric can be any measurable scalar variable that relates any two frames of interest in the workspace[2]. To name a few examples, the SCT programmer can define a transition as a threshold in (a) the vertical distance between the hand of the robot and the handle of a drawer; (b) the Euclidean distance between tip of spoon and the base of a pot; or (c), an Euler angle between the fingers and a faucet.

Wrenches: Transitions can also be based on end-effector wrenches, instead of kinematic goals. For instance, the SCT skill *release bottle* expects a vertical force when the object hits the table, and only then proceeds with subsequent states. The goal of the user is, therefore, to make the end-effector move in the direction of the expected force. In those states, the mission of the automaton is to produce commands that result in end-effector motions close to this desired trajectory. More formally, the automaton loss is defined as an angle between the motion direction resultant of a command $\boldsymbol{x_t}$, and a vector in the desired motion direction.

2) Local Optimization: With a distance metric, the goal is to find the best command to transition to the next state for any state and end-effector pose. This can be formalized as an optimization problem:

$$\boldsymbol{x}_t^* = \arg\min_{\boldsymbol{x}} \mathcal{L}_s^{s+1}(H_t). \tag{4}$$

We emphasize two points from Equation 4:

1) We minimize the loss with respect to the user command even though it is defined as a function of a frame H. Therefore we need to use the mapping from (1).
2) Recall that the automaton does not plan a series of commands: it greedily chooses the optimal command at each time step. In motion planning, such a greedy command generation could be prone to local minima.

There are two main reasons to explain the success of this procedure in the context of shared control templates. First, the motions are not complex, as they are generated within one state of the finite state machine. A simple point-to-point operation in input space often suffices to successfully traverse a state. Second, the search takes place in the 3D input space and not in the configuration space of the robot because it is *easier* to solve the task there. This intuition comes from (2): by searching on input space $\boldsymbol{x}$, using φ_s, we guarantee that the robot will stay within the manifold $M(s)$. This greatly limits the search space, because poses in $M(s)$ have a higher chance of solving the task than any random pose in $SE(3)$.

The main idea behind Shared Control is that it assists the human by allowing simple low-dimensional inputs to achieve a complex task. In our design of SCIA, the autonomous task execution benefits from this dimensionality reduction as well: not only it allows to seamlessly trade autonomy, but it also keeps the trajectories smooth, prevents collisions due to the Active Constraints, and reduces the risk of local minima. In short, we argue that the feasibility of using such a simple greedy algorithm is a feature of the shared control framework, highlighting that approaches facilitating control for humans also simplify command generation for algorithms.

Unfortunately, due to the heuristic nature of the Active Constraints, there is not a clear way for inverting φ_s from (1), nor is it simple to obtain a gradient. It is nevertheless straightforward to evaluate the mapping at any point. We therefore define a sampling-based Evolution Strategy, described next.

3) Stochastic commands: The scale (i.e., the 2-norm) of the 3D command $\boldsymbol{x}$ correlates with the length of the movement on a given time t (and therefore with the speed of the action). Without loss of generality, we disentangle the scale and the direction of the command by modeling it as a unit vector in $\Re^3$. We are interested in finding the *direction that minimizes the loss*, and set the scale as a hyper-parameter of the algorithm, $\gamma \in (0, 1]$. This is analogous to the process we follow with human commands, because the gain of the input devices (e.g., the sEMG amplifier) can be tuned according to user's preference.

[1]Examples of the switching while on a task sequence are available in the attached video.

[2]This is similar to some related work in constraint-based action representations [17].

We generate a pool of n candidate commands that we use to find the best direction in (4). As sampling randomly from the whole unit sphere is highly inefficient and can miss the minima, we aim to reduce the sampling manifold. We model the sampling pool as a von Mises-Fisher (vmF)[3] distribution,

$$\boldsymbol{x}_t \sim vMF(\boldsymbol{\mu_x}, \kappa), \tag{5}$$

from which we sample[4] the candidate commands. We thus introduce two parameters, the mean vector $\boldsymbol{\mu_x}$ and the concentration scalar κ (cf. Fig. 4).

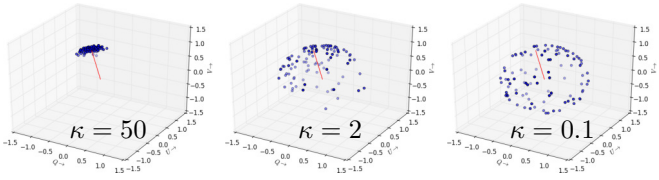

Fig. 4: Effect of the concentration parameter on vMF samples ($n = 200$). The red line is $\boldsymbol{\mu_x}$.

4) Evolution Strategy: The choice of these parameters is of special interest: κ relates to the exploration noise around the mean $\boldsymbol{\mu_x}$. It is desirable to increase n, but it is limited on the computational power available. Fixing these parameters would either make the automaton slower, or hinder the way it explores and generalizes to different scenarios.

For this reason we devised an Evolution Strategy that adapts $\boldsymbol{\mu_x}$ and κ during task execution. We summarize it in Algorithm 1 and explain it next:

Algorithm 1 Algorithm for the automaton to complete a task after an autonomy button press

Input: Current end effector pose H_0. Number of samples n. Percentage of elite samples p_{elite}. Goal state m, and plan through states $1...2...s...s+1...m$. Scaling parameter γ.
Output: Commands in user input space $\boldsymbol{x}$

```
Initialize μ_x and κ
while not on goal state m do
    Sample n unit vectors x_i from vMF(μ_x, κ)
    for each x_i do
        compute the resulting frame H_i = φ_s(γ x_i, H_{t-1})
        Li = D_s^{s+1}(H_i)
    end for
    x* = arg min_x L
    Take the p_elite best performing samples as x_elite
    Given the current μ_x, estimate a new concentration κ from x_elite
    Estimate a new μ_x from x_elite
    Clip κ in the range [0.1, 100]
    Issue the robot command γ x*
end while
```

`Step zero`: Once the autonomy button is pressed, the automaton initializes $\boldsymbol{\mu_x}$ as a unit vector in the direction between the task frames of interest (e.g. the end-effector and the task target) and $\kappa = 3$. `Step one`: the agent takes n vMF samples from (5). `Step two`: the automaton evaluates the mapping and the loss in (1) & (3) for all the samples (scaled by γ), and takes a percentage $p_{elite} = 20\%$ of the best-performing samples [5]. `Step three`: the automaton estimates a new κ based on the spread of the elite samples given the current $\boldsymbol{\mu_x}$ (i.e., the mean that generated them), and then it estimates a new $\boldsymbol{\mu_x}$ based only on them[6]. *Intuition:* If the mean direction performs really well in minimizing the loss, these elite samples will be concentrated around the mean, the confidence of the automaton on this command direction will be high, and κ should increase; if it does not, the elite samples should be sparse and point towards the minima, therefore κ should decrease to allow the agent to explore. `Step four`: the automaton issues the command that minimizes the loss (given (4), and scaled by γ), and begins again with step one by taking n samples with the new parameters. The automaton will keep updating $\boldsymbol{\mu_x}$ and κ throughout the task[7]. Furthermore, we clip κ to the range $[0.1, 100]$ to prevent excessive crunching or spreading.

To illustrate these concepts, we show the automatic completion of a grasping task on EDAN in Fig. 5. The Evolution Strategy described here rapidly converges into a command direction that solves the task, but it still allows the robot to react when the task requires a change of direction, and thus explore (and generalize to) new scenarios.

B. Task completion

Given that the automaton can now transverse a single SCT state, the process for automatic task completion is as follows: During Shared Control, the robot has symbolic knowledge of the user goal. In the case of EDAN, it is either because the user explicitly introduced it on the tablet GUI, or because the robot inferred it. EDAN has an inference pipeline that uses a robot vision system and position heuristics [20]. An example of an inferred task is: *if there is a mug on the table and the user is driving a filled bottle towards it, perhaps the user wants to start pouring some liquid*.

Given this goal and information, the robot can follow the SCT and knows the set of transitions necessary to achieve the desired goal. As an example, if during the pouring task in Fig. 2 the goal is to *rotate the tip* (state 3) and the robot is in *translational control* (state 1), the robot will get a plan: first go to state 2, then go to state 3.

If the human is commanding the actions of the robot in Shared Control, the automaton will be ready to start finishing the task. If the human presses the SCIA button, the robot immediately queries this plan. Instantly, it retrieves the current state s and the sets of transitions it needs to reach the goal; then, it starts producing input commands autonomously as explained before. Since the robot does not require re-planning, we posit that this transition is seamless. We show

[3]The vMF distribution is an analogous of a 2D normal distribution projected to a sphere, and it restricts the co-variance contours to be only circular.

[4]We use an implementation of the spherical distribution that includes rejection sampling [23].

[5]We set the value of p_{elite} the initial κ empirically. We argue $\kappa = 3$ is reasonable as it allows to explore a significant portion of the sphere.

[6]Since there is no closed form solution for a maximum likelihood estimator of the concentration parameter, we use an empirical approximation provided by Dhillion and Sra [24].

[7]Only to be reset in specific cases like task completion, or moments in which the user input is blocked to wait for the robot to finish another motion.

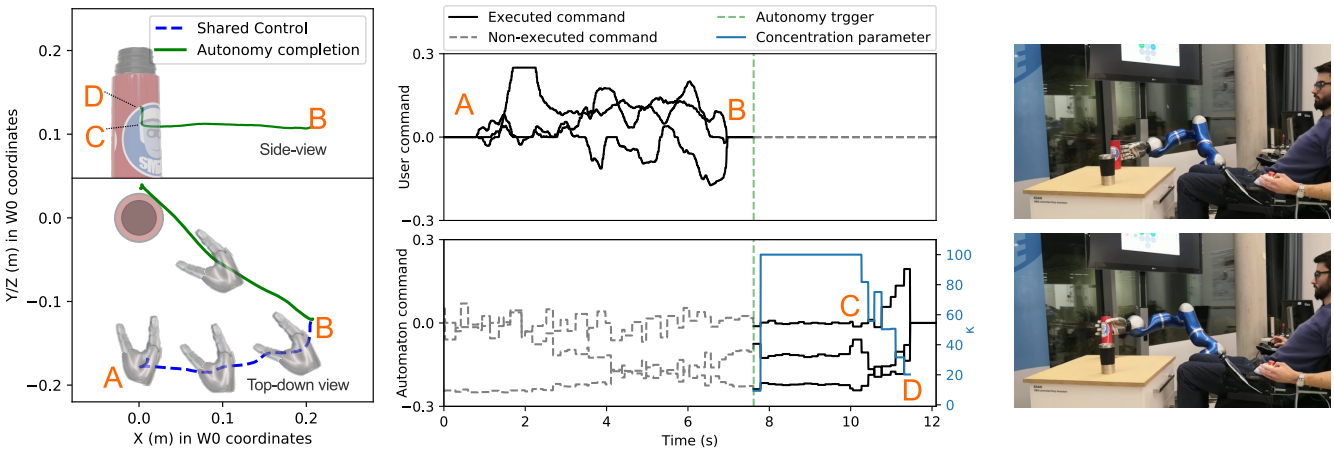

Fig. 5: Grasping task on EDAN. We show the robot's measured trajectory (left), the user/automaton commands (center) and photos of EDAN while grasping (right). In segment **AB**, the user explores the Shared Control Manifold, choosing from where to grasp. Notice that the end effector rotates towards the direction of the object, as defined in the grasping SCT skill. During this time the automaton (center) is ready on the background, and we show in gray the command it would execute if the autonomy would be started. This happens in **B**, then the automaton issues commands to finish the task. In **BC**, κ (center) increases rapidly and stays in the maximum as the confidence in the task direction is high. In **CD** a change of the task direction happens, and κ briefly decreases to allow the automaton to explore more parts of the command space. The user and automaton commands are scaled by the same factor $\gamma = 0.25$. We emphasize how the user may stop the automaton commands with a click at any time during execution.

this in Fig. 5. The robot starts moving in the direction that solves the task as soon as the user presses the autonomy trigger. Therefore, the SCIA procedure circumvents what is a known problem in adaptive autonomy, precisely that autonomous agents cannot keep track of what the human is doing during teleoperation [12]. Additionally, the user can seamlessly regain control from the robot. This will happen if either they request it within an SCT execution (e.g. with another button press) or if the automaton finishes the task.

This simple procedure can be used to enable robots to autonomously complete different tasks. Currently, our robots can use Shared Control with Integrated Autonomy on several activities of daily living. We show some examples in Fig. 5(right) and Fig. 6. We discuss the pitfalls of the local planner in Section VI.

Whole-body control: SCTs support whole-body control of the robot platform while the user is in control, which increases the workspace and reachability of the robot [22]. The automaton can also use these features when it is in control (e.g. moving EDAN's wheelchair to open a drawer).

V. Experiments

We present and discuss two sets of experiments with our robots EDAN and Rollin' Justin. These experiments aim to show how SCIA enables Supervised Autonomy after a button press, focusing on testing efficiency (set 1) and robustness (set 2) of the autonomous agent. Furthermore, we show the generality of the method by showing an example of two robots using the same skill. All runs of the automaton start in Shared Control and switch to Supervised Autonomy after a button press at the start of the trial. The experiments on EDAN were conducted with an RGB-D-based perception pipeline for detecting and estimating the pose [20] of the bottle, the mug, and the drawer.

A. Experiment set 1: Speed test

To measure the difference, we collected 10 successful trials with the automaton (triggered by the experimenter), and 10 successful trials with an expert human (EH). EH is part of the authors, does not have any motor impairment, and has previous experience with SCTs. EH was instructed to solve the task as quickly as possible.

The task was a grasp & pour experiment on EDAN with the following parameters: *Phases*: Grasp bottle from the table, lift it, pour actual water in the mug, retreat from the mug. *Initialization*: Hand, mug and bottle in fixed positions. *Success criteria*: No collision with the bottle, no water spilling, and water is transferred to the mug.

The results are shown in Table I. Task times are calculated so that a fair comparison is possible, for instance by leaving out those SCT skill states in which the experimenter had to press the autonomy button, and also those in which EDAN would move without a user command (for example, states that silence the user input while the robot would orient itself).

TABLE I: Comparison of execution times (s), $\mu \pm \sigma$ over 10 trials.

Task	Expert human	Automaton
Grasp ($p > 0.05$)	1.87 ± 0.25	2.06 ± 0.12
Pour, approach ($p < 0.001$)	7.10 ± 1.08	5.71 ± 0.08
Pour, retreat ($p > 0.05$)	8.22 ± 1.26	8.70 ± 1.01

Discussion: In one case (Pour, approach), the Automaton was significantly faster than the EH (non-parametric unpaired two-sided Mann–Whitney U-test), and in the others there was no significant difference. This confirms that the Automaton performs at least as good as the human or better, which we expect to be an important factor for acceptance.

B. Experiment set 2: Activities of the daily living

The other key aspect of seamless autonomy is the ability to complete a task successfully, from a good range of starting positions where the user could trigger it. We aim therefore to evaluate the automaton performance on different tasks while being triggered from controlled bounding boxes. Since the SCTs automatically orient the robot end-effector during the

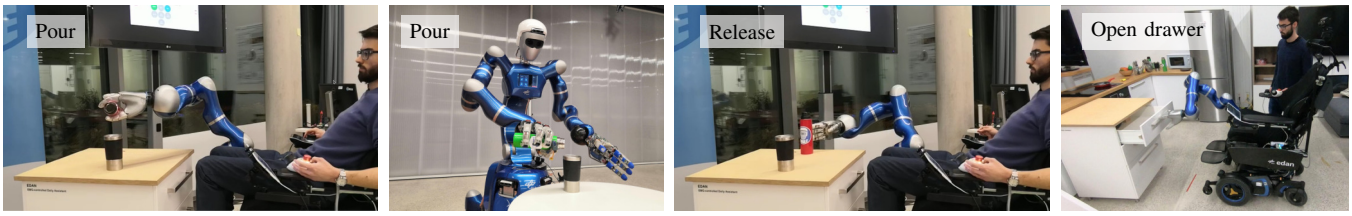

Fig. 6: Daily living tasks our robots can complete using the automaton.

task, we fixed the starting orientation of the robots. We now describe the task phases, success criteria, and initialization.

`EDAN pour`. *Phases*: While a bottle is grasped, approach the mug, pour a small rubber ball from the bottle to the mug, and then move away. *Initialization*: Mug in random position on a bounded surface in the table. EDAN's hand starting position in a random position on a bounded surface. Wheelchair position fixed. *Success Criteria*: The rubber ball is transferred successfully, no significant collision between the robot (or the bottle) and the mug.

`Justin pour`. *Phases*: Same as EDAN, using the same SCT skill with little parameter adaptation. *Initialization*: Mug and platform position fixed. Justin hand starting in a random position in a bounded surface. *Success Criteria*: Same as EDAN.

`EDAN pick & place`. *Phases*: Grasp bottle from the table, lift it a few centimeters, move down, release it and move away. *Initialization*: Bottle in random position in a control surface in the table. EDAN's hand starting position in random position in a bounded surface. Wheelchair position fixed. *Success Criteria*: No collision with the bottle and bottle grasp not extremely titled, in the experimenter's judgment. Bottle upright after releasing.

`EDAN drawer opening`. *Phases*: Open the drawer a few centimeters with active whole-body control. *Initialization*: Wheelchair position arbitrarily set by the experimenter. Hand position fixed with respect to the wheelchair (not to the drawer). *Success Criteria*: Open drawer, robot forces not exceeding a safety threshold.

We report in Table II the success rate of both robots. Similarly, we show the robot trajectories for EDAN's pick & place and pouring tasks in Fig. 7, which illustrates the bounded surfaces we used.

TABLE II: Success rate of Activities of the Daily Living. The task was performed on EDAN unless stated otherwise.

Task	Number of trials	Successful trials
Pour (with EDAN)	45	44 (98%)
Pour (with Justin)	15	14 (93%)
Grasp and Release	15	14 (93%)
Drawer opening	15	15 (100%)

Discussion: Table I and Figure 7 highlight that the automaton is effective in performing tasks of daily living, and we validate our approach with two real robotic systems. We show the capabilities of the system to allow switching from Shared Control to Supervised Autonomy and finish tasks in different settings, including a moving platform with whole-body control, and demonstrate the generality of the concept by transferring it to a different robotic system. This switch

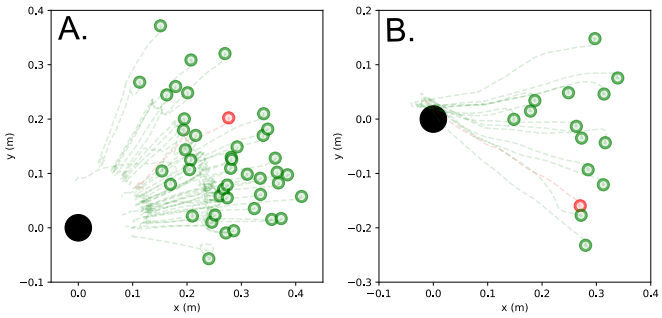

Fig. 7: Robot motions (top view) during two activities of the daily living. We depict the starting EE positions in circles (green in successful trials, red otherwise). We show in **A.** the pouring motion relative to the detected mug position (black circle, centered on (0,0)), and in **B.** the grasping motion relative to the bottle position (idem). Note that the target (mug and bottle respectively) was not located on a fixed position but on a random place of the workspace (see text for details), and we center it for illustration purposes. We do not show the retreat from pouring motion in **A.** and the release motion in **B.**

is seamless. In comparison, if the system were to stop the SCT and switch to an autonomy representation as in Fig.1 (e.g. the Action Templates) switching would be slow since a full motion plan is required, and back and forth transitions are difficult as the semantic state could not be easily shared between representations.

However, we have found empirically that the automaton success depends on whether it is started by the user in a feasible position or elsewhere. As an example, if the pouring movement in Fig. 7A. would start south west of the control volume, often it would lead to task failure. This is despite the fact that the automaton could find a task-space trajectory towards the goal, and that it would lie on the Shared Control Manifold, because the robot would have had limited manipulability or the motion results in self collision. We discuss these shortcomings in the next Section.

VI. Conclusion and Future Work

In this paper, we have proposed a novel concept for human-triggered adjustable autonomy in the context of assistive robotics, by expanding the Shared Control Templates framework to include Shared Control with Integrated Autonomy. Using the same action representation, users can seamlessly switch between Shared Control and Supervised Autonomy for task completion. In particular, we have shown how the formalism of SCTs with its virtual fixtures allow a black-box optimizer, the *automaton*, to fulfill a complex manipulation goal using local optimization. Such a framework allows the autonomous agent to keep track of the actions of the human (in Shared Control) and switch instantly if needed.

SCTs reason in task space and are object-centric. This facilitates the transfer of skills between robots. However, it entails that they are unaware of limitations of specific robots, for instance limited manipulability. In future work, we will account for manipulability by including workspace and joint limits of the specific robot platform in the automaton loss. We will then also enable avoidance of obstacles that are not relevant to the task, and are thus not addressed by the Active Constraints. This is included in motion planners like CHOMP [25] or STOMP [26]. Finally, we will further integrate the automaton with our user inference modules, and aim to increase the planning horizon of the autonomy.

This article focuses on the technical contribution of providing a framework that enables seamless switching between control modes. Our next step is to focus on the impact that this has on the usability of shared control, by conducting extensive user studies.

ACKNOWLEDGMENTS

The authors thank the EDAN and Rollin' Justin teams at DLR for productive discussion and continuous support, specially Adrian S. Bauer, Annette Hagengruber, Hanna Riesch, Maged Iskandar, Milena Eisemann and Ulrike Leipscher.

REFERENCES

[1] G. Quere, A. Hagengruber, M. Iskandar, S. Bustamante, D. Leidner, F. Stulp, and J. Vogel, "Shared control templates for assistive robotics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1956–1962.

[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. [Online]. Available: http://ieeexplore.ieee.org/document/5941028/

[3] D. Bullock, S. Grossberg, and F. H. Guenther, "A Self-Organizing Neural Model of Motor Equivalent Reaching and Tool Use by a Multijoint Arm," *Journal of Cognitive Neuroscience*, vol. 5, no. 4, pp. 408–435, Oct. 1993. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/jocn.1993.5.4.408

[4] L. V. Herlant, R. M. Holladay, and S. S. Srinivasa, "Assistive teleoperation of robot arms via automatic time-optimal mode switching," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Christchurch, New Zealand: IEEE, Mar. 2016, pp. 35–42. [Online]. Available: http://ieeexplore.ieee.org/document/7451731/

[5] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, June 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913490324

[6] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, June 2018. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364918776060

[7] D. Gopinath, S. Jain, and B. D. Argall, "Human-in-the-loop optimization of shared autonomy in assistive robotics," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 247–254, Jan. 2017. [Online]. Available: https://doi.org/10.1109/lra.2016.2593928

[8] M. Behery, "A knowledge-based activity representation for shared autonomy teleoperation of robotic arms," RWTH-Aachen University," Master's Thesis, 2016.

[9] D. S. Leidner, *Cognitive Reasoning for Compliant Robot Manipulation*, ser. Springer Tracts in Advanced Robotics. Cham: Springer International Publishing, 2019, vol. 127. [Online]. Available: http://link.springer.com/10.1007/978-3-030-04858-7

[10] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL, The Planning Version Domain Definition Language. Version 1.2," Yale Center for Computational Vision and Control, AIPS-98 Planning Competition Committee, Tech. Rep., 1998.

[11] S. Hayati and S. T. Venkataraman, "Design and implementation of a robot control system with traded and shared control capability," in *Proceedings, 1989 International Conference on Robotics and Automation*, 1989, pp. 1310–1315 vol.3.

[12] D. Kortenkamp, R. P. Bonasso, D. Ryan, and D. Schreckenghost, "Traded control with autonomous robots as mixed initiative interaction," in *AAAI Symposium on Mixed Initiative Interaction*, 1997, pp. 89–94.

[13] T. Inagaki, "Situation-adaptive autonomy: Trading control of authority in human-machine systems," in *Automation technology and human performance: Current research and trends*. Lawrence Erlbaum Associates, Inc., 1999, pp. 154–159.

[14] D. A. Abbink, M. Mulder, and E. R. Boer, "Haptic shared control: smoothly shifting control authority?" *Cognition, Technology & Work*, vol. 14, no. 1, pp. 19–28, Nov. 2011. [Online]. Available: https://doi.org/10.1007/s10111-011-0192-5

[15] K. Muelling, A. Venkatraman, J.-S. Valois, J. E. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell, "Autonomy infused teleoperation with application to brain computer interface controlled manipulation," *Autonomous Robots*, vol. 41, no. 6, pp. 1401–1422, Aug. 2017. [Online]. Available: http://link.springer.com/10.1007/s10514-017-9622-4

[16] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth, "Robotic roommates making pancakes," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*. Bled: IEEE, Oct. 2011, pp. 529–536. [Online]. Available: http://ieeexplore.ieee.org/document/6100855/

[17] G. Bartels, I. Kresse, and M. Beetz, "Constraint-based movement representation grounded in geometric features," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Atlanta, GA: IEEE, Oct. 2013, pp. 547–554. [Online]. Available: http://ieeexplore.ieee.org/document/7030027/

[18] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, "Manipulation planning with workspace goal regions," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 618–624.

[19] C. Pérez-D'Arpino and J. A. Shah, "C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4058–4065.

[20] 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 International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[21] J. Vogel and A. Hagengruber, "An sEMG-based Interface to give People with Severe Muscular Atrophy control over Assistive Devices," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Honolulu, HI: IEEE, July 2018, pp. 2136–2141. [Online]. Available: https://ieeexplore.ieee.org/document/8512689/

[22] M. Iskandar, G. Quere, A. Hagengruber, A. Dietrich, and J. Vogel, "Employing Whole-Body Control in Assistive Robotics," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China: IEEE, Nov. 2019, pp. 5643–5650. [Online]. Available: https://ieeexplore.ieee.org/document/8967772/

[23] E. Fraenkel, "Kent distribution," 2017. [Online]. Available: https://github.com/edfraenkel/kent_distribution

[24] I. S. Dhillon and S. Sra, "Modeling data using directional distributions," The University of Texas at Austin, Tech. Rep., 2003.

[25] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, Aug. 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913488805

[26] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 4569–4574. [Online]. Available: http://ieeexplore.ieee.org/document/5980280/