Applying the Popov-Vereshchagin Hybrid Dynamics Solver for Teleoperation under Instantaneous Constraints

Padmaja Kulkarni[1,2] Sven Schneider[1] Maren Bennewitz[2] Dirk Schulz[3] Paul Plöger[1]

***Abstract*— Teleoperation is still the de-facto mode of operation for robotic manipulators in hazardous and unknown environments. The objective is to move the manipulator under the influence of a plenitude of constraints, mainly following the human operator's commands, but also the avoidance of adverse effects such as joint limits or the exertion of external forces. A classic approach to incorporate such non-instantaneous behavior into the instantaneous motion of the kinematic chain is the Closed-Loop Inverse Kinematics (CLIK) control scheme.**

In this paper, we present PV-CLIK, a novel CLIK realization that for the *first time* practically applies the Popov-Vereshchagin (PV) hybrid dynamics solver to map the instantaneous constraints to motion commands. By relying on the PV solver, PV-CLIK offers several benefits over traditional CLIK implementations such as linear runtime complexity, handling constraints on the dynamics level or fostering composable software architectures. In the experimental evaluation, we show that our implementation of PV-CLIK outperforms existing kinematics solvers in Cartesian trajectory-following tasks at high velocities.

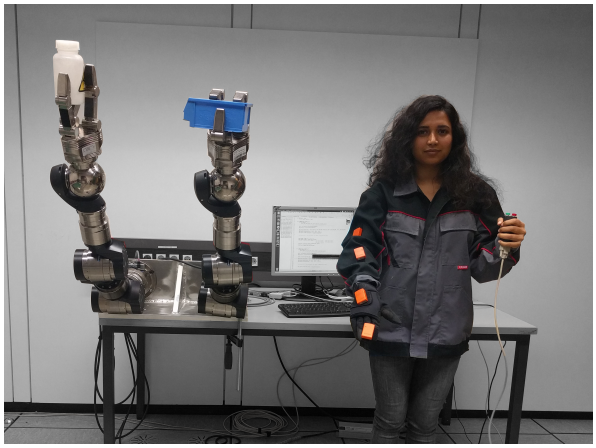


Fig. 1: Experimental setup for demonstrating an application of the PV solver for real-time human-robot motion mapping.

I. Introduction

Operating in potentially hazardous and unknown environments poses a challenge to human safety (e.g., a disaster-stricken area or a nuclear power plant). In such scenarios, teleoperation of robots is more likely to succeed than completely autonomous behavior due to the incorporation of human decision making.

Structural and physical differences between robots and humans make teleoperation with human-robot motion mapping challenging. Moreover, tasks involving coordinated motions of arms, e.g., transferring a box, need additional end effector constrains in terms of maintaining a constant distance between end effectors. Such tasks are difficult to perform using teleoperation, as without having the box in the hand, the human is very likely to fail in maintaining the constant distance between the hands. This demands a degree of autonomy on the robot's side and flexible control architecture, which allows seamless and instantaneous transfer between manual, shared, and autonomous control. Such an architecture must also allow *composability* and thereby warrant its ability to operate in different scenarios without requiring internal or semantic changes to the overall system [1].

The Popov-Vereshchagin hybrid dynamics solver offers a promising solution for computing human-robot motion mapping under these challenges [2], [3]. It provides an intuitive way to account for kinematic and dynamic constraints and allows to generate instantaneous motions that respect end-effector constraints. Despite these advantages, this solver has so far not been used for any practical applications like teleoperation. In this work, we build a teleoperation system based on the PV solver and integrate the necessary constraints, which respect kinematic joint-limits, while accounting for the Cartesian end-effector constraints.

In the experiments, we compare PV-CLIK with existing inverse kinematics solvers from the control perspective and evaluate it for coordinated dual-arm tasks. We show an application of the PV solver for teleoperation using a single robot arm and validate the solver's extensibility by performing bimanual teleoperation by transferring sand using the setup shown in Fig. 1.

The main contributions of this paper are (i) integration of joint-limit avoidance as a composable layer in the PV solver-based control scheme, and (ii) an application of the PV solver to real-world teleoperation.

The remainder of this paper is structured as follows: After discussing related work in the following section, we will explain the PV solver in Section III. Sections IV and V elaborate the application of the solver for human-robot motion mapping and integration of joint-limit avoidance respectively. In Section VI, we describe the hardware used for experimentation. Section VII presents experimental results demonstrating the advantages of our approach. Finally, we close the paper with conclusions and future work.

[1]Padmaja Kulkarni, Sven Schneider and Paul Ploeger are with the Department of Computer Science Hochschule Bonn-Rhein-Sieg, Germany. Corresponding author Email: padmaja.kulkarni@smail.inf.h-brs.de

[2]Maren Bennewitz and Padmaja Kulkarni are with the Humanoid Robots Lab, University of Bonn, Germany.

[3]Dirk Schulz is with Fraunhofer FKIE, Wachtberg, Germany.

II. Related work

Applications of teleoperation range from medical applications, for example, for minimally invasive surgeries like the Da Vinci surgical system, where *supervisory control* is practiced, to space applications like Eurobot [4] where *direct control* using exoskeletons is employed [5]. To keep the human in the loop, visual, force, or combined feedback is transferred to the human operator [6]. These methods require a controlled environment and state-of-the-art sensors and equipment, which are either costly, likely to introduce time delay, not easily transportable, or impede human arm movement [7] [8].

For cases where master-slave control or haptic feedback is not feasible, Stoyanov et al. develop a teleoperation method based on the Stack-Of-Tasks approach employing *shared control*, [9]. Some degree of autonomy is available at the human's disposal in this control paradigm like the alignment of the gripper, joint-limit, and obstacle avoidance. In contrast to the PV solver, their approach does not allow for the consideration of environmental forces. Khatib et al. [10] introduce a whole-body operational space control for motion avoiding near-body objects, joint-limits, and self-collisions. They control the robot posture by using a task-constrained Jacobian matrix in the force domain. They define the task as Cartesian accelerations and obtain the required joint torques to perform the task. Rodehutskors et al. design a bi-manual teleoperation system for the DARPA Robotics Challenge, which uses 3D visualization and motion tracking for teleoperation [11]. However, the control strategy cannot account for forces in the environment. Moreover, the teleoperation interface is complex and needs multiple users to operate the robot.

The Instantaneous Task Specification using Constraints (iTaSC) framework is well known for task-based control. It enables the generation of robot motions when constraints between the robot and its environment are known [12]. Focused on task specifications, this framework provides programming support for complex task execution. A task could be an interaction or motion among objects. Task specifications are obtained by imposing constraints on the modeled motion or interactions. Schutter et al. [12] demonstrate the system performance for tasks like contour tracking or human-robot comanipulation. However, the generic nature of this framework does not allow most efficient real-time control.

Based on [3], Shakhimardanov et al. proposed a control approach using instantaneous constraints [2]. This approach uses the PV solver to solve for constrained motion tasks. Unlike iTaSC [12], this approach is capable of performing efficient teleoperation, as the PV solver has linear time complexity. Due to the dynamic interface, it can account for forces acting on the robot end-effector and is potentially usable for compliant-motion based tasks, unlike in [9] and [11]. However, despite its capabilities, this solver has not been used or evaluated for online applications like teleoperation, to the best of our knowledge. In this paper, we present the first practical application of the PV solver for teleoperation.

III. Popov-Vereshchagin hybrid dynamics solver

Hybrid dynamics algorithms solve equations of motions for a kinematic chain, given partial joint motions or forces. This calculation requires: (i) computations of joint forces or torques using outward recursions of positions, velocities, and bias accelerations; (ii) inward recursions of inertia and forces; and (iii) final outward recursions of torques and accelerations. The Popov-Vereshchagin hybrid dynamics solver, additionally, introduces Cartesian acceleration constraints in their computational sweeps. Algorithm 1 shows the computational sweeps of the solver [2]. All the variables are in Plücker coordinates. This our case, it means that a variable is a six dimensional vector consisting respective Cartesian and angular components. The suffix i corresponds to the i^{th} joint or segment, 0 being the root and N being the leaf.

The explanation of the algorithm below assumes that the reader is familiar with the basic dynamics algorithms, e.g., the Articulated Body Hybrid Dynamics Algorithm [2]. Even though we briefly explain the solver, this is not the primary focus of this work. To realize the contribution of our work, the reader needs to understand *only* the inputs and the outputs of the PV solver given in Table I and its usages explained in Section IV-B. These variables form the inputs and the outputs of the PV-CLIK architecture are explained in Section IV-A.

In the *outward sweep*, the variables mentioned in the Table II are computed using outward recursions from the root to the leaf. Table III addresses the variables recursively computed from the leaf joint to the root joint, a.k.a. *inward sweep*. The main difference from the existing dynamics solvers are the terms *constraint force* A, *acceleration energy* U, and *coupling matrix* $\mathcal{L}$, which are computed in Algorithm 1, Lines 18, 19, and 21 respectively. Using the variables computed above, the Lagrange multiplier ν which minimizes the acceleration energy of the system is computed.

Output variables joint acceleration $\ddot{q}$ and Cartesian joint acceleration $\ddot{X}$ are calculated in the final outward sweep computations in Line 27 and Line 29, recursively from the root to the end-effector segment.

A more detailed explanation of this algorithm can be found in [2].

IV. Applying PV-CLIK to Motion Mapping

In this section, we will explain the control architecture used in our approach and the usages of the solver for human-robot motion mapping.

A. Control Architecture

As realized in [13] and [14], Closed-Loop Inverse Kinematics (CLIK) architecture can consider constraints instantaneously. Hence, we propose a CLIK scheme based on the PV solver, as shown in Fig. 2. The benefits of our approach originate from the application of the PV solver because it (i) can handle constraints on the dynamics level and, hence, is agnostic to the particular control scheme (e.g., position control or impedance control); (ii) is composable with controllers to handle a plenitude of tasks (e.g., end-effector tracking, joint-limit avoidance).

TABLE I: Solver input and output variables

Variables	Denotations
Joint angles	q
Joint velocities	$\dot{q}$
Feed-forward joint torque	τ
External force acting on a segment	F_{ext}
Unit constraint force matrix $[6 \times 6]$	A_N
End-effector Cartesian acceleration energy	b_N
Output torque required to perform motion	$\tau_{control}$
Joint accelerations	$\ddot{q}$
Cartesian acceleration of a segment	$\dot{X}$

TABLE II: Variables computed in the outward sweep of positions, velocities, and accelerations

Variables	Denotations
Cartesian pose of a segment	X
Cartesian velocity of a segment	$\dot{X}$
Initialization of the bias acceleration of a segment	$\ddot{X}_b$
Initialization of the bias force acting on a segment	F_b
Articulated inertia of the segment	I^A

TABLE III: Variables computed in the inward sweep of the solver

Variables	Denotations
Articulated body inertia in joint subspace	D
Projection operator for the inertia	P^A
Apparent inertia	I
Bias force including articulated bias force	F_b^a
Constraint force	A
The acceleration energy produced due to the contribution of child segment, bias forces, and feed-forward torques	U
Coupling matrix whose rows represent acceleration energy generated at the segment	$\mathcal{L}$

An input to the architecture can be the desired acceleration of the end-effector, desired feed-forward joint-torques, and even external forces acting on the end effector. The output is joint-level accelerations and torques. In our application, we generate the velocity commands by integrating obtained accelerations.

B. Motion Mapping

We use the PV solver to compute human-robot motion mapping for teleoperation of a robot. We assume that a motion capture system computes the pose (position and orientation) of a human hand-tip with respect to the shoulder. Using a simple scaling and transformation to match axes, we compute the desired pose of robot end-effector in its base frame. We obtain the current robot end-effector pose using forward kinematics. *At this step, both the desired and current robot end-effector positions and orientations are known*. With a given desired end-effector pose for the next time step, we compute joint accelerations using PV solver.

First, we calculate the instantaneous Cartesian velocity based on the difference between current and desired robot end-effector positions,

$$\begin{aligned} v &= (\frac{x_{des} - x_{curr}}{\delta t}, \frac{y_{des} - y_{curr}}{\delta t}, \frac{z_{des} - z_{curr}}{\delta t}) \\ &= (v_x, v_y, v_z). \end{aligned} \quad (1)$$

Algorithm 1: Constrained Hybrid Dynamics Solver

Input : $\mathbf{Robot\ model}$, $\mathbf{q}$, $\dot{\mathbf{q}}$, τ, $\ddot{\mathbf{X}}_{\mathbf{0}}$, $\mathbf{F_{ext}}$, $\mathbf{A_N}$, $\mathbf{b_N}$
Output: $\tau_{\mathbf{control}}$, $\ddot{\mathbf{q}}$, $\ddot{\mathbf{X}}$

begin
  // Outward sweep
  for i = 0 to N − 1 do
    ${}_{i}^{i+1}X = ({}^{d_i}X_i \ \ {}^{i+1}X_{d_i}(q_i))$;
    $\dot{X}_{i+1} = {}^{i+1}X_i\dot{X}_i + S_{i+1}\dot{q}_{i+1}$;
    $\ddot{X}_{b,i+1} = \dot{X}_{i+1} \times S_{i+1}\dot{q}_{i+1}$;
    $F^A_{b,i+1} = \dot{X}_{i+1} \times I_{i+1}\dot{X}_{i+1} - F^{ext}_{i+1}$;
    $I^A_{i+1} = I_{i+1}$;
  end
  // Inward sweep of inertia, force contributions
  for i = N − 1 to 0 do
    $D_{i+1} = d_{i+1} + S^T_{i+1}I^A_{i+1}S_{i+1}$;
    $P^A_{i+1} = 1 - I^A_{i+1}S_{i+1}D^{-1}_{i+1}S^T_{i+1}$;
    $I_i = P^A_{i+1}I^A_{i+1} + \sum {}^{i}X^T_{i+1}I^a_{i+1}{}^{i}X_{i+1}$;
    $F^A_{b,i+1} = P^A_{i+1}F^A_{b,i+1} + I_{i+1}S_{i+1}D^{-1}_{i+1}\tau_{i+1}$
      $+I_{i+1}\ddot{X}_{b,i+1}$;
    $F^a_{b,i} = F^A_{b,i} + \sum {}^{i}X^*_{i+1}F^a_{b,i+1}$;
    $A_i = {}^{i}X^T_{i+1}P^A_{i+1}A_{i+1}$;
    $U_i = U_{i+1} + A^T_{i+1}\{\ddot{X}_{b,i+1} + S_iD^{-1}_i$
    $(\tau_{i+1} - S^T_i(F^A_{b,i+1} + I_{i+1}\ddot{X}_{b,i+1}))\}$;
    $\mathcal{L}_i = \mathcal{L}_{i+1} - A^T_{i+1}S_{i+1}D^{-1}_{i+1}S^T_{i+1}A_{i+1}$;
  end
  // Balance of acceleration energy at the base link
  $\nu = \mathcal{L}^{-1}_0(b_N - A^T_0\ddot{X}_0 - U_0)$;
  // Outward sweep of accelerations
  for i = 0 to N − 1 do
    $\ddot{q}_{i+1} = D^{-1}_{i+1}\{\tau_{i+1} - S^T_{i+1}(F^A_{b,i+1} +$
      $I_{i+1}({}^{i+1}X_i\ddot{X}_i + \ddot{X}_{b,i+1}) + A_{i+1}\nu)\}$;
    $\ddot{X}_{i+1} = {}^{i+1}X_i\ddot{X}_i + S_{i+1}\ddot{q}_{i+1} + \ddot{X}_{bias,i+1}$;
  end
end

Multiplication with the rate of change of the rotation matrix and the inverse of the current rotation matrix gives a skew-symmetric matrix comprising the angular velocity components,

$$W = \dot{R}\,R^{-1}_{cur}, \ where\ W = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \quad (2)$$

Using W, the end-effector angular velocity is given as

$$\omega = (\omega_x, \omega_y, \omega_z). \quad (3)$$

Linear and angular end-effector accelerations are chosen to be proportional to the respective velocities:

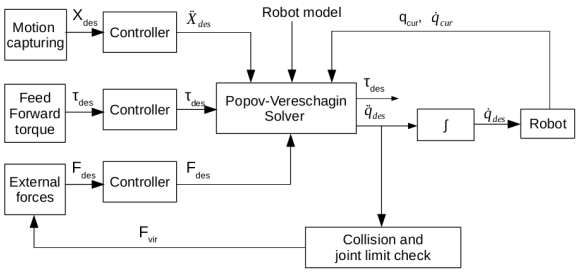


Fig. 2: Control architecture for PV-CLIK with position level input constraints and velocity level output commands to the robot. Acceleration and torque could be used to command the robot depending upon the available robot interface. The variables subscribed using *des* and *cur* are the desired and current variables respectively.

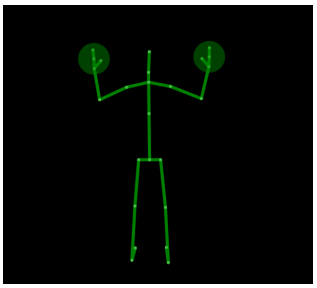


Fig. 3: Skeleton tracking using the Kinect-based motion capturing system. Here, only the position of the joints can be obtained.

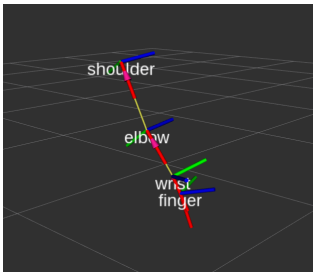


Fig. 4: Arm pose obtained using the IMU-based motion capturing system. Here, position and orientation of the arm-joints can be obtained.

$$\ddot{X}_{input} = \begin{bmatrix} k_1 v & k_2 \omega \end{bmatrix}. \tag{4}$$

Here, k_1 and k_2 are proportionality constants, selected by initial offline trials with the robot.

This acceleration is equivalent to the input b_N to the PV solver used in Algorithm 1, Line 24. Since we specify motion in all six DOFs, the unit constraint forces A_N is a 6×6 unit diagonal matrix. Feed-forward torque τ are considered to be zero.

V. Joint Limit Avoidance Controller

For the robot to avoid joint-limits, a virtual external force is applied such that it rotates the joints away from the joint-limits. A virtual external force proportional to the Cartesian velocity is applied on the end-effector to achieve the desired end effector motion, while avoiding joint-limits.

Since, we use external force interface, the constraint matrix for A_N is a 6×6 null matrix and

$$\ddot{X}_{input} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5}$$

The end effector constraints are imposed using the virtual external force given as

$$F_N^{ext} = \begin{bmatrix} c_1 v & c_2 \omega \end{bmatrix}^T. \tag{6}$$

c_1 and c_2 are proportionality constants, selected by initial offline trials with the robot, as the inertial model of our robot is not known. In our case, all the joints are revolute joints rotating about the z-axis. Hence, the applied force is the dot product of the force vector and the motion subspace matrix. They are used in Algorithm 1, Line 7 and calculated as

$$\begin{aligned} F_i^{ext} &= S_i \cdot F, \\ S_i &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \\ F &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & n_z \end{bmatrix}^T. \end{aligned} \tag{7}$$

The value of n_z is determined in initial trials on the robot. Using this external force as solver input, we avoid joint-limits.

VI. Experimental Setup

In this section, we will describe our motion capturing setup and robotic platform.

A. Motion Capturing System

This paper uses a combined IMU and Kinect-based[1] motion capturing approach. The Kinect tracks full body motion as shown in Fig. 3. We only use joints positions of shoulder, elbow, wrist, and hand-tip. The Kinect cannot capture the orientations of the hand-tip, and hence, we use an additional IMU-based tracking system described in [15]. This system uses four Xsens MTw Awinda wireless motion trackers[2] on the operator's humerus, radius, hand, and finger and creates a model of the human arm as shown in Fig. 4. The IMUs are mounted on a jacket and are visible in Fig. 1. In this work, we only use the information about the position and orientation of the fingers or the hand-tip with respect to the shoulder.

B. Robotic Hardware

We use two Schunk LWA-4D arms[3] for evaluation. The arm has seven DoF and seven revolute joints. The arms are equipped with two Schunk Dexterous hands[4] as shown in Fig. 1. Velocity and position interfaces are available for commanding a motion. We use the velocity interface to achieve more flexible instantaneous control over the trajectory, with the control frequency of 30 Hz.

VII. Experimental Evaluation

This paper evaluates the PV-CLIK with three different types of experiments including, (i) imitation of predefined motions with a single arm, (ii) teleoperation with a single arm, and (iii) coordinated dual-arm motion. Additionally, we

[1] https://msdn.microsoft.com/en-us/library/dn782025.aspx (03/09/2017)

[2] https://www.xsens.com/products/mtw-awinda/(03/09/2017)

[3] http://www.schunk-modular-robotics.com/en/home/products/dextrous-lightweight-arm-lwa-4d.html (24/10/2018)

[4] https://schunk.com/de_en/gripping-systems/series/sdh/ (24/10/2018)

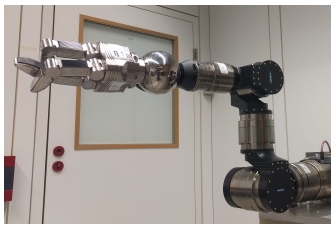


Fig. 5: Setup for imitation of predefined end-effector motions with a single arm.

show the easy extensibility of the proposed architecture by demonstrating its usages for a teleoperation task with a dual-arm robot.

The first experiment evaluates the performance of our control system against predefined motions and compares this performance with the widely used TRAC-IK[5] solver. We chose the TRAC-IK solver as it uses algorithms based on Newton's convergence methods and Sequential Quadratic Programming simultaneously to improve the success rate. It claims a solution rate of above 99% for Schunk LWA-4D arms, which is better than any individual kinematics solver.

The second experiment evaluates the intuitiveness of the proposed system. The third experiment evaluates the ability of the system to perform a coordinated task which requires precision (e.g., carrying a box with two arms), with partial autonomy at the robot's end. The final experiment demonstrates dual-arm teleoperation using the PV-CLIK.

A. *Imitation of Predefined Motions with a Single Arm*

The deviation between desired and actual motion is measured using the Fréchet distance, *which is the maximum distance by which a given curve deviates from the original curve*.

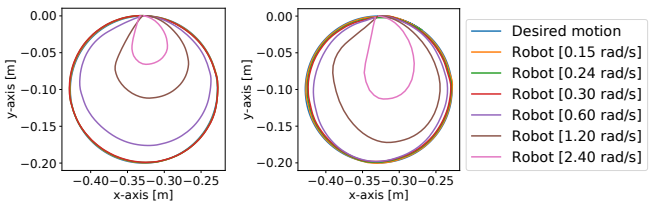
1) Setup: In these experiments, we generated circular and square trajectories. We sent the position generated to the solver at 30 Hz. In the robot arm base link, the x axis is pointing downwards, and the z axis is the pointing forward. The direction of the y-axis follows the right hand rule. The effect of a variation of the speed on the end-effector motion is analyzed. Fig. 5 shows the setup.

2) Experiments and Results: First, we present results for desired motion generation at variable speeds. The circle radius and the square sides are 0.1 m with the same end-effector orientation. The solver attempts to reach the *latest* commanded position. As a consequence, if the velocity is varied, the robot end-effector trajectory varies. Fig. 6 and 7 show the variation of the trajectory as the speed of the commanded motion changes using both the TRAC-IK and PV-CLIK as solution methods.

z

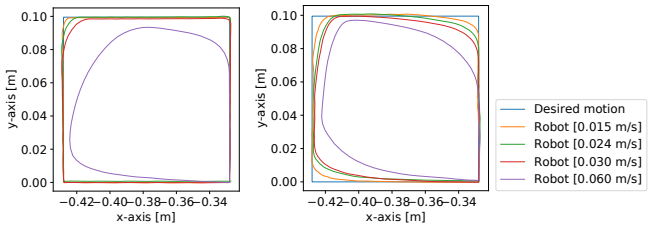
Fig. 8 and 9 show box plots for such deviations at different speeds and Table IV enumerates the results. The average latencies of the PV-CLIK and TRAC-IK solver were 0.22 ms and 0.50 ms respectively.

[5]https://bitbucket.org/traclabs/trac_ik.git (10/13/2018)



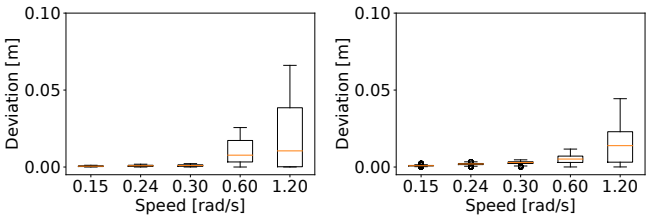
(a) TRAC-IK solver (b) PV-CLIK.

Fig. 6: Robot end-effector motion response to circular motion generated at different speeds. As can be seen, at higher speeds, the PV-CLIK performs better trajectory tracking than the TRAC-IK solver.



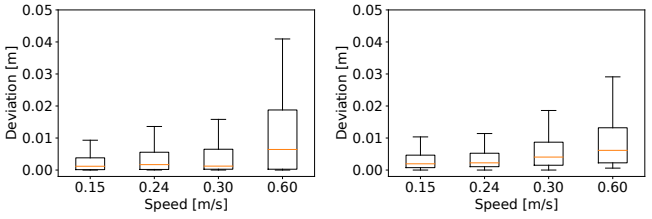
(a) TRAC-IK solver. (b) PV-CLIK.

Fig. 7: Robot end-effector motion response to square motion generated at different speeds. As can be seen, at higher speeds, the PV-CLIK performs better trajectory tracking than the *TRAC-IK* solver.



(a) TRAC-IK solver. (b) PV-CLIK.

Fig. 8: Deviation of the robot end-effector from the desired *circular* motion. As can be seen, at higher speeds, the TRAC-IK solver deviates more from the desired trajectory than the PV-CLIK.



(a) TRAC-IK solver. (b) PV-CLIK.

Fig. 9: Deviation of the robot end-effector from the desired *square* motion. As can be seen, at higher speeds, the TRAC-IK solver deviates more from the desired trajectory than the PV-CLIK.

TABLE IV: Experimental results for trajectory following with a single-arm. At the highs speeds, measuring the deviation from the desired trajectory, the PV-CLIK performed better than that of the TRAC-IK solver.

Shape	Speed	Max Deviation using the TRAC-IK solver [m]	Max Deviation using the PV solver [m]
Circle	0.15 rad/s	0.001	0.002
	1.20 rad/s	**0.089**	**0.056**
Square	0.15 rad/s	0.0018	0.0039
	0.6 m/s	**0.041**	**0.030**

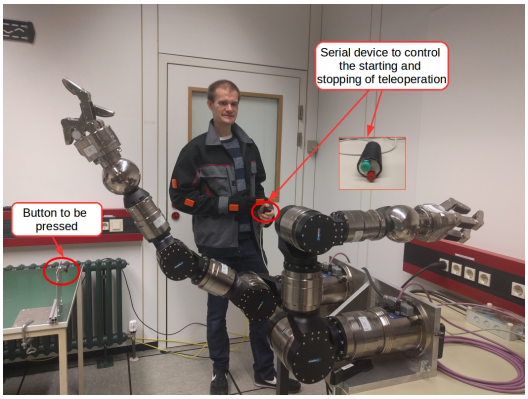


Fig. 10: Experimental setup for the task of pressing a button using teleoperation. It shows the relative position of user, robot arm and button.

In summary, the PV-CLIK outperforms the TRAC-IK solver in terms of latency and end-effector pose tracking. The tracking error for both solvers increases as the speed of the desired motion increases. However at the high speeds, the PV-CLIK performed better than the best-performing IK solver. At low speeds, its performance was comparable to that of the TRAC-IK solver.

B. Teleoperation with a Single Arm

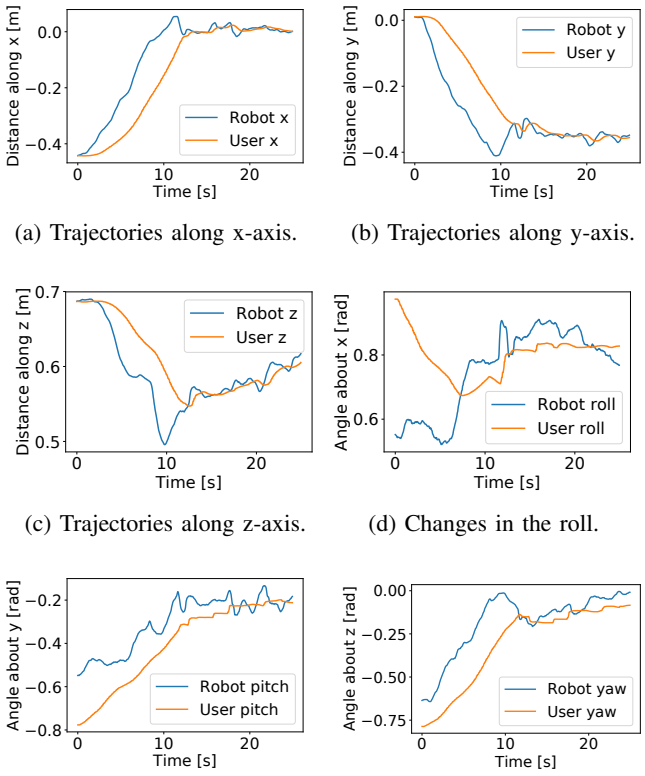
In this section, we evaluated the teleoperation system for the task of pressing a button using *only* the PV-CLIK.

1) Setup: The participants wore a jacket with IMUs mounted on it. They were given a serial device to start and stop the manipulator control and to reset the starting position. This button ensures that the users can relax their arm, if necessary. We mounted the button to be pressed using teleoperation on the table in the reachable workspace of the robot. The experimental setup is as shown in Fig. 10.

2) Experiments and Results: Ten participants were asked to press the black button shown in Fig. 10, while teleoperating a single arm. These participants were in the age of 15-35 years old and naïve users of the system. We gave these users an instruction explaining the working of the system and the task. They were given 300 seconds to train and 300 seconds to accomplish the task. We noted the trajectory of both - the robot and the user, the time needed, the number of restarts, and whether they could accomplish the task. The robot's end-effector and the user's hand positions are calibrated to be aligned at the start of teleoperation, but not the orientations.

TABLE V: Evaluation with users for the task of pressing a button with teleoperation.

User	Task completion	Time needed [s]	Number of restarts
1	Yes	53	0
2	Yes	104	0
3	Yes	56	0
4	Yes	48	1
5	Yes	300	1
6	Yes	30	0
7	Yes	28	0
8	Yes	54	0
9	Yes	75	0
10	Yes	104	0



(a) Trajectories along x-axis. (b) Trajectories along y-axis.

(c) Trajectories along z-axis. (d) Changes in the roll.

(e) Changes in the pitch. (f) Changes in the yaw.

Fig. 11: Trajectory of the end-effector and the user's hand while performing the task of pressing a button with single arm teleoperation. As can be seen, the robot-end effector is able to follow the user's trajectory.

We use absolute orientations, as it is counterintuitive to perform relative orientations due to the symmetric gripper.

Table V shows the task performance of the participants. All participants learned to use the system in the given 300 seconds. Every participant was able to perform the task successfully. In Fig. 11, we see that the robot end-effector was able to follow the user's position and orientation. The system had a latency of 20-24 milliseconds.

We observed that the users inadvertently tend to change the orientation of their hand, while moving their arm, or while resetting the robot's position using the red button. To avoid this inadvertent orientation change for the robot, the angle tolerance is kept large, i.e., 0.1 radians. Consequently,

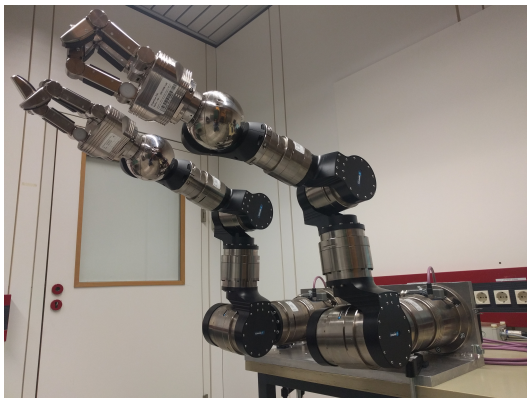


Fig. 12: Experimental setup for evaluating coordinated dual-arm motion.

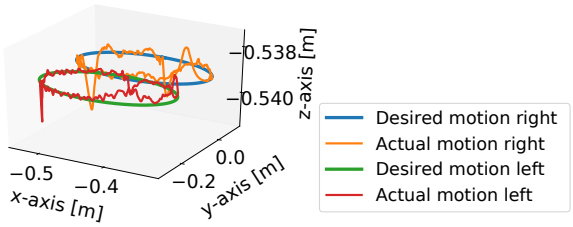


Fig. 13: End-effector motion response of the right and the left arm to the circular motion generated at a speed of 0.15 rad/s.

Figs. 11d, 11e, 11f show the robot followed the human roll, pitch, and yaw only up to the tolerance of 0.1 radians.

C. Coordinated Dual-Arm Motion

This section involves testing and experimentation for a coordinated dual-arm task using *only* the PV-CLIK. This is important when a robot wants to perform coordinated tasks, e.g., carrying a box, where the distance between two end-effectors should remain constant, while the arms are moving.

With a dual-arm setup, we imposed the task constraint that the distance between two arms must be constant while moving towards the desired pose. A trajectory of only one arm is known to the robot. The second arm trajectory is computed based on the constant relative distance between two arms. In this scenario, the second arm moves autonomously to follow the first arm, giving the robot control of the tasks, when precision in maintaining a fixed distance is needed.

1) Setup: The experimental setup is as shown in Fig. 12. The two arms are in the starting position such that the commanded trajectory is in the reachable workspace of the robot.

2) Experiments and Result: We evaluated the performance of the solver while commanding circular and square motions at various speeds. For coordinated motions, the critical constraint is that the Euclidean distance between the end-effectors along every axis is constant. The left arm in Fig. 12 was in the trajectory following mode, and the right arm maintained the distance between two arms at any given instance.

Fig. 13 shows the trajectory of the end-effector of the arm for this constraint for a circle of 0.1 m radius and at a speed of 0.15 rad/s. Fig. 14 shows the trajectory for square of 0.1 m side and at a speed of 0.15 m/s.

Fig. 15 shows the distance between the two arms as a function of time and Fig. 16, the box plot for this experiment. Figs. 17 and 18 show the respective pots for the square trajectory.

3) Discussion: The trajectories are commanded in such a way that both the arms have to travel different distances in the joint space to be able to follow the trajectory. This insures that the solvers for two arms work independently.

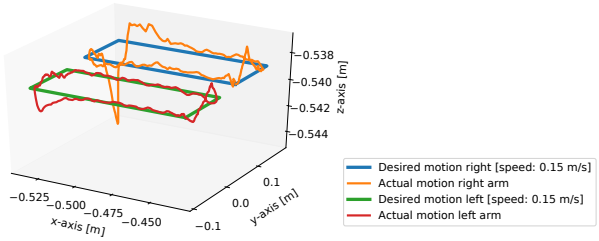


Fig. 14: End-effector motion response of the right and the left arm to the square motion generated at a speed of 0.15 rad/s.

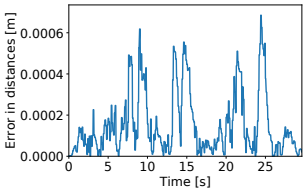


Fig. 15: Deviations in the distance between two end-effectors for a speed of 0.15 rad/s from the initial distance, for *circular* motion.

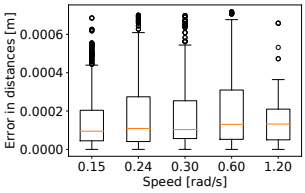


Fig. 16: Deviations in the distance between two end-effectors from the initial distance, for *circular* motion at various speeds.

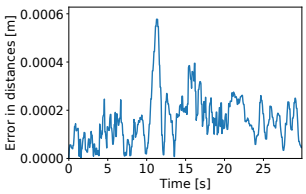


Fig. 17: Deviations in the distance between two end-effectors for a speed of 0.15 m/s from the initial distance, for *square* motion.

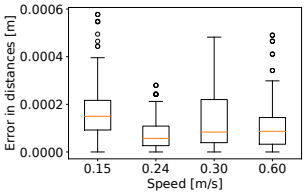


Fig. 18: Deviations in the distance between two end-effectors from the initial distance, for *square* motion at various speeds.

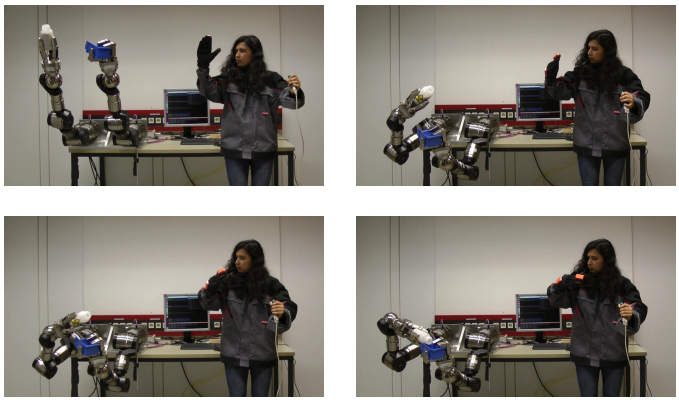


Fig. 19: Dual-arm teleoperation task of pouring sand from one hand to the other hand

The Fréchet distance for the dual-arm setup for the desired circular motion at the speed of 0.15 rad/s was observed to be 0.0012 m and 0.0021 m for the right and the left arm respectively. For the square motion, they were 0.0038 m and 0.0042 m for the right and the left arm respectively.

The priority of the task is to maintain the distance between the arms. These experiments show that the solver could maintain the distance between the two arms' end-effectors within a sub-millimeter range, while still being able to maintain the trajectory within 0.0042 m accuracy. The maximum deviation between the end-effectors was 0.0008 m and it is independent of the speed or the trajectory shape.

D. Extensibility of the Approach: Teleoperation with a Dual-Arm Robot

Finally, to show the easy extensibility of our control architecture, we demonstrate the performance of our system for the task of teleoperation of a dual-arm robot. The task is to pour sand from one hand to the other as shown in Fig. 19. Both Xsens and the Kinect camera track the motion of the user. Self-collision checking is performed using the Flexible Collision Checking (FCL) library[6]. The motion-mapping controller stops the manipulators and informs the user if a collision is imminent in the intended motion direction. The video of this experiment is available online[7].

VIII. Conclusion

This paper demonstrates the first application of the Popov-Vereshchagin hybrid dynamics solver for human-robot motion mapping under instantaneous motion constraints. To realize this, we propose the PV-CLIK scheme and incorporate joint-limit avoidance as a composable layer in the same. In various experimental evaluations, we show that our control methodology using the PV solver: (i) performs better than existing kinematics solvers at higher speeds in terms of following trajectories, (ii) leads to an intuitive control interface for teleoperation, (iii) allows for partial autonomy of the robot for coordinated tasks, and (iv) is capable of performing bi-manual teleoperation.

Future work involves testing system performance under the influence of external forces, including compliant motion following, and providing a solution for singularity avoidance.

[6]https://github.com/flexible-collision-library/fcl(24/09/2018)

[7]https://www.youtube.com/watch?v=M3TondB7tlQ

References

[1] S. Schneider and H. Bruyninckx, "Exploiting linearity in dynamics solvers for the design of composable robotic manipulation architectures," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, to appear.

[2] A. Shakhimardanov, "Composable robot motion stack: Implementing constrained hybrid dynamics using semantic models of kinematic chains," Tech. Rep., PhD thesis, KU Leuven, 2015.

[3] A. F. Vereshchagin, "Modelling and control of motion of manipulation robots," in *Soviet Journal of Computer and Systems Sciences*, vol. 27, 1989, p. 2938.

[4] A. Schiele, M. D. Bartolomei, and F. V. D. Helm, "Towards intuitive control of space robots: A ground development facility with exoskeleton," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 1396–1401.

[5] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[6] K. D. Katyal, C. Y. Brown, S. A. Hechtman, M. P. Para, T. G. McGee, K. C. Wolfe, R. J. Murphy, M. D. M. Kutzer, E. W. Tunstel, M. P. McLoughlin, and M. S. Johannes, "Approaches to robotic teleoperation in a disaster scenario: From supervised autonomy to direct control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[7] D. Cannon and M. Siegel, "Perceived mental workload and operator performance of dexterous manipulators under time delay with master-slave interfaces," in *IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, Jun 2015, pp. 1–6.

[8] J. Rebelo, T. Sednaoui, E. B. den Exter, T. Krueger, and A. Schiele, "Bilateral robot teleoperation: A wearable arm exoskeleton featuring an intuitive user interface," *IEEE Robotics Automation Magazine*, vol. 21, no. 4, pp. 62–69, Dec 2014.

[9] T. Stoyanov, R. Krug, A. Kiselev, D. Sun, and A. Loutfi, "Assisted telemanipulation: A stack-of-tasks approach to remote manipulator control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–9.

[10] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole-body dynamic behavior and control of human-like robots." vol. 10, pp. 29–43, Mar. 2004.

[11] T. Rodehutskors, M. Schwarz, and S. Behnke, "Intuitive bimanual telemanipulation under communication restrictions by immersive 3d visualization and motion tracking," in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov. 2015, pp. 276–283.

[12] J. D. Schutter, T. D. Laet, J. Rutgeerts, W. Decr'e, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *Int. J. Rob. Res.*, vol. 26, no. 5, pp. 433–455, May 2007.

[13] A. Balestrino, G. D. Maria, and L. Sciavicco, "Robust control of robotic manipulators," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 2435 – 2440, 1984, 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667017613478

[14] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 403–410, Aug 1988.

[15] J. Hoffmann, B. Brüggemann, and B. Krüger, "Automatic calibration of a motion capture system based on inertial sensors for telemanipulation," in *7th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Jun. 2010.