Specification and Control of Human-Robot Handovers using Constraint-Based Programming

Maxim Vochten[1,2], Lander Vanroye[1,2], Jeroen Lambeau[1], Ken Meylemans[1], Wilm Decré[1,2], and Joris De Schutter[1,2]

[1] KU Leuven, Department of Mechanical Engineering, 3001 Leuven, Belgium.
[2] Flanders Make, Core Lab ROB, 3001 Leuven, Belgium.
maxim.vochten@kuleuven.be

Abstract. This paper introduces a constraint-based specification of an object handover in order to simplify the implementation of reactive handover controllers. A number of desired robot behaviors are identified in different phases of the handover and specified as constraints. The degrees-of-freedom in the reaching motion towards the tracked object, such as rotational symmetry in the object, are easily expressed with constraints and used by the controller. During the physical transfer, a desired force interaction and compliance can be specified. Deviations from the nominal behavior are also dealt with, such as breaking of the handover intent, a moving object and disturbance forces. The controller is validated on a real robot setup showcasing a bidirectional handover. Thanks to the modular approach of combining constraints the developed task specification can be easily extended with more reactive behaviors in the future.

Keywords: human-robot handover, constraint-based programming, reactive control, object handover, task specification

1 Introduction

Object handovers such as depicted in Figure 1 are challenging to implement on a robot since many different aspects need to be taken into account [15]. For a *human-to-robot handover*, the intent to start the handover must be detected and ideally a prediction is made where the *Object Transfer Point* (OTP) will be in space and time. The generated reaching motions towards the OTP must be safe and comfortable to the human, while a suitable grasp configuration must be planned for the particular geometry of the object. During the physical transfer of the object, the evolution of the grip force should be accurately timed, while some compliance is required to make the physical interaction feel more natural. The robot must react to unforeseen changes during the execution such as breaking off the intent for starting the handover and avoiding joint limits and collisions.

Combining all these behaviors and ensuring reactivity is challenging to achieve with conventional control approaches. In this paper, we instead make use of *constraint-based programming*. Different behaviors that are relevant to the handover are described and specified as constraints, both nominal behaviors *and* deviating reactive behaviors to deal with disturbances.

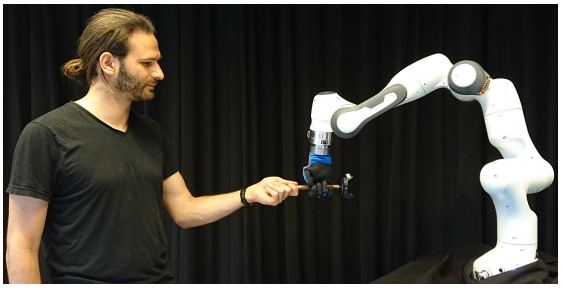


Fig. 1. Example of a handover between human and robot of a small cylindrical object.

The main contribution of this paper is the implementation of the different phases of a bidirectional human-robot handover in the constraint-based task specification and control framework of eTaSL, enabling a detailed and accurate specification of the desired robot behavior throughout the handover. When the robot is reaching towards the object, any rotational symmetry in the object or other degrees-of-freedom can be expressed and exploited by the controller. The robot's workspace is easily limited in joint- or in task-space. When human and robot are both holding the object, a desired force interaction and impedance behavior can be specified. The different phases of the handover are implemented using a finite state machine where measurements of the object's motion and the contact force are used to decide when to transition to a new phase.

The remainder of this paper is structured as follows. Section 2 discusses insights from human handover studies in literature and lists different approaches to control human-robot handovers. Section 3 elaborates the proposed constraint-based controller for implementing the handover. Section 4 shows practical results on a real robot setup, while Section 5 finishes with conclusions.

2 Related work

This section first reviews studies on human handovers in order to extract characteristic behaviors that can be implemented in the proposed controller. Afterwards, existing approaches for controlling human-robot handovers are discussed. For a broader overview on handovers we refer to a recent survey in [15].

2.1 Insights from human handover studies

Human handovers are typically divided in different phases [15, 12]. In this paper, we distinguish the following four phases: (i) the *communication phase* where an intent to start the handover is communicated and recognized, (ii) the *approach phase* where the giver and receiver are jointly moving their hands towards the intended object transfer point, (iii) the *passing phase* where physical contact is made between giver and receiver and the object is transferred, and (iv) the *retraction phase* where both actors move away from the transfer point.

During the *communication phase* the intent for starting a handover can be communicated in multiple ways such as eye gaze, body stance and speech. Recognizing these cues may help the robot to decide when the human is ready to start the interaction [21].

During the *approach phase* the reaching motions are typically smooth [2] and exhibit a minimum-jerk characteristic [9]. Implementing these minimum-jerk motions on the robot as a giver was found to improve the human's reaction and reduce the handover duration compared to trapezoidal profiles [9]. The average and peak robot movement speeds preferably remain below human speeds in order to avoid perceived discomfort [16, 8]. Besides motion, a suitable grasping configuration must be planned based on the identified object geometry [24, 21].

The *Object Transfer Point* (OTP) is the point in space and time where the object is passed between the actors. Generally, the OTP is located in the middle between the two actors with only small deviations [14]. It is therefore mainly dependent on the interpersonal distance and height. When the robot is the giver, a reaching motion and OTP should be planned that ensure comfort and safety to the human, as well as visibility of the object during the approach [19]. When the robot is the receiver, a prediction of the OTP helps to make the interaction more reactive. On-line prediction of the OTP can be achieved by measuring the human's motion and predicting the remainder using identified models such as minimum-jerk models or models learned from human demonstrations [23, 14].

In the *passing phase* the giver and receiver are both holding the object. Initially the giver has the highest load share, which is then gradually transferred to the receiver. The timing of this transition is mostly determined through visual and force feedback [4, 7]. The grip force evolves almost linearly with the estimated load share and is therefore the main driver for the evolution of the passing phase [4, 12]. Both participants also move their hands during the transfer [12]. The receiver slightly raises their hand while the giver lowers their hand. This helps signaling to the giver that the receiver is increasing their load share.

2.2 Motion planning and control of human-robot handovers

Methods for generating the reaching motion towards the OTP are commonly divided in planning-based and controller-based approaches [15, 12]. *Planning-based approaches* optimize a trajectory over a certain horizon and can range from fully pre-planned to repeated online replanning to ensure adaptability. Combinations of different planning strategies are also possible such as in [16, 11] where a global plan for grasping the object was combined with local replanning during execution. As mentioned earlier, the generated motion trajectories are preferred to be smooth, minimum-jerk trajectories. In [16], minimum-jerk motion profiles were applied in the trajectory planner based on Bézier curves.

Controller-based approaches calculate the desired robot motion at each controller time instant throughout the execution to ensure a reactive robot behavior. This is typically achieved using a goal-directed controller that always drives the robot to the estimated location of the tracked object, either with pure feedback such as visual servoing [3, 13] or with a dynamical system approach [17, 12]. The

latter combines goal feedback with feed-forward terms that shape the trajectory according to a human model. These models are typically learned from human demonstrations. Other *learning by demonstration* approaches use probability distributions to model and execute the motions [14]. An alternative learning approach based on *reinforcement learning* was used in [10] where the reward function during policy search was based on feedback of the human's preference.

The generated motions are executed by low-level joint controllers which may take additional requirements into account, for example exploiting redundancy to avoid joint limits and collisions during motion and to obtain more human-like joint configurations [18]. Interaction controllers such as impedance and admittance controllers can help to deal with unexpected contact forces during execution and to achieve a compliant behavior during the passing phase [12, 16]. For the robot hand itself, a force control strategy for the grip force in function of the estimated load force was proposed in [5]. This approach was also implemented successfully on a compliant under-actuated hand [6] that is position-controlled.

This section and the previous one listed many of the desired robot behaviors and control aspects. Implementing all of these together in a conventional controller can be very challenging. Therefore, in the next section, we propose using a constraint-based approach to specify different behaviors separately and combine them automatically in a corresponding controller.

3 Constraint-based programming of handovers

This section first introduces the general principle of constraint-based programming and control using the *expressiongraph-based Task Specification Language* (eTaSL) [1]. Afterwards, the desired behaviors for each phase of the handover are defined and specified as constraints.

3.1 Principle of constraint-based programming in eTaSL

The general principle of constraint-based programming is explained below[3]. For each behavior, a scalar-valued task function e is defined that must go to zero:

$$e(\mathbf{q}, t) \to 0, \tag{1}$$

in which $\mathbf{q}$ are the robot joint variables and t is the time. The task function is forced to evolve as a first-order system with time constant $k^{-1}[s]$:

$$\frac{d}{dt}e(\mathbf{q}, t) = -ke(\mathbf{q}, t). \tag{2}$$

After expanding the time-derivative, the resulting control law can be written as:

$$\mathbf{J}_e(\mathbf{q})\ \dot{\mathbf{q}} = -ke(\mathbf{q}, t) - \frac{\partial e(\mathbf{q}, t)}{\partial t}, \tag{3}$$

[3] For a more detailed discussion on the eTaSL framework in its most general form we refer to the original eTaSL implementation paper [1].

using the definition of Jacobian $\mathbf{J}_e(\mathbf{q}) = \frac{\partial e}{\partial \mathbf{q}}$. The first term in the control law (3) on the right side can be seen as a proportional feedback term with feedback gain k, while the second term is a feed-forward velocity term to improve the error tracking. The Jacobian $\mathbf{J}_e(\mathbf{q})$ is constructed automatically by eTaSL using *algorithmic differentiation* on the analytically defined task function $e(\mathbf{q}, t)$, using information from the kinematic model of the robot as defined in a *Universal Robot Description Format* (URDF) file.

In constraint-based programming, the controller is defined by the combination of different task functions e_i where i signifies the task number. These task functions define different behaviors and sensor-based reactions such as trajectory following, obstacle avoidance and physical interaction control. Because combining these task functions might result in conflicting joint velocities, a slack variable ϵ_i is added to each control law (3). The values are a measure of the deviation to the ideal first-order system behavior of each control law.

To determine the desired robot joint velocities for the combined control laws, eTaSL solves the following quadratic program:

$$\underset{\dot{\boldsymbol{q}},\boldsymbol{\epsilon}}{\text{minimize}} \quad \|\boldsymbol{\epsilon}\|^2_{W_\epsilon} + \mu \|\dot{\boldsymbol{q}}\|^2_{W_{\dot{q}}} \tag{4a}$$

$$\text{subject to} \quad \boldsymbol{J}_e\, \dot{\boldsymbol{q}} = -\mathbf{k} \circ \mathbf{e} - \frac{\partial \mathbf{e}}{\partial t} + \boldsymbol{\epsilon}, \tag{4b}$$

$$\boldsymbol{L} \leq \dot{\boldsymbol{q}} \leq \boldsymbol{U}. \tag{4c}$$

All control laws are now summarized in (4b) where $\boldsymbol{k}$, $\boldsymbol{e}$ and $\boldsymbol{\epsilon}$ are vectors stacking respectively the feedback constants k_i, the task function values e_i, and the slack variables ϵ_i of each task, and $\circ$ denotes the element-wise product. The inequality constraints (4c) are used to enforce the robot joint position and velocity limits, explained in more detail in [22].

The first term in the minimized cost function (4a) contains the weighted sum of the squared slack variables ϵ_i of each task. In case of conflicting constraints, the chosen weights w_i in the diagonal weighting matrix W_ϵ will determine the weighting of the constraints in the solution. The second term in (4a) is a regularization term to deal with kinematic redundancies and remaining degrees-of-freedom when the system is not fully constrained. The regularization parameter μ should be chosen small so that the regularization of joint velocities has a negligible influence on the task function constraints. The weighting matrix $W_{\dot{q}}$ can be used to decide which joints will receive more regularization than others.

The solution of the quadratic program (4) results in joint velocities $\dot{\boldsymbol{q}}$ that are sent as control inputs to the robot. The quadratic program is repeatedly solved at each time step (typically running at $200 - 400$ Hz) using the most recent sensor and system information.

The remainder of this methodology section explains the specification of the task functions that result in the necessary robot behavior during the three different phases of the handover (*reaching, passing, retracting*). Figure 2 provides an overview of the evolution of the different phases for the bidirectional handover developed in Section 4. Starting in the top-left *idle state*, the human-to-robot

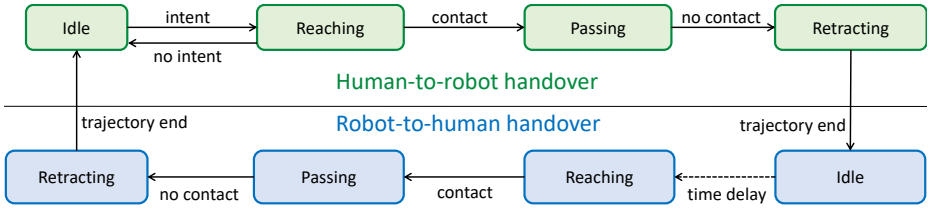


Fig. 2. Visualization of phases and transitions for a bidirectional handover.

handover is initiated as soon as an *intent* for a handover is detected from the human. The intent recognition was simplified here by defining a certain workspace in Cartesian space and checking whether the object has entered this workspace.

During the *reaching phase*, the robot moves towards the estimated object transfer point. This phase can be interrupted if the human decides to move the object outside the robot's workspace, returning again to the *idle state*. When the reaching phase progresses normally, contact between human and robot will be made after which the *passing phase* starts and the object is handed over from human to robot. Contact is detected when the measured force at the robot's wrist lies above a certain threshold. The passing phase finishes when the contact force is below another threshold. The robot now holds the object and moves towards a given location during the *retracting phase*, after which the handover is finished. The robot-to-human handover progresses similarly, except that the robot now initiates the handover by presenting the object at a given location.

3.2 Reaching phase

During the reaching phase the robot moves towards the object transfer point in free space. Figure 3 sketches the situation, where {w} is the fixed world

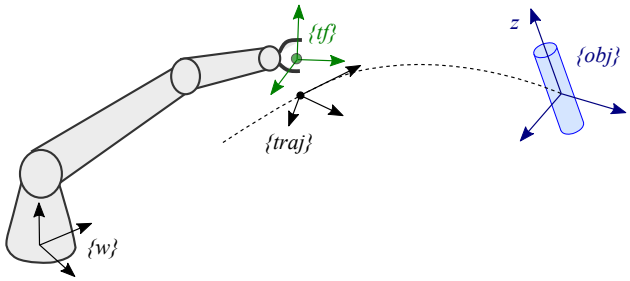


Fig. 3. Trajectory following towards object transfer point during reaching phase.

reference frame, {tf} is a task frame attached to the robot's end effector, {obj} represents the reference frame for the object transfer point and {traj} represents

the current set point on the trajectory. Ideally the generated reaching trajectory takes many aspects into account, for example smoothness, human-likeness and safety (see Section 2). Since the focus of this work is on control, the trajectory generation is simplified by calculating linear trajectories between the current pose (*i.e.* position and orientation) of the robot {tf} and the measured pose of the object {obj}. The trajectories are continuously updated while the object is moving. The generated position and orientation trajectories are represented by the position vector $\boldsymbol{p}_{\text{traj}}(t)$ and the rotation matrix $\boldsymbol{R}_{\text{traj}}(t)$ with respect to the fixed world frame {w} and as a function of time t.

To follow the reference position trajectory, the following task function is added to the controller expressing the error between the position of the robot task frame $\boldsymbol{p}_{\text{tf}}(\mathbf{q})$, which is a function of the joint angles $\mathbf{q}$, and the position of the set point on the trajectory $\boldsymbol{p}_{\text{traj}}(t)$:

$$\boldsymbol{e}_{\text{transl}} = \boldsymbol{p}_{\text{tf}}(\boldsymbol{q}) - \boldsymbol{p}_{\text{traj}}(t), \tag{5}$$

For the orientation trajectory, a similar task function is defined that expresses the error between the rotation matrix of the task frame $\boldsymbol{R}_{\text{tf}}(\mathbf{q})$ and the rotation matrix of the trajectory set point $\boldsymbol{R}_{\text{traj}}(t)$:

$$\boldsymbol{e}_{\text{rot}} = \boldsymbol{\omega}_{\text{error}}(\boldsymbol{q}, t) = \log\left(\boldsymbol{R}_{\text{traj}}^{T}(t)\ \boldsymbol{R}_{\text{tf}}(\mathbf{q})\right)^{\vee}, \tag{6}$$

where $\vee$ stands for the operator that maps the skew-symmetric matrix, resulting from the matrix logarithm, to the three-dimensional rotation error vector $\boldsymbol{\omega}_{\text{error}}$ in the axis-angle representation [20].

In special cases, the trajectory following constraints can be relaxed in order to exploit additional degrees-of-freedom in the application. For example, as shown in Figure 3, there is rotational symmetry of the object around its Z-axis. This rotational redundancy can be encoded by transforming the error vector $\boldsymbol{\omega}_{\text{error}}$ to the object frame {obj} and only constraining the X and Y components:

$$\boldsymbol{e}_{\text{rot,x}} = \begin{bmatrix}1\ 0\ 0\end{bmatrix} \boldsymbol{R}_{\text{obj}}^{T}\ \boldsymbol{\omega}_{\text{error}}, \tag{7}$$

$$\boldsymbol{e}_{\text{rot,y}} = \begin{bmatrix}0\ 1\ 0\end{bmatrix} \boldsymbol{R}_{\text{obj}}^{T}\ \boldsymbol{\omega}_{\text{error}}. \tag{8}$$

For the handover this means that the robot may approach the object from any direction rotated around the Z-axis of the object frame.

3.3 Object passing phase

During the passing phase, the human and robot make contact to transfer the object. To make the physical interaction not feel stiff and therefore more natural, a spring-like impedance behavior is implemented. This is achieved in eTaSL through the combination of position tracking and force tracking constraints.

Object translation and rotation tracking
Pose tracking constraints ensure that the robot remains at the position $\boldsymbol{p}_{\text{target}}$

and orientation $\boldsymbol{R}_{\text{target}}$ of the object as determined during the transition from reaching to passing phase. The translation and rotation constraints are implemented similarly as in (5)-(6):

$$\boldsymbol{e}_{\text{transl}} = \boldsymbol{p}_{\text{tf}}(\boldsymbol{q}) - \boldsymbol{p}_{\text{target}}, \tag{9}$$

$$\boldsymbol{e}_{\text{rot}} = \boldsymbol{R}_{\text{tf}}(\boldsymbol{q}) \log(\boldsymbol{R}_{\text{tf}}^T(\boldsymbol{q})\ \boldsymbol{R}_{\text{target}})^\vee, \tag{10}$$

except that the rotation error has now been explicitly expressed in the world frame through left-multiplication with $\boldsymbol{R}_{\text{tf}}$. This is required for later combining with force constraints in the same reference frame.

Force and moment tracking

Since eTaSL works with velocity controllers, force and moment tracking constraints cannot be applied directly. Instead, the measured force and moment $\boldsymbol{F}^{\text{meas}}$ and $\boldsymbol{M}^{\text{meas}}$ from the force sensor are related to the motion of the robot using a specified model. In this case, the force is related to the robot's motion via a compliance model, ultimately resulting in the following admittance control laws that become part of the set of control laws in (4b):

$$\frac{d\boldsymbol{p}_{\text{tf}}(\boldsymbol{q})}{dt} = -k_f C_f \left(\boldsymbol{F}^{\text{meas}} - \boldsymbol{F}^{\text{des}}\right), \tag{11}$$

$$\frac{d \log \boldsymbol{R}_{\text{tf}}(\boldsymbol{q})^\vee}{dt} = -k_m C_m \left(\boldsymbol{M}^{\text{meas}} - \boldsymbol{M}^{\text{des}}\right), \tag{12}$$

where C_f and C_m are the specified compliance matrices for force and moment, k_f and k_m are the proportional feedback control gains for force and moment, and $\boldsymbol{F}^{\text{des}}$ and $\boldsymbol{M}^{\text{des}}$ are the desired force and moment values. The left-hand side of equations (11) and (12) corresponds to the translational and rotational velocity of the robot end effector respectively. In this way, force and moment errors are transformed into a desired velocity of the robot end effector. Note that all coordinates are expressed in the world frame {w}.

Resulting impedance behavior

When the position and force tracking constraints (9)-(12) are combined in the quadratic program (4), the conflicting constraints result in an impedance behavior as visualized in Figure 4. The corresponding equivalent spring constants may be determined by deriving the necessary conditions for stationarity. These are found by substituting each slack variable ϵ_i in (4a) using their corresponding control law (3) and afterwards putting the gradient of the cost function (4a) towards the robot joint velocities $\dot{\mathbf{q}}$ to zero. Assuming that the inequalities (4c) are inactive and that the robot is in a non-singular joint configuration and neglecting the influence of the regularization term by setting $\mu = 0$, the following relations can be derived as the necessary conditions for stationarity:

$$\boldsymbol{F}^{\text{meas}} - \boldsymbol{F}^{\text{des}} = K_{\text{eq,f}}\ \left(\boldsymbol{p}_{\text{tf}}(\boldsymbol{q}) - \boldsymbol{p}_{\text{target}}\right), \tag{13}$$

$$\boldsymbol{M}^{\text{meas}} - \boldsymbol{M}^{\text{des}} = K_{\text{eq,m}}\ \boldsymbol{R}_{\text{tf}}(\boldsymbol{q}) \log(\boldsymbol{R}_{\text{tf}}^T(\boldsymbol{q})\ \boldsymbol{R}_{\text{target}})^\vee. \tag{14}$$

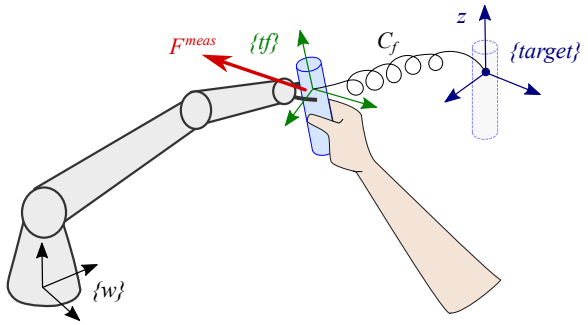


Fig. 4. Conflicting position and force constraints result in a spring-like impedance behavior during the passing phase.

In these relations, $K_{\mathrm{eq,f}}$ and $K_{\mathrm{eq,m}}$ are the *equivalent spring constants for translation and rotation* and they are expressed as a function of previously defined parameters as follows:

$$K_{\mathrm{eq,f}} = C_{\mathrm{f}}^{-1} \frac{w_{\mathrm{t}} k_{\mathrm{t}}}{w_{\mathrm{f}} k_{\mathrm{f}}}, \qquad K_{\mathrm{eq,m}} = C_{\mathrm{m}}^{-1} \frac{w_{\mathrm{r}} k_{\mathrm{r}}}{w_{\mathrm{m}} k_{\mathrm{m}}}, \tag{15}$$

with w_t, w_r, w_f, w_m the weights of the translation, rotation, force and moment constraints respectively, while k_t, k_r, k_f, k_m are the corresponding feedback constants.

3.4 Retraction phase

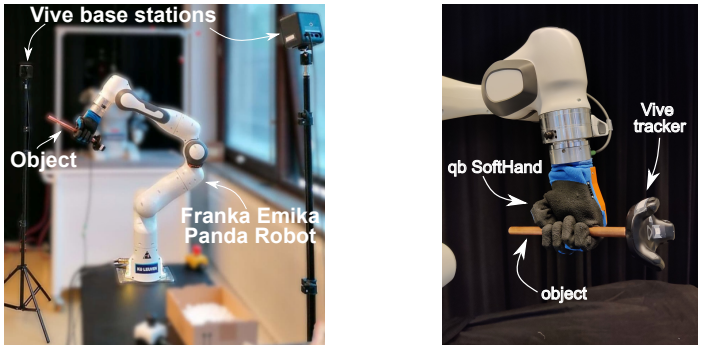
After the object has been handed over to the robot (or when the reaching phase is interrupted), the robot moves the object towards a given location. In our implementation, this was achieved as a linear trajectory $\mathbf{q}_{\mathrm{traj}}(t)$ in joint space between the current robot configuration and a predefined joint configuration. The corresponding task function is given by: $\mathbf{e}_{\mathrm{joint}} = \mathbf{q} - \mathbf{q}_{\mathrm{traj}}(t)$.

4 Experiments

The proposed constraint-based framework for bidirectional handovers is now tested on a real robot setup.

4.1 Experimental setup and parameter choices

Figure 5 visualizes the experimental setup. It consists of a 7-DoF *Franka Emika Panda* robot arm on which a *qb SoftHand* from *qbrobotics* has been mounted. This compliant hand intends to mimic human hand grasping and consists of 19 flexible joints in the fingers driven by a single actuator. The considered object is a cylindrical rod of which the position and orientation are measured online by an



(a) Robot with HTC Vive (b) Close-up of object

Fig. 5. Overview of the experimental setup. A Franka Emika Panda robot holds an object using a qb Softhand. The object consists of a cylindrical rod with an HTC Vive tracker mounted on top. The position and orientation of this tracker is measured by two static HTC Vive base stations placed on tripods.

HTC Vive measurement system using a tracker that was attached to the object. The contact force and moment during the passing phase are estimated using the internal joint torque sensors of the robot and filtered in a pre-processing step using a low-pass filter (cut-off frequency: 5 Hz) to attenuate the effect of noise.

Table 1 lists the values of the feedback constants k_i for each constraint controller of Section 3. In addition to that, the stiffness matrices C_f^{-1} and C_m^{-1}

Table 1. Chosen values for the controller feedback constants k_i in each handover phase and the stiffness matrices C_f^{-1} and C_m^{-1} in the passing phase.

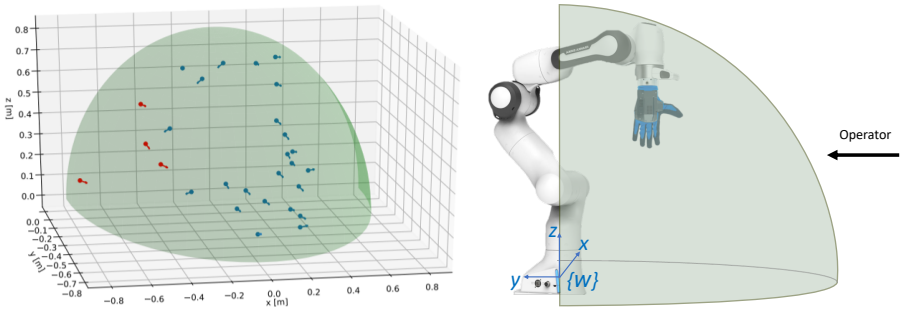
phase	*constraint*	feedback gain k_i [1/s]	impedance values
reaching (Section 3.2)	translation	2	
	rotation	2	
passing (Section 3.3)	translation	1.5	$C_f^{-1} = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 70 & 0 \\ 0 & 0 & 170 \end{bmatrix} \frac{\text{N}}{\text{m}}$, $C_m^{-1} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix} \frac{\text{Nm}}{\text{rad}}$
	rotation	1	
	force	1.5	
	moment	1	
retracting (Section 3.4)	joints	3	

for the force and moment constraints are provided of which the coordinates are expressed in reference frame {w} as depicted in Figure 6b. The stiffness in the Y-direction has been set lower than X and Z since a more compliant behavior in that direction was found to improve human-likeness. Since the conflicting constraints have the same feedback constants and the weights w_i are all set to 1, the equivalent spring constants in (15) are the same as C_f^{-1} and C_m^{-1}. Finally, in the

force and moment tracking constraints (11)-(12), all components of the desired force and moment $\boldsymbol{F}^{\text{des}}$ and $\boldsymbol{M}^{\text{des}}$ are set to zero except for the Z-component of $\boldsymbol{F}^{\text{des}}$ which is set to 10 N. This was done to accomplish the upward motion of the receiver during the passing phase as mentioned in Section 2, which helps signaling to the human that the robot is supporting the load.

4.2 Results

To validate the results a total of 28 handovers were executed. In each trial the human operator is offering the object at a different position and orientation in the workspace of the robot, which is determined as a quarter-sphere with a radius of $0.8m$ as visualized in Figure 6a.



(a) (b)

Fig. 6. (a) Successful (blue) and failed (red) attempts at handovers, drawn within the workspace of the robot (green). Each short line piece indicates the object orientation when starting the transfer. (b) The same workspace shown with respect to the robot.

Figure 7 shows snapshots of one of the experiments, where the different phases during the handover are each indicated with a number. The corresponding distance and contact force between object and robot throughout the experiment is given by the graphs in Figure 8. The robot starts in the *idle phase* ① , waiting for the object to enter the robot's workspace. In ② , the object was briefly brought inside the workspace resulting in a start of the reaching motion but was then interrupted when the object was removed again in ③ . Another approach was initiated in ④ causing the robot to again reach for the object. The *passing phase* with interaction control starts in ⑤ when a contact force above 1N is detected, closing the hand. When the hand is closed and the contact force drops below 2N, the robot *retracts* with the object in hand ⑥ . After fully retracting, the human-to-robot handover is finished.

Next, the robot-to-human handover starts with the robot presenting the object to the human ⑦ . After detecting contact, the passing phase occurs again ⑧ . After the passing phase the robot retracts ⑨ and waits for a new object

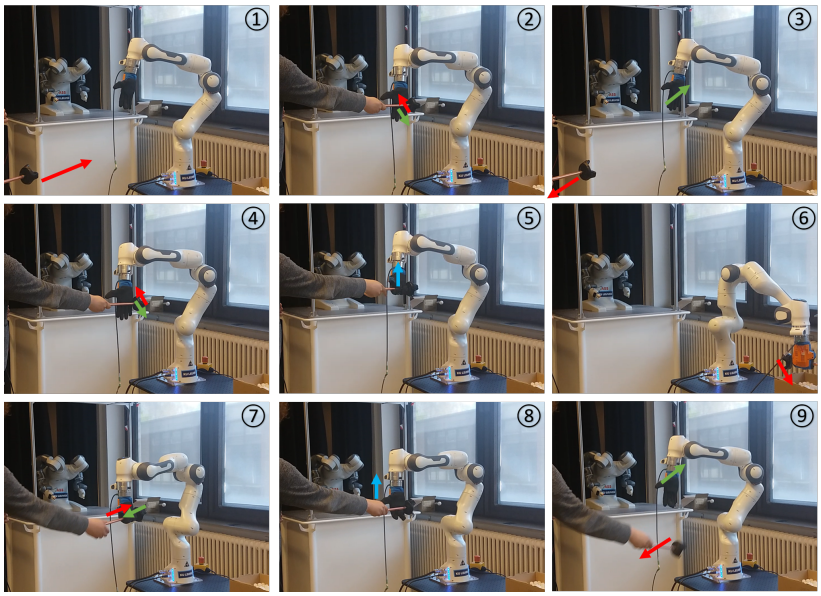


Fig. 7. Overview of different phases during a practical bidirectional handover. The human's motion is indicated with red arrows, the robot's motion with green arrows and the estimated contact force direction in blue. For the human-to-robot handover, ① - ④ corresponds to the *reaching phase*, ⑤ is the *passing phase* and ⑥ is the *retraction phase*. ⑦ - ⑨ is the robot-to-human handover.

approach, restarting again the human-to-robot handover. A video of one of the experiments can be found in the link in the footnote[4].

The success rate for all 28 handovers together is given by Figure 6a. Success was evaluated based on the fact whether the human had to correct their motion given the robot's actions. Most of the handovers succeeded except for the four on the left side of the robot's workspace. This happened because the reaching motion is calculated as a linear trajectory between the current hand pose and the object. When the object is offered to the left behind the back of the robot's hand, a collision will therefore occur. This can be remedied by better trajectory planning algorithms that take this aspect into account.

The timing of the handover was measured between the start of the approach phase and the start of the retraction phase (③ to ⑤). An average human-to-robot handover took 2.87 seconds (averaged over 12 regular handovers), and the standard deviation was 0.98 seconds.

[4] Video of one of the experiments: https://youtu.be/JdbtuFxH6QA

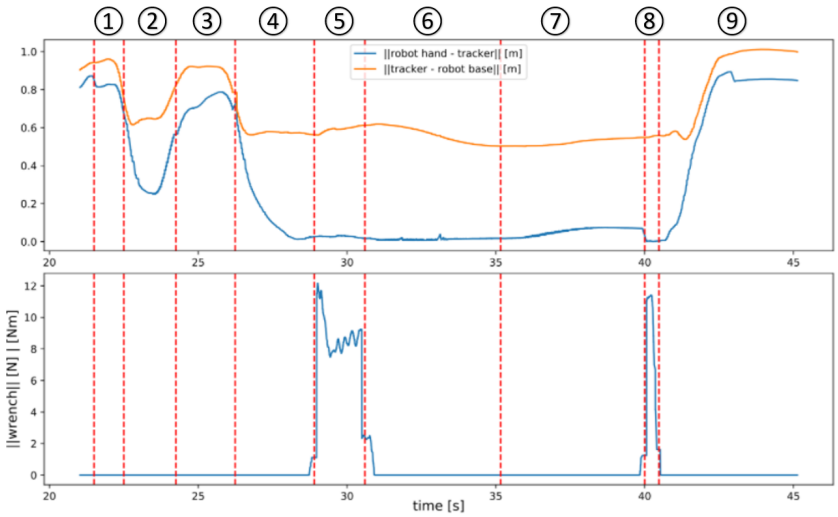


Fig. 8. Position and force evolution during a bidirectional handover. Top: distance between robot hand position and object position (orange) and distance between robot base and object (blue). Bottom: norm of the filtered contact force (blue).

5 Conclusions

The objective of this paper was to specify the control of bidirectional human-robot handovers using the formalism of constraint-based programming. The different phases of the handover (*reaching, passing, retracting*) were characterized and implemented with a corresponding set of position and force tracking constraints. The resulting method was successfully tested on the Franka Panda robot for a large number of handovers in different locations.

The main advantage of our approach is that with a simple set of constraints, many human-like behaviors during the handover are already achieved, such as taking the object's rotational symmetry into account during the reaching phase and compliance in the passing phase. Behaviors are implemented with velocity-resolved controllers. This means that no dynamical models of the robot are required which helps to transfer the task specification to other robot platforms. Since the constraint-based approach is modular, it can be readily extended with additional reactive behaviors to deal with disturbances. An example could be to add on-line obstacle avoidance or to deal with physical interaction during the reaching phase.

Acknowledgments

This result is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 788298).

Bibliography

[1] Aertbeliën E, De Schutter J (2014) etasl/etc: A constraint-based task specification language and robot controller using expression graphs. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp 1540–1546

[2] Basili P, Huber M, Brandt T, Hirche S, Glasauer S (2009) Investigating human-human approach and hand-over. In: Human centered robot systems, Springer, pp 151–160

[3] Bdiwi M, Kolker A, Suchý J (2014) Transferring model-free objects between human hand and robot hand using vision/force control. In: 2014 IEEE 11th International Multi-Conference on Systems, Signals Devices (SSD14), pp 1–6

[4] Chan WP, Parker CA, Van der Loos HM, Croft EA (2012) Grip forces and load forces in handovers: implications for designing human-robot handover controllers. In: Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, pp 9–16

[5] Chan WP, Parker CA, Van der Loos HM, Croft EA (2013) A human-inspired object handover controller. The International Journal of Robotics Research 32(8):971–983

[6] Chan WP, Kumagai I, Nozawa S, Kakiuchi Y, Okada K, Inaba M (2014) Implementation of a robot-human object handover controller on a compliant underactuated hand using joint position error measurements for grip force and load force estimations. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp 1190–1195

[7] Controzzi M, Singh HK, Cini F, Cecchini T, Wing A, Cipriani C (2018) Humans adjust their grip force when passing an object according to the observed speed of the partner's reaching out movement. Experimental Brain Research 236:3363–3377

[8] Fujita M, Kato R, Tamio A (2010) Assessment of operators' mental strain induced by hand-over motion of industrial robot manipulator. In: 19th International Symposium in Robot and Human Interactive Communication, pp 361–366

[9] Huber M, Rickert M, Knoll A, Brandt T, Glasauer S (2008) Human-robot interaction in handing-over tasks. In: RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication, pp 107–112

[10] Kupcsik A, Hsu D, Lee WS (2018) Learning dynamic robot-to-human object handover from human feedback. In: Robotics research, Springer, pp 161–176

[11] Marturi N, Kopicki M, Rastegarpanah A, Rajasekaran V, Adjigble M, Stolkin R, Leonardis A, Bekiroglu Y (2019) Dynamic grasp and trajectory planning for moving objects. Autonomous Robots 43(5):1241–1256

[12] Medina JR, Duvallet F, Karnam M, Billard A (2016) A human-inspired controller for fluid human-robot handovers. In: 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), IEEE, pp 324–331
[13] Micelli V, Strabala K, Srinivasa S (2011) Perception and control challenges for effective human-robot handoffs. In: Proceedings of 2011 Robotics: Science and Systems, Workshop RGB-D
[14] Nemlekar H, Dutia D, Li Z (2019) Object transfer point estimation for fluent human-robot handovers. In: 2019 International Conference on Robotics and Automation (ICRA), IEEE, pp 2627–2633
[15] Ortenzi V, Cosgun A, Pardi T, Chan WP, Croft E, Kulić D (2021) Object handovers: A review for robotics. IEEE Transactions on Robotics pp 1–19
[16] Pan MK, Knoop E, Bächer M, Niemeyer G (2019) Fast handovers with a robot character: Small sensorimotor delays improve perceived qualities. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp 6735–6741
[17] Prada M, Remazeilles A, Koene A, Endo S (2014) Implementation and experimental validation of dynamic movement primitives for object handover. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2146–2153
[18] Rasch R, Wachsmuth S, König M (2019) Combining cartesian trajectories with joint constraints for human-like robot-human handover. In: 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), pp 91–98
[19] Sisbot EA, Marin-Urias LF, Broquere X, Sidobre D, Alami R (2010) Synthesizing robot motions adapted to human presence. International Journal of Social Robotics 2(3):329–343
[20] Solà J, Deray J, Atchuthan D (2018) A micro lie theory for state estimation in robotics. ArXiv abs/1812.01537
[21] Strabala K, Lee MK, Dragan A, Forlizzi J, Srinivasa SS, Cakmak M, Micelli V (2013) Toward seamless human-robot handovers. Journal of Human-Robot Interaction 2(1):112–132
[22] Vergara Perico CA (2020) Combining modeled and learned information to rapidly deploy human-robot collaborative tasks. PhD thesis, Arenberg Doctoral School, KU Leuven. Faculty of Engineering Science
[23] Widmann D, Karayiannidis Y (2018) Human motion prediction in human-robot handovers based on dynamic movement primitives. In: 2018 European Control Conference (ECC), pp 2781–2787
[24] Yang W, Paxton C, Mousavian A, Chao YW, Cakmak M, Fox D (2021) Reactive human-to-robot handovers of arbitrary objects. In: IEEE International Conference on Robotics and Automation (ICRA), IEEE