2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Shared Control Templates for Assistive Robotics

Gabriel Quere, Annette Hagengruber, Maged Iskandar, Samuel Bustamante,
Daniel Leidner, Freek Stulp and Jörn Vogel

***Abstract*— Light-weight robotic manipulators can be used to restore the manipulation capability of people with a motor disability. However, manipulating the environment poses a complex task, especially when the control interface is of low bandwidth, as may be the case for users with impairments. Therefore, we propose a constraint-based shared control scheme to define skills which provide support during task execution. This is achieved by representing a skill as a sequence of states, with specific user command mappings and different sets of constraints being applied in each state. New skills are defined by combining different types of constraints and conditions for state transitions, in a human-readable format. We demonstrate its versatility in a pilot experiment with three activities of daily living. Results show that even complex, high-dimensional tasks can be performed with a low-dimensional interface using our shared control approach.**

I. Introduction

The aim of assistive robotic arms is to restore manipulation capabilities of people with disabilities, thereby enabling them to perform tasks of daily living. An example is EDAN (EMG-controlled Daily AssistaNt), which consists of a DLR Light-Weight Robot III with a DLR-HIT hand, mounted on a power-wheelchair, see Fig. 1.a. Since goal-directed physical manipulation of the environment is often complex and intricate, controlling the robotic arm can be difficult, and may lead to a high cognitive workload. This is especially the case for mobile manipulation systems such as EDAN, as they have many degrees of freedom (DoFs) which all need to be controlled appropriately to achieve a task. For instance, opening and going through a door poses a real challenge, as grasping the handle is intricate, and opening the door requires the coordination of both arm and wheelchair movements [1].

Commercial systems typically make use of manual control methods, where user commands are mapped either to wheelchair motion or to subsets of robot motion (e. g. translational, rotational or gripper aperture), depending on the selected control mode. While already in use [2], manual control can vary in usability from bothersome to difficult, depending on the number of DoFs that have to be controlled in tandem to achieve the required task. This becomes even more crucial when controlling assistive devices with interfaces based on bio-signals, which often results in noisier commands and low-throughput.

All authors are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany. Email: `gabriel.quere@dlr.de`. This work is partly supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy, within the project "SMiLE" (LABAY97) and "SMiLE2gether" (LABAY102).

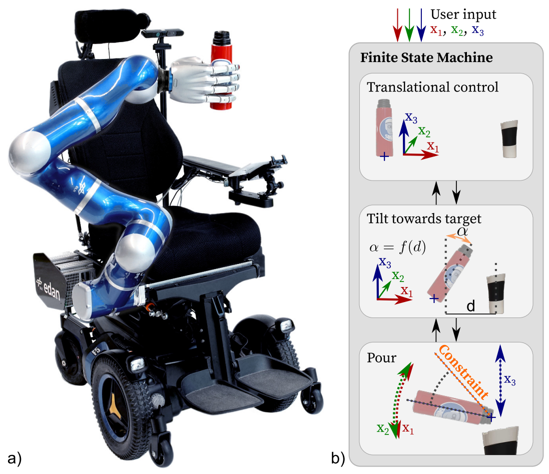

Fig. 1: *a*) The DLR assistive robotic system EDAN. *b*) The skill *Pour liquid* is divided into multiple phases each with different motion constraints and input mappings. The user is able to convey commands using a 3D signal from either a sEMG-based or a joystick-based interface.

On the other end of the spectrum, there has been substantial work aimed at achieving full robot autonomy. When given some prior knowledge, robots are able to perceive and reason, allowing planning and execution of a task desired by the user. Autonomy can, in principle, enable activities of daily living to be achieved, but in practice there are several limitations. First, it is frustrating when the robot fails to accomplish a task because of modeling errors, e. g. failing at grasping a door handle due to a decalibrated vision system. Second, it has been shown that users prefer to be in control of the robotic arm, even when this implies a higher workload for them [3].

With this in mind, we propose a shared control method that allows the user to intuitively control the end-effector along task-relevant dimensions. This provides empowerment and the ability to solve complex tasks. Our contributions are twofold. First, we design a Shared Control Template (SCT) for action representation. It defines task-specific skills as Finite State Machines in which each phase specifies task-relevant input mappings and active constraints (cf Fig. 1.b). The SCT automatically coordinate all DoFs with the whole-body controlled robotic system to achieve the task. Second, we evaluate our concept in a pilot study with three scenarios involving activities of daily living.

This paper is structured as follows: Section II presents related work. Section III describes the Shared Control Template and the definition of skills. Section IV describes the assistive robot EDAN and the components integration. In Section V, we present a pilot study with a 3-DoF joystick and a surface Electromyography (sEMG) interface [4].

II. Related work

In shared control, both user and assistive system control the state variables together, while the user decides which task to perform [5]. Teleoperating a robot requires an external device to control the robot and has been applied with shared control in various domains. Examples of interfaces with force feedback are a second robotic arm [6], [7], specifically designed controllers [8], [9] or joysticks [10].

Joysticks are also commonly used as interfaces without feedback. For example, Herlant et al. present in [11] automatic mode switching under some optimality conditions. Various interfaces adapted to disabilities investigate how to best map a low-dimensional, low-throughput signal to control a robotic arm. Broad et al. use a sip-and-puff interface to control a hierarchical FSM in [12]. With Body-Machine Interfaces [13], Jain et al. propose using piecewise virtual guiding fixtures. Vogel et al. in [14] use virtual fixtures for grasping known objects. Muelling et al. in [15] implement intent inference and capture envelopes with Brain Computer Interfaces. Blending of the user input and the assistive command can also be adaptive, for example in [16] where it depends on the robot estimate of its own confidence w.r.t the user goal and is associated with customizable input retargeting. In [17] if no commands are given the assistance progressively takes over and finishes the movement by itself. In [18] constraints are both learned and applied online.

Finally, full autonomy approaches have also been successful for many tasks, ranging from feeding [19] to cleaning [20] and pancake flipping [21]. This latter work, as well as some of the above examples, uses active constraints. Also called virtual fixtures, they are high-level control algorithms implementing virtual constraints on the robot - as opposed to mechanical ones. They are usually used to guide the user along a task-specific path, adapt the robot stiffness to the task or restrict the workspace for safety or efficiency reasons [9], [22]. In particular, an action representation based on active constraints can be found in the iTaSC framework, which uses a systematic constraint-based approach to specify complex skills [23]. Bartels et al. use it to solve *pancake flipping* by defining geometric constraints with differentiable feature functions [21]. We derive a similar description language for constraints, however while they extract a control law from closed-loop kinematic constraints, we apply user inputs to a geometrically constrained target end-effector pose, tracked with impedance control.

III. Method: Shared Control Templates

Fig. 2 provides an overview of our approach. A Shared Control Template (SCT) is a Finite State Machine (FSM) that models the different phases of a skill, e. g. 'Translational control', 'Tilt towards goal', 'Pour' as in Fig. 1.b to solve the task *Pour water*. Each phase in the FSM contains input mappings and active constraints. An Input Mapping (IM) maps the low-dimensional user inputs to task-relevant displacements of an end-effector target pose. Active Constraints (ACs) additionally constrain the target end-effector pose, to assist with successful task execution. In summary, the output of the Shared Control Template is an end-effector target pose H_{tar}, which depends on user inputs $\mathbf{x}$ and respects the motion constraints during the different phases of the skill.

This target pose is sent to a pose interpolator [24], which outputs the desired end-effector pose H_{ipol} for a whole-body impedance controller, which coordinates all DoFs of the mobile system to achieve the task [1].

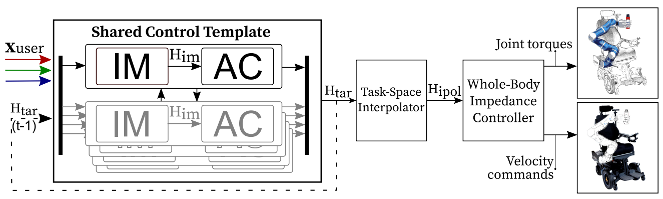

Fig. 2: System architecture. User commands are processed by an Input Mapping (IM) to apply a displacement on the precedent local target end-effector pose and obtain a new pose H$_{im}$. Active Constraints (AC) then apply constraints to obtain a new end-effector pose H$_{tar}$.

A. Input Mapping

An Input Mapping is a set of transformations (IM$_{n=1:N}$) that maps user inputs to end-effector (EE) displacements. It takes as input the target EE pose from the previous time step $H_{tar_{t-1}}$, consecutively applies the different IM$_n$, and outputs an intermediate target EE pose H_{im}. A simple mapping, for instance from a 3-DoF input, is x_1, x_2, x_3 mapped to the x, y, z translations of the end-effector. This is used in 'Translational Control' in Fig. 1.b, which allows the user to move the bottle to a desired position. On the other hand, a more convenient input mapping for pouring water (Fig. 1.b, 'Pour') is to map the x_3 input to a vertical translation, but the x_1, x_2 inputs to a rotation around a coordinate frame of interest: the thermos tip. Such phase-specific mappings facilitate the execution of the task. The procedure is illustrated in Algorithm 1 and Fig. 3:

Algorithm 1 Input Mapping

```
Input: User input $\mathbf{x}$, precedent EE target pose $H_{tar_{t-1}}$,
    Input Mapping IM
Output: Unconstrained target end-effector pose $H_{im}$
 1: $H_{im} \leftarrow H_{tar_{t-1}}$
 2: for each $input\_mapping$ in IM do
 3:     // Compute reference frame from IM
 4:     $F \leftarrow input\_mapping$.reference_frame
 5:     // Compute static transform from
 6:     // target EE pose to reference frame.
 7:     $T_F^{H_{im}} \leftarrow F^{-1} * H_{im}$
 8:     // Compute displacement of the reference frame
 9:     $F_d \leftarrow input\_mapping.map(F, \mathbf{x})$
10:     // Update target EE pose from reference frame
11:     $H_{im} \leftarrow F_d * T_F^{H_{im}}$
12: end for
13: return $H_{im}$
```

IM, as well as ACs and the overall SCT, are specified in human-readable *YAML* files. This makes it easy to develop and edit skills, without having to modify (or have knowledge

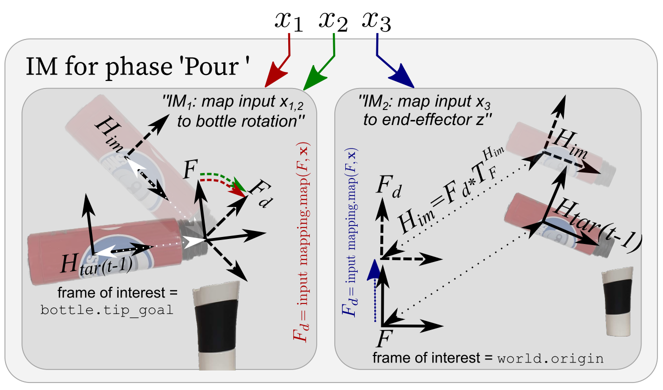
x_1 x_2 x_3
IM for phase 'Pour'
"IM$_1$: map input $x_{1,2}$ to bottle rotation"
H_{im}
F
F_d
$H_{tar(t-1)}$
frame of interest = bottle.tip_goal
F_d = input_mapping.map(F, x)
"IM$_2$: map input x_3 to end-effector z"
$H_{im} = F_d * T_F^{H_{im}}$
$H_{tar(t-1)}$
frame of interest = world.origin

Fig. 3: The two input mappings for the 'Pour' phase. Note that H poses are poses of the end-effector (not shown), not the bottle.

of) the system or control software. The template is an object-centric task representation and its main components are transforms, used for example to encode a pose or a frame of reference. In the *YAML* files, any coordinate frame of interest F can be specified, e. g. the end-effector position, the tip frame of a grasped object oriented towards the goal or the grasp frame of a target. Those frames are computed from an object database with property inheritance, cf Section IV-A.

The *YAML* snippet for the running example is listed in Listing 1. Coordinate frames are described as `[object, frame]`. `frame:[wheelchair, origin]` indicates that the frame of interest is static w. r. t. the user sitting in the wheelchair. We use a 6 DoFs pose, defining orientation as Euler angles: $[x, y, z]$ for position, $[\alpha, \beta, \gamma]$ for orientation, concatenated as a *Euler pose* $[x, y, z, \alpha, \beta, \gamma]$. This comes with known problems (non-unique solutions and Gimbal lock) but is nevertheless the most intuitive way to define local rotation constraints. `mapping:[0,0,x_3,0,0,0]` means that the user input x_3 is mapped to the z axis of the reference frame. `scaling` represents an additional scaling factor to weight the user inputs depending on the tasks most relevant motion directions. The second input mapping (Lines 5-12) uses the tip of the bottle as frame of interest. An auxiliary function `bottle_rotation` (L7-12) maps the user inputs x_1 and x_2 to a rotation of the bottle at its tip (fifth element in `mapping` L6, which represents a pitch in the frame of reference of the bottle tip). Hence, instead of moving in the xy plane, the x_1 and x_2 commands rotate the target position of the bottle.

```
1  input_mapping:
2   - frame:   [wheelchair, origin]
3     mapping: [0,0,x_3,0,0,0]
4     scaling: [0,0,1,0,0,0]
5   - frame:   [bottle, tip_goal]
6     mapping: [0,0,0,0,bottle_rotation,0]
7     auxiliary_functions:
8      bottle_rotation:
9       function: normalized_scalar
10      target_frame: [bottle, tip, X]
11      mapping: [x_1,x_2,0,0,0,0]
12      scaling: [2,2,0,0,0,0]
```

Listing 1: Input Mapping for phase 'Pour' of the skill *Pour liquid*.

The IM can be adapted to the desired amount of autonomy and the expected quality of the user input. It can range from one-dimensional virtual guide following to fine-grained multi-DoFs control.

B. Active Constraints

After computing a new target EE pose H_{im} from user inputs, various constraints defined in the *YAML* skill file can be computed. As with IM, ACs can be applied on any frame of interest F, whether they are related to the EE or to a grasped object. Additionally, these frames of interest can themselves be defined w. r. t. a static world frame, the wheelchair or a task's target pose: as an example when opening a drawer, the orientation of the end-effector depends on the drawer orientation. The Active Constraints procedure is described in Algorithm 2:

Algorithm 2 Active Constraint

```
Input: Commanded end-effector target pose $H_{im}$, Active
    Constraints ACs
Output: New target end-effector pose $H_{tar}$
 1: $H_{tar} \leftarrow H_{im}$
 2: for each $active\_constraint$ in ACs do
 3:     $F \leftarrow active\_constraint$.reference_frame
 4:     $T_F^{H_{tar}} \leftarrow F^{-1} * H_{tar}$
 5:     // Apply constraints on the reference frame
 6:     $F_c \leftarrow active\_constraint.constrain(F)$
 7:     // Update target EE pose from reference frame
 8:     $H_{tar} \leftarrow F_c * T_F^{H_{tar}}$
 9: end for
10: return $H_{tar}$
```

Our template provides three types of constraints: fixed values, function values and manifolds. A fixed value of one of the end-effector DoFs allows for example to keep the EE at a specific height. For more flexibility, functions can be used instead of fixed values: inequalities, polynomials, additions, scalings, dot products and hand-crafted functions. Inputs to those functions are distances and transforms, e. g. Fig. 4, Left, where α is a function of the distance.

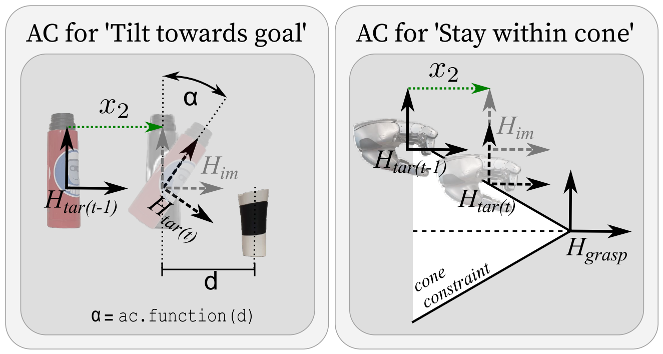

Fig. 4: Left: Constraint for phase 'Tilt towards goal'. Right: Constraint for phase 'Stay within cone'.

Another option for constraints is to use manifolds. A complete definition is out of the scope of this paper. The current manifolds are heuristically defined for the considered tasks, e. g. a cone pointing towards a target frame (Fig. 4, Right) or a cylinder for opening a door. Future work will integrate a more general representations of manifolds.

A simple implementation example is the phase ‘Pour’ in Fig. 1, where the thermos should not tilt too far to avoid spillage. A limit on the maximum tilt angle of the thermos is defined in Listing 2:

```
active_constraints:
 - frame:   [object, tip_goal]
   mapping: [x,x,x,x,tilt_angle_limit,x]
   auxiliary_functions:
    tilt_angle_limit:
     function: stay_within_range
     reference: [object, tip_goal]
     scaling : [0,0,0,0,1,0]
     range: [-inf, 0.4]
```

Listing 2: Constraint for the phase ‘Pour’. It defines an inequality: the computed angle has to stay below 0.4 rad. We limit the Euler angles pitfalls by using appropriate frame of interest for each constraint.

C. Finite State Machine

A SCT defines a FSM, which models the different phases of a skill and the transitions between them, e. g. indicated in Fig. 1.b. Transitions between the phases can depend on different metrics that relate two frames in space, typically distances between the EE and an object frame. Any number of DoFs can be used to compute the distances, in either position or orientation, and if necessary weighted. For example, consider again *Pour water* from Fig. 1.b: a transition occurs when the distance between the grasped object (the thermos) and a target object (the mug) reaches a certain threshold. Transitions can also depend on the external forces applied to the end-effector (estimated via the joint torque sensors) in directions of interest.

A transition can point to any phase and is often defined with inequalities or a range of values. It is also possible to combine transitions with OR / AND operators. The current phase is estimated recursively, which could be prompt to hysteresis and unrequited infinite loops, hence requiring careful design. The different states and transitions are described in the same *YAML* file as the IM and ACs, so that the SCT for a skill is contained in one *YAML* file.

D. Controller Parameterization

While the SCT and its modules are robot-agnostic, a successful realization will also require the definition of robot-specific parameters. In our approach, these parameters are also specified within the *YAML* file of the skill and therefore can be phase dependent. For the EDAN system, several options are available. For one, the Cartesian Impedance Controller allows adjustment of the end-effector stiffness and a definition of a null-space attractor in terms of a virtual spring attracting the elbow of the LWR. Secondly, the joint configuration and stiffness of any single DoF of the torque controlled hand can be specified. Furthermore, EDAN has a built-in safety mechanism, which reacts – if needed with a phase-dependent activation threshold – on unexpected external forces and puts the system in a maximally compliant control mode, in which only gravity compensation is active. As a result, the arm cannot exert force or torque to the environment, providing safety to the users as well as to the hardware.

Finally, we use a Whole-Body Controller (WBC) that expands the workspace of the robot arm, to allow for tasks necessitating a large range of motion, e. g. opening a door. The WBC is implemented as a low-level controller (Fig. 2) which realizes complex tasks requiring continuous coordination between the robotic arm and the wheelchair. As such, the wheelchair follows the EE to maintain manipulability of the arm as soon as the EE crosses geometric boundaries, which can be defined as skill parameters. For example, when opening the door, the wheelchair follows when the arm gets out of reach so that there is no need to switch to wheelchair control. Similarly, the wheelchair moves back when the arm gets too close to the user when opening a drawer. With this approach, the user can focus on controlling the end-effector with shared control guidance, while the local commands to the robotic arm *and* the wheelchair are generated from the WBC, reducing the users workload. More details about this feature are provided in [1].

IV. System Integration

Our concept is integrated into the EDAN system, which consists of a commercially available wheelchair on which a DLR Light-Weight Robot III is mounted, equipped with a dexterous torque controlled five-fingered DLR-HIT hand for grasping and manipulating. It is being used as a research platform for rehabilitation robotics on topics such as human robot control interfaces [4], assistive control [14] and whole-body control [1].

A. World Representation and Object Database

For the concept of object-centric action representation, we use the implementation described in [25]. Skills are defined for object classes in a database while a centralized world representation handles instances of objects. The world representation describes the robot belief of the current state of the world. Object classes are subject to inheritance: a thermos derives from the virtual class bottle, which derives from container, and skills can be defined for any of them.

For each experiment, the available instances (i. e. the objects that the robot can interact with) are localized with EDAN’s vision system, using a online bounding box object detector and depth localization, cf [1] for details.

B. User Interface

EDAN’s high-level user interface is displayed on a tablet mounted on the wheelchair. Users have two options to convey their intention: the first is a head-switch, used to switch between controllable devices: the robotic arm, the wheelchair or the tablet. The second – used by the Shared Control Templates – is a continuous 3D velocity interface. We provide a spacemouse or sEMG-based interfaces as input modalities [4]. An additional trigger signal (button click

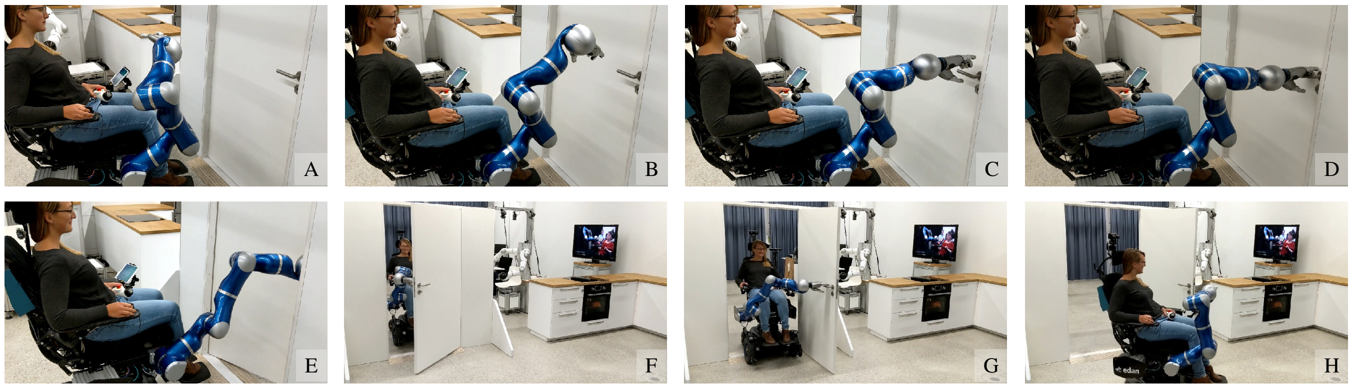

Fig. 5: Photo series of the different phases of the skill *Open door and go through* with one of our subject: approach and alignment to the door (A), open the door (B-F) and drive through (G-H), executed by means of shared control and whole-body impedance control on the EDAN system.

on the spacemouse or grasp signal with the sEMG-based interface) automatically finishes a grasp or releases an object. In tablet mode, it serves to select an action. The tablet displays measured control signals, the current task status and perception module feedback, as well as a list of available tasks, inferred from the world state.

V. Pilot Study

To test our method, users were asked to perform activities of daily living. We use those tests to illustrate the effect of our approach on the end-effector trajectories.

A. Study Design

Three common daily life tasks were considered for this pilot study: *Open drawer*, *Pour water* and *Open and go through door*. Three users tested those tasks, with two continuous 3 DoFs interfaces: spacemouse and sEMG-based. Users are able-bodied and have various levels of system experience: User A belongs to the author list, User B has sEMG-based interface experience but no system nor method knowledge and user C has neither. Users tried to accomplish each task four times, first with the spacemouse, then with the sEMG-based interface. For each task, we used three trials for training and the last one – with no live advice given – for testing. To conclude the experiment, users tried *Open drawer* and *Pour liquid* with manual control.

B. Results

To begin with, test trials with the spacemouse interface were successful for the three users. Using the sEMG-based interface, user A successfully completed all testing tasks, user B opened the drawer and went through the door multiple times, while user C did not succeed any trial. This hints that in this case the interface experience may be more relevant than the control method itself.

Time completion for the task *Pour water* for users test trials as well as expert user results, with both shared control and manual control, using a spacemouse, are shown in Table I. For both users and expert, shared control is faster than manual control. Users additionally reported preference for the shared control. We note that manual control is time-consuming for non-experienced users, partly because of the non-intuitive rotational control. Additionally, the practical

Task: *Pour water*	Users (Test average)	Expert user: Author (average of 10 trials)
Shared Control	37s	30s
Manual Control	95s	67s

TABLE I: Task *Pour water* results.

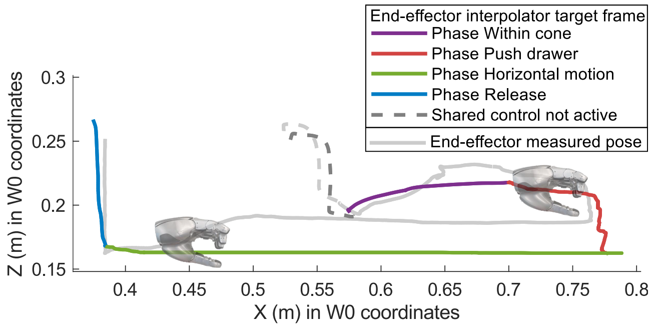

Fig. 6: Skill *Open drawer*. Side view of the trajectories resulting from the end-effector commanded and measured poses during a user test trial with an sEMG-based interface. The measured end-effector pose follows a parallel trajectory to the commanded pose due to the impedance control and the force applied on the drawer handle.

difference during execution of the skill is the need to switch control mode (translational, rotational and fingers), which happened on average 6.7 times for the expert trials. As the rotation happens around the Tool Center Point of the end-effector and not around the tip of the grasped object, there is often a need to switch the mode to correct the position of the tip to be able to pour properly, even when one has task experience.

Users failed at the task *Open drawer* in manual mode by crossing the torque safety threshold. *Open door and go through* is too complex to solve in a reasonable time with manual mode, especially with sEMG, as it demands precise commands and synchronization of the wheelchair and arm movements. An adapted manipulator would make the task easier, but decrease manipulation capabilities.

C. Framework Effect

We present user A test trials results with the sEMG-based interface to illustrate the effects of our method.

1) Open drawer: The easiest task – according to users evaluation – was the drawer opening, shown in Fig. 6. The commands are mapped to a translational displacement for

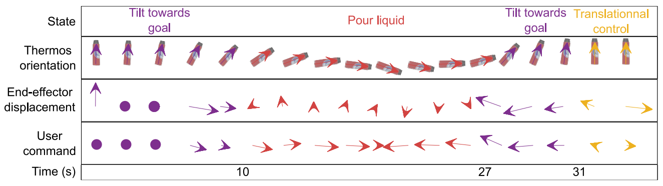

Fig. 7: Time-line of task *Pour water* during a user test trial with an sEMG-based interface.

all phases, with phase dependent scaling. During the phase 'Horizontal motion', no downward motion is possible and lateral motion are scaled down, making a clean trajectory possible, even with noisy commands.

2) Pour water: A time-line of the task 'Pour water' is shown in Fig. 7. Starting with an automatic upward motion after grasping, the phase 'Tilt towards target' constrains the tilt angle of the grasped object w. r. t. the distance to the goal. The end-effector is additionally constrained to be oriented perpendicular to the target, to make the 'Pour' motion mostly depend on the wrist joint, increasing the workspace of this specific task. Once above the bottle, phase 'Pour' maps the user input to allow rotation around the tip of the thermos. All this creates a smooth bottle orientation trajectory, conveying the shared control intent to the user (by pointing towards the estimated target) while lowering requirements on the user commands.

3) Open door and go through: Shown as a photo series in Fig. 5, the skill execution is detailed in Fig. 8, which shows the smooth constraints applied on the end-effector. Phase 'Within cone' keeps the EE within a cone pointing towards the door handle grasp frame (cf Fig. 4, Right) as well as sets an adapted EE orientation for the task. Phase 'Push door' constrains the EE in a cylindric motion, with a minimum height. The input mapping and the whole-body control allow the user to perform this complex task efficiently using mostly forward arm commands. When passing the door with only 11cm of margin, an absolute orientation controller is acting to keep the wheelchair orientation fixed, normal to the door [1]. Starting from the phase 'Push door', the user can give at any moment backwards inputs instead and close the door, with the wheelchair following backwards accordingly.

D. Discussion

These applications show the benefits of using shared control with whole-body control for tasks requiring a wide range of motions and whole-body coordination. Implementing and testing our method in a realistic setting revealed some challenges. For example some of the trials failed when crossing the safety torque threshold.

The IM in tasks *Open drawer* and *Open door* allows experienced users to correct for target pose estimation errors, while the *Pour water* skill implementation is more constrained. For the latter, the more complex input mapping creates a workspace manifold (the poses available to the EE according to the IM and ACs) not overlapping well with the space of model errors (such as the bottle position in the horizontal plane). One option is using a different IM, with

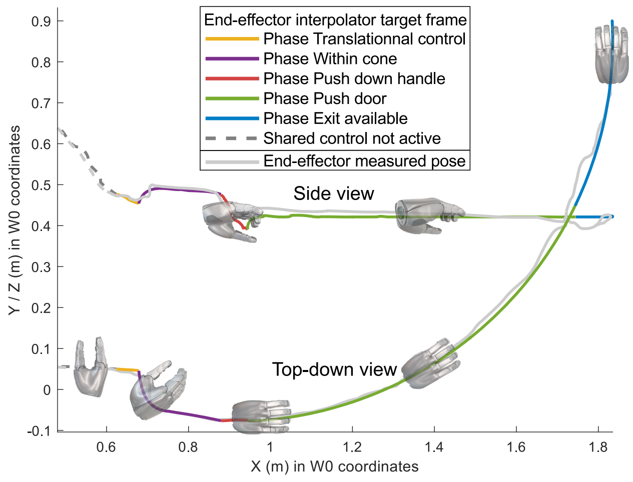

Fig. 8: Skill *Open door and go through*. Top-down and side view of the trajectories resulting from the end-effector commanded pose and measured position during a user test trial with an sEMG-based interface.

x_1 and x_2 mapped to horizontal motion while x_3 is mapped to rotation around the tip of the grasped object. This doesn't rely on the object pose estimation as much, but as a result requires a more precise user input. Alternatively we plan to investigate *correcting modes* where we could learn from user corrections using appropriately mapped commands to reduce model errors.

From a scalability standpoint, constraints generalize by using an object description hierarchy, but their descriptions tend to become long for complex skills (200 lines for *Open door and go through*). This could be alleviated with a modular hierarchical constraints description. Task time completion can only provide a partial evaluation, as full autonomy could in principle execute the task faster, but the user may prefer control authority over speed. Those results motivate a larger user study in the near future, to evaluate user preferences, method acceptance and input mapping personalization. Visualization and legibility of the FSM that represents the skills are important points for transparency and acceptability, and will also be investigated.

VI. Conclusion

In this paper, we have presented a new human-in-the-loop action representation called Shared Control Templates, and have successfully tested it with multiple users on activities of daily living. Describing shared control skills using human readable *YAML* files, SCTs define input mappings and high-level geometric constraints within phases of a finite state machine, and their output is combined with low-level whole-body impedance control to enable robotic systems with many DoFs to be intuitively controlled. As a result, this provides a safe, intuitive way of reducing the workload of the user and can be used with low-throughput interfaces. A future wheelchair-mounted robotic arm system should not only provide a shared control method, but a range of options with different degrees of user control, from manual mode to full autonomy. Task-dependent control based on user preferences should increase system utility.

References

[1] M. Iskandar, G. Quere, A. Hagengruber, A. Dietrich, and J. Vogel, "Employing Whole-Body Control in Assistive Robotics," in *Proc. of the 2019 IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2019.

[2] A. Blom and H. Stuyt, "Assistive robotic manipulators," in *Robotic Assistive Technologies: Principles and Practice*, P. Encarnação and A. Cook, Eds. Boca Raton, Fl: CRC Press, 2017, ch. 3, pp. 71–97.

[3] 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, 2011.

[4] 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)*. IEEE, 2018, pp. 2136–2141.

[5] M. R. Endsley, "Level of automation effects on performance, situation awareness and workload in a dynamic control task," *Ergonomics*, vol. 42, no. 3, pp. 462–492, 1999.

[6] M. J. A. Zeestraten, I. Havoutis, and S. Calinon, "Programming by demonstration for shared control with an application in teleoperation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1848–1855, 2018.

[7] K. Hertkorn, *Shared grasping: A combination of telepresence and grasp planning*. KIT Scientific Publishing, 2016.

[8] C. J. Pérez-del-Pulgar, J. Smisek, I. Rivas-Blanco, A. Schiele, and V. F. Muñoz, "Using gaussian mixture models for gesture recognition during haptically guided telemanipulation," *Electronics*, vol. 8, no. 7, p. 772, 2019.

[9] G. D. Hager, A. M. Okamura, P. Kazanzides, L. L. Whitcomb, G. Fichtinger, and R. H. Taylor, "Surgical and interventional robotics: part iii [tutorial]," *IEEE robotics & automation magazine*, vol. 15, no. 4, pp. 84–93, 2008.

[10] J. Artigas, R. Balachandran, C. Riecke, M. Stelzer, B. Weber, J.-H. Ryu, and A. Albu-Schaeffer, "Kontur-2: force-feedback teleoperation from the international space station," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1166–1173.

[11] L. V. Herlant, R. M. Holladay, and S. S. Srinivasa, "Assistive teleoperation of robot arms via automatic time-optimal mode switching," in *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*. IEEE Press, 2016, pp. 35–42.

[12] A. Broad and B. Argall, "Path planning under interface-based constraints for assistive robotics," in *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[19] D. Gallenberger, T. Bhattacharjee, Y. Kim, and S. S. Srinivasa, "Transfer depends on acquisition: Analyzing manipulation strategies

[13] S. Jain, A. Farshchiansadegh, A. Broad, F. Abdollahi, F. Mussa-Ivaldi, and B. Argall, "Assistive robotic manipulation through shared autonomy and a body-machine interface," in *2015 IEEE international conference on rehabilitation robotics (ICORR)*. IEEE, 2015, pp. 526–531.

[14] J. Vogel, K. Hertkorn, R. U. Menon, and M. A. Roa, "Flexible, semi-autonomous grasping for assistive robotics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4872–4879.

[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, p. 1401–1422, 2017.

[16] A. D. Dragan, S. S. Srinivasa, and K. C. T. Lee, "Teleoperation with intelligent and customizable interfaces," *Journal of Human-Robot Interaction*, vol. 2, no. 2, pp. 33–57, 2013.

[17] 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, 2018.

[18] N. Mehr, R. Horowitz, and A. D. Dragan, "Inferring and assisting with constraints in shared autonomy," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6689–6696.
for robotic feeding," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 267–276.

[20] D. S. Leidner, *Cognitive reasoning for compliant robot manipulation*. Springer, 2019.

[21] 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)*. IEEE, 2013, pp. 547–554.

[22] G. Raiola, S. S. Restrepo, P. Chevalier, P. Rodriguez-Ayerbe, X. Lamy, S. Tliba, and F. Stulp, "Co-manipulation with a library of virtual guiding fixtures," *Autonomous Robots*, vol. 42, no. 5, pp. 1037–1051, 2018.

[23] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter, "itasc: A tool for multi-sensor integration in robot manipulation," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, 2008, pp. 426–433.

[24] R. Weitschat, A. Dietrich, and J. Vogel, "Online motion generation for mirroring human arm motion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4245–4250.

[25] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 429–435.