Supplementary Material

Prescient teleoperation of humanoid robots

Submitted to IEEE HUMANOIDS 2023

Luigi Penco[1], Jean-Baptiste Mouret[2] and Serena Ivaldi[2]

[1]Luigi Penco is with the Florida Institute for Human and Machine Cognition. lpenco@ihmc.org

[2]Jean-Baptiste Mouret and Serena Ivaldi are with Inria Nancy — Grand Est, CNRS and Université of Lorraine.
{jean-baptiste.mouret,serena.ivaldi}@inria.fr

Overall, our *prescient* whole-body teleoperation system relies on the following components (Fig. 2):

- a whole-body controller based on quadratic optimization;
- a dataset of whole-body trajectories retargeted from human motions;
- a set of ProMPs that can predict future trajectories given observations;
- a computation of the round-trip delay to select the appropriate commands from the prediction, so that the robot anticipates both the operator-to-robot and the robot-to-operator delays;
- a blending to keep the trajectory smooth, at the start of a trajectory or in case of changes of delays;
- a video streaming system that uses a jitter buffer to cope with the stochastic part of the backward delay.

APPENDIX I
HUMANOID ROBOT AND WHOLE-BODY CONTROLLER

A. The iCub humanoid robot

iCub [34] is a research-grade open-source humanoid robot designed by the Italian Institute of Technology (IIT) to experiment with embodied artificial intelligence. It measures 104 cm in height and weighs 22 kg, which roughly corresponds to the body dimensions of a five-year-old child. iCub has 53 actuated degrees of freedom: 7 in each arm, 9 in each hand (3 for the thumb, 2 for the index, 2 for the middle finger, 1 for the coupled ring and little finger, 1 for the adduction/abduction), 6 in the head (3 for the neck and 3 for the cameras), 3 in the torso/waist, 6 in each leg. In this work, we do not use the fingers of the hands and we do not move the eyes, therefore we control 32 degrees of freedom. The head has stereo cameras in a swivel mounting in the corresponding location of the human eye sockets. iCub is also equipped with six 6-axial force/torque (F/T) sensors in the upper arms, legs and feet, an IMU in the head, and a distributed tactile skin.

B. Whole-body controller

The whole-body motion of the robot is defined by the following trajectories: center of mass ground projection, waist height, hands positions, arms postures, neck posture, torso posture, which are either given by the delayed retargeted human motion, or generated by the delay compensation algorithm during the execution of the main task. Each sample of each of these trajectories represents a control reference $\hat{\boldsymbol{y}}$. Given $\hat{\boldsymbol{y}}$, the robot commands $\boldsymbol{q}$ are generated by solving the redundant inverse kinematics, which can be formulated as a constrained quadratic programming problem with equality and inequality constraints [2], [42]:

$$\begin{aligned} \arg\min_{\dot{\boldsymbol{q}}} & \textstyle\sum_i w_i f_i + \sum_j w_j g_j + \boldsymbol{C}\dot{\boldsymbol{q}} \\ f_i &= ||\boldsymbol{J}_i\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}_i||^2 \\ g_j &= ||\dot{\boldsymbol{q}}_j - \dot{\boldsymbol{q}}_j^r||^2 \\ \text{subject to } & \boldsymbol{J}\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}} \\ & \boldsymbol{A}\dot{\boldsymbol{q}} \leq \boldsymbol{b} \end{aligned} \tag{S1}$$

The cost function consists of terms f_i with relative weight w_i concerning the pose of a specific body link i, where $\boldsymbol{J}_i$ is the Jacobian matrix for body link i and $\dot{\boldsymbol{x}}_i = \dot{\hat{\boldsymbol{y}}}_i$ are the reference velocities for body link i. Additionally terms g_j with relative weight w_j concern the posture of certain joints j, where $\dot{\boldsymbol{q}}_j^r = \dot{\hat{\boldsymbol{y}}}_j$ are the reference joint velocities for joints j. $\boldsymbol{C}\dot{\boldsymbol{q}}$ is a regularization term used to get a unique solution and avoid singularities, where $\boldsymbol{C}$ is a linear cost vector.

In our implementation, we considered in the terms f_i the hand positions with $w_i = 1$ and the waist height with $w_i = 0.65$. Instead, the terms g_j include the head posture with $w_j = 1$ and the torso posture with $w_j = 0.72$, the elbow and wrist postures with $w_j = 0.11$. We computed optimal priorities with a multi-objective stochastic optimization that was run in simulation [2]. More details about the whole-body controller can be found in [2].

The equality constraints correspond to the highest priority task, which should be solved exactly. In our implementation, these include the center of mass x position and the feet poses. The inequality constraints contain the robot joint velocity bounds and zero moment point bounds, which is constrained to stay inside the support polygon.

Our controller is based on the OpenSOT framework [35] and the qpOASES quadratic programming solver [45].

This controller is run at 100 Hz, which is also the frequency of the motor commands.

C. Motion retargeting

The captured human information cannot be directly used as a reference for the humanoid, due to differences in kinematics (e.g. joint limits, body dimensions) and dynamics (e.g. mass distribution) between the human and the robot.

TABLE S1

PREDICTION ERROR AFTER OBSERVING DIFFERENT PORTIONS OF THE COMMANDED TRAJECTORIES (DATASET MULTIPLE TASKS). WE EVALUATED THE DIFFERENCE BETWEEN THE ACTUAL TRAJECTORY (COMMANDS RETARGETED FROM THE OPERATOR) AND THE PREDICTED TRAJECTORY FOR THE 20 TESTING MOTIONS FROM THE BOTTLE REACHING SCENARIO (FIG. S1) AND THE 21 TESTING MOTIONS FROM THE BOX HANDLING SCENARIO (FIG. S2). TO UNDERSTAND THE INFLUENCE OF THE CONDITIONING OF THE PROMPS, WE COMPUTED THE MEAN ERROR BY FOLLOWING THE MEAN OF THE PROMP SELECTED BY HAND ('NO OBS'), AFTER THE INITIAL RECOGNITION ('RECOGNITION') THAT TAKES ABOUT 1S, AFTER A FOURTH OF THE MOTION (1/4 MOTION) AND AFTER HALF OF THE MOTION (1/2 MOTION). THANKS TO THE CONDITIONING, WHEN MORE DATA IS USED, THE PREDICTION IS MORE ACCURATE, WHICH MEANS THAT THE PREDICTION IS ADJUSTED TO SUIT THE PARTICULAR MOTIONS OF THE OPERATOR (THAT IS, THE ROBOT DOES NOT SIMPLY FOLLOW THE MEAN TRAJECTORY ONCE IT HAS RECOGNIZED IT). EXAMPLES OF PREDICTED TRAJECTORIES ARE DISPLAYED IN FIG. S1 AND FIG. S2.

Trajectory	Box handling **RMS error [rad]**				Bottle reaching **RMS error [rad]**			
	no obs	recognition	1/4 motion	1/2 motion	no obs	recognition	1/4 motion	1/2 motion
head yaw	0.112±0.049	0.080±0.028	0.045±0.012	0.012±0.009	0.083±0.038	0.040±0.015	0.023±0.011	0.010±0.008
torso pitch	0.155±0.064	0.120±0.046	0.054±0.030	0.018±0.008	0.103±0.056	0.061±0.022	0.044±0.015	0.019±0.008
torso roll	0.119±0.049	0.103±0.038	0.055±0.028	0.017±0.008	0.082±0.042	0.054±0.016	0.032±0.010	0.016±0.008
torso yaw	0.168±0.053	0.136±0.049	0.079±0.032	0.049±0.019	0.088±0.046	0.065±0.039	0.049±0.023	0.033±0.018
r. should. yaw	0.164±0.059	0.109±0.050	0.053±0.019	0.040±0.011	0.172±0.059	0.114±0.056	0.055±0.024	0.027±0.011
r. elbow	0.169±0.072	0.146±0.038	0.097±0.032	0.033±0.010	0.220±0.083	0.097±0.032	0.065±0.017	0.034±0.011
r. wrist pros.	0.113±0.049	0.092±0.022	0.043±0.012	0.026±0.008	0.124±0.052	0.071±0.022	0.048±0.016	0.021±0.008
	RMS error [cm]				**RMS error [cm]**			
	no obs	recognition	1/4 motion	1/2 motion	no obs	recognition	1/4 motion	1/2 motion
r. hand x	3.76±0.91	2.10±0.48	1.57±0.49	0.78±0.12	3.54.±0.88	1.62±0.65	0.61±0.33	0.48±0.14
r. hand y	3.55±0.93	1.91±0.40	0.97±0.32	0.52±0.10	3.71±0.89	2.02±0.48	0.89±0.19	0.35±0.11
r. hand z	2.98±0.91	2.34±0.83	1.22±0.27	0.66±0.19	2.71±0.80	1.99±0.79	0.77±0.29	0.50±0.13
com x	2.40±0.76	0.98±0.26	0.57±0.21	0.23±0.14	0.78±0.53	0.39±0.22	0.29±0.12	0.12±0.10
com y	1.12±0.55	0.58±0.26	0.29±0.18	0.20±0.13	0.80±0.44	0.52±0.29	0.29±0.12	0.11±0.10
waist z	3.53±1.04	2.74±0.93	1.68±0.57	0.65±0.22	0.95±0.36	0.65±0.22	0.39±0.19	0.28±0.14

Hence, motion retargeting is employed to map the human information into feasible reference trajectories for the robot. For transferring the translational movements of the end-effectors we used a fixed scaling factor (0.4). For transferring postures, the joint angles of the human joints are manually identified and mapped to the corresponding joints of the robot [1]. The instantaneous reference value of the robot is then computed as:

$$\Delta q_{i_R} = q_{0_R} + (q_{i_H} - q_{0_H}) \tag{S2}$$

where q is the vector of current joint positions, Δq is the vector of joint variations with respect to the initial posture, the indices 0 and i refer to measurements at an initial time and at time i, and the subindices H and R indicate measurements on human and robot, respectively. The same applies to the Cartesian positions of the end-effectors.

For the center of mass, the normalized offset-based reconstruction is used [1]. We consider the ground projection of the human center of mass $\boldsymbol{p}^g_{com}$. Its position with respect to the left foot is projected onto the line connecting the two feet. The result is then normalized to obtain an offset $o \in [0, 1]$

$$o = \frac{(\boldsymbol{p}^g_{com} - \boldsymbol{p}^g_{lFoot}) \cdot (\boldsymbol{p}^g_{rFoot} - \boldsymbol{p}^g_{lFoot})}{|| \, \boldsymbol{p}^g_{rFoot} - \boldsymbol{p}^g_{lFoot} \, ||^2}$$

where $\boldsymbol{p}^g_{lFoot}$ and $\boldsymbol{p}^g_{rFoot}$ are the ground projections of the left and right foot respectively. When the human is in a symmetric pose, the offset o has a value around 0.5 and when the human stands on a single foot, it is either 0 (left foot) or 1 (right foot). The robot center of mass ground projection is then reconstructed on the line connecting its feet by means of this offset value. To also retarget changes of the center of mass that are not on the line connecting the feet, we can apply the same concept while considering the maximum backward and forward center of mass displacement in the orthogonal direction of the line connecting the feet as done in [1].

APPENDIX II
DATASETS

To train our method, we teleoperated the robot in an ideal network without any delay and recorded the corresponding robot motion. Every demonstration contains several Cartesian and postural trajectories that determine the whole-body motion (Fig. S2): the center of mass ground projection, the waist height, the hand positions, the arms posture (shoulder rotation, elbow flexion, forearm rotation), the neck posture (flexion and rotation) and the torso posture (flexion, rotation and abduction). We record the retargeted trajectories, that is the reference trajectories for the whole-body controller of the robot.

We used three different datasets (Table I), each one divided into a training set and a test set. The first dataset (Dataset Multiple Tasks) is designed to test how well the robot recognizes tasks and deals with the intrinsic variability of the operator's movements. This is the dataset used to perform the experiments on the real robot. The second one (Dataset Obstacles) is designed to evaluate the approach with unexpected obstacles. The third dataset (Dataset Goals) is designed to evaluate the approach with novel goal positions.

The datasets are available online at `https://doi.org/10.5281/zenodo.5913573`.

A. Dataset Multiple Tasks

This dataset covers two scenarios: reaching a bottle with the right hand (Fig. S1 and handling a box (Fig. S2).

The bottle task consists of demonstrations of 2 distinct whole-body reaching primitives: one primitive is for reaching the bottle on the table, the other one is for reaching the bottle on the top of the box. For each primitive, we recorded 6 repetitions of the task for training, for a total of 12 training whole-body demonstrations with an average duration of 6.1s. Every demonstration provided by the human operator is different, since it is not possible to exactly reproduce the same whole-body movement twice; to further increase the variance of the demonstrated movements, in 3 repetitions out of the 6, an obstacle was placed in between the robot and the bottle. To assess the quality of the predictions, 10 different testing repetitions were recorded for each of the two primitives; 5 with the obstacle in between the robot and the bottle, and 5 without any obstacle, for a total of 20 motions. In this dataset, the obstacles are at the same positions in both the training set and the testing set (see the dataset "Obstacles" below).

The second scenario consists of demonstrations of 7 distinct whole-body box handling primitives: 3 for picking up the box — from a low position, from a mid-height and from the table; 4 for placing the box at a specific location — on the floor, on the table, inside a container, or in a person's hands. For each primitive, we recorded 6 different repetitions for training, for a total of 42 training whole-body demonstrations, with an average duration of 7.2 s for the pick-up motions and of 5.8 s for the box-placing motions. For the test set, 3 new different repetitions of the 7 motions were recorded, for a total of 21 testing motions.

B. Dataset Obstacles

The training set is composed of 6 demonstrations of bottle reaching motions with 3 different locations of an obstacle (Fig. S3): 2 repetitions without obstacles, 2 with an obstacle in between the robot and the bottle, and 2 with a different obstacle. The average duration of the demonstrations is 6.9s. The test set consists of motions for the same task but with obstacles at different locations (Fig. S3b): 3 repetitions for each of the 3 distinct scenarios with different obstacles.

C. Dataset Goals

The training set is composed of 7 demonstrations of bottle reaching motions (Fig. S4), with the goal located in 7 different positions. The average duration of the demonstrations is 6.1s. The test set consists of motions reaching the same bottle but at 10 different locations (Fig. S4b).

APPENDIX III
DELAYED TELEOPERATION

A. Hardware and communication setup

The human motion is captured by the Xsens MVN system [47], which considers a human model comprising 66 degrees of freedom (corresponding to 22 spherical joints). The user teleoperating the robot is equipped with the wearable motion capture suit MVN Link, consisting of a Lycra suit with 17 inertial measurement units (IMUs) and a wireless data link transmitting at a frequency of 240Hz. Our compensation method receives the delayed data from the motion capture system at 100Hz and transmits to the robot controller at 50Hz.

The user teleoperating the robot is also equipped with a VR Oculus headset. Through the headset, the operator can visualize the delayed images from both an external camera at the robot side, as a third-person view of the teleoperated robot, and the robot cameras, for a first-person immersive experience. The communication protocol employed by the network is UDP with a bandwidth of 3Mbps. The forward delay is artificially generated, using the standard way to delay packets in Linux with the "netem" scheduling policy, which is based on the "iproute2" set of tools [48].

The images from the cameras at the robot side are delayed by using the open-source application Kinovea [49], which allows the user to set a constant delay for the streaming of the video. The resulting delayed streaming is projected onto the VR headset through the application Virtual Desktop [50].

B. Delay generation

The round-trip delay $\tau(t)$ at time-step t is divided into a forward $\tau_f(t)$ (operator to robot) and a backward delay $\tau_b(t)$ (robot to operator):

$$\tau(t) = \tau_f(t) + \tau_b(t)$$

Each one-way delay is composed of two parts, one deterministic and one stochastic [25]. The deterministic component corresponds to the transmission and propagation delays. It does not change when all the transmitted packets have the same format and size and use the same physical link [25]. The stochastic part, often called the "jitter", is mainly associated with the queueing delay [52] and varies from packet to packet, even when the packets have the same size and format.

If we denote by $\tau_{f,D}$ the deterministic part of τ_f and by $\tau_{f,S}$ the stochastic part:

$$\tau_f(t) = \tau_{f,D} + \tau_{f,S} \tag{S3}$$
$$\tau_b(t) = \tau_{b,D} + \tau_{b,S} \tag{S4}$$

In our experiments, we generate a forward delay that follows a normal distribution:

$$\tau_f(t) = \tau_{f,D} + \mathcal{N}\big(0, \sigma_{\tau_f}\big)$$

For both simulations and real experiments, we set the deterministic part of the forward delay $\tau_{f,D}$ between 100ms to 1s (depending on the experiment, see the captions of each figure) and the jitter σ_{τ_f} to $\frac{2}{15}\tau_{f,D}$, which is in line with what the jitter usually represents [53].

$$\tau_{f,D} \in [100, 1000] \text{ (depending on the experiment)} \tag{S5}$$
$$\sigma_{\tau_f} = \frac{2}{15}\tau_{f,D} \tag{S6}$$

For the stochastic part of the backward delay, we assume that the robot uses a video streaming software that implements a jitter buffer (sections "Delay estimation by the robot" and "Jitter buffer"), which is the case of all the modern video streaming systems. If we set the jitter buffer length to d, then this buffer adds an additional deterministic delay of d: all the packets that arrive before d seconds are re-ordered and packets that are not arrived are dropped (dropping a few frames has little consequence for a state-of-the-art video codec). As a consequence, from the operator's perspective, the backward delay is constant. This is why, for both simulations and real experiments, we generate a constant backward delay:

$$\tau_b(t) = \tau_{b,D}(t) \tag{S7}$$

For simplicity, we set $\tau_b(t) = \tau_{f,D}$ in all our experiments, but nothing in our system requires these two delays to be equal; in particular, it would be equivalent to set $\tau_b(t) = \tau_{b,D}(t) + K$ with K any constant delay caused by the jitter buffer.

C. Delay estimation by the robot

We assume that the clocks of the robot and of the computer of the operator are synchronized. In our real experiments, we synchronize the clock using the NTP system [26], which is the standard Unix protocol for time synchronization. The two clocks typically differ from less than 1ms on a local network [55]. Alternatively, GPS receivers can provide a highly accurate and absolute clock reference with an error of a few nano-seconds [55].

We add a time-stamp to each of the packets sent by the operator, which makes it possible for the robot to compute the forward delay $\tau_f(t)$ (this includes both the deterministic and stochastic part) when a packet is received at time t :

$$\tau_f(t) = \text{clock}_{\text{robot}}(t) - \text{timestamp}_{\text{operator}}(t) \tag{S8}$$

Please note that this does not assume that the delay follows a normal distribution. If necessary, the robot can re-order the packets according to the time-stamps to condition the trajectory predictions.

While the robot needs an estimate of the backward delay, it cannot know in advance the stochastic part before sending it. Our approach is to rely on the jitter buffer (section "Jitter buffer") implemented in the video streaming system to make the backward delay deterministic: if we set the jitter buffer length to d s on the operator receiving side, then we know that the delay caused by the jitter will be exactly equal to d.

In our experiment, we therefore assume that the backward delay is known and constant (100 ms, 250ms, etc. depending on the experiments). To keep the implementation simple and easy to reproduce, we assumed that the deterministic backward delay was always equal to the deterministic forward delay (a reasonable assumption given that the same link is used for both directions) and that the stochastic part is negligible (because we chose to not add any jitter on the backward delay in the robot experiment, see the Jitter buffer section below):

$$\tau_b(t) \quad = \overline{\tau_f(t)} = \tau_{f,D}(t) \tag{S9}$$

$$\tau_{b,S}(t) \qquad = 0 \tag{S10}$$

In a deployed setup, the robot would benefit from a better estimate of the average backward delay (the deterministic part) and the length of the jitter buffer. To do so, most video streaming systems use the RTCP protocol [28] to get out-of-band statistics and control information for a video streaming session that follows the RTP protocol [57]. This data would need to be sent periodically from the operator's computer to the robot so that the robot knows both d and $\tau_{b,D}$ (which are not expected to change at high speed). Alternatively, a custom system can be designed by using time-stamps on the image packets to gather statistics about the delay.

D. Jitter buffer

The jitter buffer is the component of a video streaming system [27], [58] that re-orders packets if they are delayed by less than the length of the buffer and drops packets that are too late. Much work has been dedicated to adapt its length automatically [27]: if it is too small, then the video is jittery, but if it is too large, delays are added, which is detrimental to the user (in particular during video calls). In our system, we assume that the length is fixed and known to the robot, for simplicity.

We did not implement a jitter buffer because we wanted to avoid modifying the video streaming system: reordering or dropping packets would require a considerable expertise in the internals of both the encodings (e.g., MP4) and the video streaming software. Instead, we consider that video streaming with delay and jitter is a problem that is well solved by all the current video streaming systems, as experienced by the million of users who watch videos online on smartphones with non-ideal connections.

To summarize, from the point of view of our system, the jitter buffer results in an additional but deterministic delay. However, we assume that the robot knows the value of this additional delay as well as the deterministic part of the delay.

APPENDIX IV
PROBABILISTC MOTION PRIMITIVES (PROMPS) FOR PRESCIENT TELEOPERATION

A. Definition of Probabilist Motion Primitives

A ProMP [31] is a probabilistic model for representing a trajectory distribution. The movement primitive representation models the time-varying mean and variance of the trajectories and is based on basis functions. A single trajectory is represented by a weight vector $\boldsymbol{w} \in \mathbb{R}^m$. The probability of observing a trajectory $\boldsymbol{y}$ given the underlying weight vector is given as a linear basis function model

$$\boldsymbol{\xi}_t = \boldsymbol{\Phi}_t \boldsymbol{w} + \boldsymbol{\epsilon}_\xi, \tag{S11}$$

$$p(\boldsymbol{y}|\boldsymbol{w}) = \prod_t \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\Phi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_\xi), \tag{S12}$$

where $\boldsymbol{\Sigma}_\xi$ is the observation noise variance, $\boldsymbol{\epsilon}_\xi \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\xi)$ is the trajectory noise. The matrix $\boldsymbol{\Phi}_t \in \mathbb{R}^m$ corresponds to

the m normalized radial basis functions evaluated at time t, with

$$\phi_c(t) = \frac{\exp\Big(-\frac{1}{2}\big(t-\frac{c-1}{m-1}\big)^2\Big)}{\sum_{c=1}^{m}\exp\Big(-\frac{1}{2}\big(t-\frac{c-1}{m-1}\big)^2\Big)}, \tag{S13}$$

where the variable $c \in \{1, 2, ..., m\}$ specifies the center of each basis function. The distribution $p(\boldsymbol{w};\boldsymbol{\theta})$ over the weight vector $\boldsymbol{w}$ is Gaussian, with parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}\}$ specifying the mean and the variance of $\boldsymbol{w}$. The trajectory distribution $p(\boldsymbol{y};\boldsymbol{\theta})$ is obtained by marginalizing out the weight vector $\boldsymbol{w}$, i.e.

$$p(\boldsymbol{y},\boldsymbol{\theta}) = \int p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w};\boldsymbol{\theta})d\boldsymbol{w}. \tag{S14}$$

B. Learning ProMPs from demonstrations

The demonstrations are trajectories retargeted from the human. These are recorded in an "offline phase", while the user teleoperates the robot within a local network (approximately without delays) to perform a variety of tasks. Since the duration of each recorded trajectory may be different, a phase variable $\upsilon \in [0, 1]$ is introduced to decouple the movement from the time signal, obtaining a common representation in terms of primitives that is duration independent [32]. For each task, the modulated trajectories $\boldsymbol{\xi}_i(\upsilon)$ are then used to learn a ProMP. The parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$ of the ProMP are estimated by using a simple maximum likelihood estimation algorithm. For each demonstration i, we compute the weights with linear ridge regression, i.e.

$$\boldsymbol{w}_i = \big(\boldsymbol{\Phi}_\upsilon^\top\boldsymbol{\Phi}_\upsilon + \lambda\big)^{-1}\boldsymbol{\Phi}_\upsilon^\top\boldsymbol{\xi}_i(\upsilon), \tag{S15}$$

where the ridge factor λ is generally set to a very small value, typically $\lambda = 10^{-12}$ as in our case, as larger values degrade the estimation of the trajectory distribution. Assuming Normal distributions $p(\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, the mean $\boldsymbol{\mu}_w$ and covariance $\boldsymbol{\Sigma}_w$ can be computed from the samples $\boldsymbol{w}_i$:

$$\boldsymbol{\mu_w} = \frac{1}{D}\sum_{i=1}^{D}\boldsymbol{w}_i, \qquad \boldsymbol{\Sigma_w} = \frac{1}{D}\sum_{i=1}^{D}(\boldsymbol{w}_i - \boldsymbol{\mu_w})(\boldsymbol{w}_i - \boldsymbol{\mu_w})^\top, \tag{S16}$$

where D is the number of demonstrations. Since a whole-body trajectory is represented by N trajectories (x, y, z position of the center of mass, of the hands, etc.), we learn a ProMP for each of the N trajectories. These ProMPs all together encode the same task k.

C. Recognizing the category of motion

Each learned k-th set of N ProMPs encodes different whole-body trajectories to accomplish a given task like picking up a box or squatting. To recognize to which set k the current teleoperated motion belongs to, we can minimize the distance between the first n_{obs} delayed observations and the mean of the N ProMPs of a group k, as done in [32]:

$$\hat{k} = \arg\min_{k\in[1:K]}\left[\sum_{n=1}^{N}\sum_{t\in T_{obs}}|\boldsymbol{y}_n(t-\tau_f(t)) - \Phi_{n,t-\tau_f(t)}\boldsymbol{\mu}_{n,\boldsymbol{w}_k}|\right], \tag{S17}$$

where K is the number of tasks in the dataset and $T_{obs} = \{t_1, ..., t_{n_{obs}}\}$ is the set of timesteps associated to the n_{obs} early observations. While computing $\hat{k}$, the ProMPs are modulated to have a duration equal to the mean duration of the demonstrations. The recognition (S17) starts whenever a motion is detected, i.e. the derivative of the observed end-effector trajectories exceeds a given threshold. The distance in (S17) is continuously computed after having identified the current motion. In this way, we can verify that the observations do not diverge from the demonstrations (exceed by a given threshold the demonstrated variance), in which case a gradual switch to delayed teleoperation is performed.

D. Time-modulation of the ProMPs

During motion recognition, we assumed that the duration of the observed trajectories is equal to the mean duration of the demonstrated trajectories, which might not be true. To match as closely as possible the exact speed at which the movement is being executed by the human operator, we have to estimate the actual trajectory duration. More specifically, we want to find the time modulation α, that maps the actual duration of a given (observed) trajectory to the mean duration of the associated demonstrated trajectories.

During the learning step, for each k-th set of ProMPs we record the set of α parameters associated to the demonstrations: $S_{\alpha k} = \{\alpha_1, ..., \alpha_n\}$. Then, from this set, we can estimate which α better fits the current movement speed. We considered the best $\hat{\alpha}$ to be the one that minimizes the difference between the observed trajectory and the predicted trajectory for the first n_{obs} datapoints:

$$\hat{\alpha} = \arg\min_{\alpha\in S_{\alpha\hat{k}}}\left\{\sum_{t\in T_{obs}}|\boldsymbol{y}(t-\tau_f(t)) - \Phi_{\alpha(t-\tau_f(t))}\boldsymbol{\mu}_{\boldsymbol{w}_{\hat{k}}}|\right\}. \tag{S18}$$

E. Updating the posterior distribution of the ProMPs

Once the $\hat{k}$-th most likely set of ProMPs and their duration has been identified, we continuously update their posterior distribution to take into account the observations that arrive at the robot side. Each ProMP has to be conditioned to reach a certain observed state $\boldsymbol{y}_t^*$. The conditioning for a given observation $\boldsymbol{x}_t^* = \{\boldsymbol{y}_t^*, \boldsymbol{\Sigma}_y^*\}$ (with $\boldsymbol{\Sigma}_y^*$ being the accuracy of the desired observation) is performed by applying Bayes theorem

$$p(\boldsymbol{w}_{\hat{k}}|\boldsymbol{x}_t^*) \propto \mathcal{N}(\boldsymbol{y}_t^*|\boldsymbol{\Phi}_{\hat{\alpha}t}\boldsymbol{w}_{\hat{k}}, \boldsymbol{\Sigma}_y^*)p(\boldsymbol{w}_{\hat{k}}). \tag{S19}$$

The conditional distribution of $p(\boldsymbol{w}_{\hat{k}}|\boldsymbol{x}_t^*)$ is Gaussian with mean and variance

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{w}_{\hat{k}}} = \boldsymbol{\mu}_{\boldsymbol{w}_{\hat{\boldsymbol{k}}}} + \boldsymbol{L}\big(\boldsymbol{y}_t^* - \boldsymbol{\Phi}_{\hat{\alpha}t}^\top\boldsymbol{\mu}_{\boldsymbol{w}_{\hat{\boldsymbol{k}}}}\big), \tag{S20}$$

$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{w}_{\hat{k}}} = \boldsymbol{\Sigma}_{\boldsymbol{w}_{\hat{\boldsymbol{k}}}} - \boldsymbol{L}\boldsymbol{\Phi}_{\hat{\alpha}t}^\top\boldsymbol{\Sigma}_{\boldsymbol{w}_{\hat{\boldsymbol{k}}}}, \tag{S21}$$

where

$$\boldsymbol{L} = \boldsymbol{\Sigma}_{\boldsymbol{w}_{\hat{k}}}\boldsymbol{\Phi}_{\hat{\alpha}t}\big(\boldsymbol{\Sigma}_y^* + \boldsymbol{\Phi}_{\hat{\alpha}t}^\top\boldsymbol{\Sigma}_{\boldsymbol{w}_{\hat{k}}}\boldsymbol{\Phi}_{\hat{\alpha}t}\big)^{-1}. \tag{S22}$$

Given the delay in the transmitted data $\tau_f(t)$, we can compute the timestep t^* at which the ProMP has to be

conditioned to a certain observation $\boldsymbol{x}_t^*$:

$$t^* = t - \tau_f(t) - t_0. \tag{S23}$$

where t_0 is the starting time of the current motion.

F. Motion anticipation

The references for the robot controller are generated at each time based on the updated ProMPs' mean trajectories $\hat{\boldsymbol{\mu}}_{\boldsymbol{w}_{\hat{k}}}$. For a given ProMP, the sample $\hat{\boldsymbol{\mu}}_{\boldsymbol{w}_{\hat{k}}}(t^*)$ corresponding to the last conditioned observation, is a reconstruction of the past retargeted human input

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{w}_{\hat{k}}}(t^*) = \hat{\boldsymbol{y}}(t - \tau_f(t)). \tag{S24}$$

The sample $\hat{\boldsymbol{\mu}}_w(t^* + \tau_f(t))$ is an estimate of the current retargeted human input

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{w}_{\hat{k}}}(t^* + \tau_f(t)) = \hat{\boldsymbol{y}}(t), \tag{S25}$$

and can be used to synchronize the human movement with that of the robot, compensating only the forward delay (see Fig. 2). In our case, we want to synchronize the motion of the human operator with what is seen from the robot side, thus compensating for both the forward and backward delays. To do so, we select the sample $\hat{\boldsymbol{\mu}}_w(t^* + \tau_f(t) + \hat{\tau}_b(t))$ as a control reference, which corresponds to a future prediction of the retargeted human movements:

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{w}_{\hat{k}}}(t^* + \tau_f(t) + \hat{\tau}_b(t)) = \hat{\boldsymbol{y}}(t + \hat{\tau}_b(t)). \tag{S26}$$

The remaining samples $[\hat{\boldsymbol{\mu}}_{\boldsymbol{w}_{\hat{k}}}(t^*+\tau_f(t)+\hat{\tau}_b(t)+1), \hat{\boldsymbol{\mu}}_{\boldsymbol{w}_{\hat{k}}}(t^*+\tau_f(t) + \hat{\tau}_b(t) + 2), ...]$ are also given to the controller. They are used as control references if a new reference cannot be computed in the next control step due to packet losses or jitter.

After generating a first prediction, the transition from delayed to predicted references can be discontinuous (Fig. S6). To smoothen the transition, a policy blending arbitrates the delayed received references $\boldsymbol{y}(t - \tau_f(t))$ and the predicted ones $\hat{\boldsymbol{y}}(t+\hat{\tau}_b(t)|t-\tau_f(t))$, determining the adjusted reference (Fig. S6):

$$\hat{\boldsymbol{y}}'(t + \hat{\tau}_b(t)|t - \tau_f) = (1 - \boldsymbol{\beta})\boldsymbol{y}_d + \boldsymbol{\beta}\boldsymbol{y}_p, \tag{S27}$$

where $\boldsymbol{y}_d = \boldsymbol{y}(t - \tau_f(t))$, $\boldsymbol{y}_p = \hat{\boldsymbol{y}}(t + \hat{\tau}_b(t)|t - \tau_f(t))$, $\boldsymbol{\beta} = \{\beta_0, ..., \beta_n, ..., \beta_N\}^\top$ with $\beta_n \in\ ]0, 1[$

$$\beta_n = \frac{1}{1 + e^{-12(\frac{i}{\Delta y_n} - \frac{1}{2})}}, \tag{S28}$$

$i = \{0, 1, ..., \Delta y_n\}$ and Δy_n is the initial difference between a delayed reference and the corresponding prediction expressed in mm (for Cartesian trajectories) or $deg \times 10^{-1}$ (for postural trajectories).

G. Teleoperation under unexpected circumstances

If something unexpected happens, or if the operator suddenly changes their mind about what to do and the ongoing motion cannot be completed or is significantly altered, the prescient teleoperation is transitioned back to the delayed teleoperation. The transition from predicted to delayed references is triggered whenever the distance between the current observation and learned mean exceeds a given threshold Δ_σ, which in the experiments was fixed equal to the learned variance plus 5cm and considered for each of the x, y, z trajectories of the hands. Since the transition can be discontinuous, a policy blending arbitrates the last predicted sample $\hat{\boldsymbol{y}}(t_{last} + \hat{\tau}_b(t_{last})|t_{last} - \tau_f(t_{last}))$ and the delayed received references $\boldsymbol{y}(t - \tau_f(t))$:

$$\hat{\boldsymbol{y}}'(t + \hat{\tau}_b(t)|t - \tau_f) = (1 - \boldsymbol{\beta})\boldsymbol{Y}_p + \boldsymbol{\beta}\boldsymbol{y}_d, \tag{S29}$$

where $\boldsymbol{y}_d = \boldsymbol{y}(t - \tau_f(t))$, $\boldsymbol{Y}_p = \hat{\boldsymbol{y}}(t_{last} + \hat{\tau}_b(t_{last})|t_{last} - \tau_f(t_{last}))$, $\boldsymbol{\beta} = \{\beta_0, ..., \beta_n, ..., \beta_N\}^\top$ with β_n defined as in (S28).

H. Comparison between ProMPs and LSTM

We implemented a LSTM (Long Short Term Memory network) [62] using the Pytorch library [63]. This LSTM predicts the next k time-steps given the previous k time-steps. To keep the network small, the trajectories are sub-sampled from 100Hz to 10Hz, so that 10 time-steps correspond to 1 second of motion. For instance, to predict the next 2 seconds of motions, the LSTM has 20 inputs and 20 outputs. The network has 10 hidden nodes (preliminary experiments showed that increasing this number did not have any qualitative effect on the predictions). It is trained with the Adam optimizer using mini-batches of size 16, for 100 epochs (our experiments show that this is enough to reach the minimum loss), and the Mean Squared loss function. To account for the stochasticity of the initialization and the stochastic gradient descent, 10 LSTMs are trained with different seeds.

Like with ProMPs, one LSTM is trained for each dimension of the prediction (e.g., a LSTM for the x-coordinate of the left hand, another one for the y-coordinate, etc.); however, contrary to ProMPs, the whole "Multi tasks" training set is used to train each LSTM. A different set of LSTMs is trained for 1-second prediction (Fig. S7) and 2-second prediction (Fig. S8). The LSTMs that were trained for 2 seconds could have been used to predict 1-second ahead, but at the risk of being lower-performing than a network specialized in 1-second predictions.

For this comparison, the ProMPs are conditioned at each time-step with all the points since the beginning of the trajectory, and the ProMP is queried to predict the value at $t + 2$ seconds(or $t + 1$s). As an additional baseline, the "delayed" trajectory is the operator's trajectory shifted in time by either one or two seconds, which corresponds to what the robot would do if there were no compensation.

For a particular trajectory T and a particular dimension D (e.g., x-position of the hand), the prediction error is the sum of the differences between the prediction and the recorded trajectory:

$$e_{T,D} = \frac{1}{N} \sum_{i=1}^{i=N} \left| x_i^{(pred)} - x_i^{(gt)} \right| \tag{S30}$$

where N is the number of points in the trajectory, $x_i^{(pred)}$ the prediction and $x_i^{(gt)}$ the point of the trajectory performed by the operator at time-step i (unknown to the robot at this time-step).

APPENDIX V
SOURCE CODE

The code of the ProMP library is available at `https://doi.org/10.5281/zenodo.7438257` and on `https://github.com/hucebot/promp`. The comparison between ProMP and LSTM is available as a python notebook at `https://doi.org/10.5281/zenodo.7441367` and on `https://github.com/hucebot/lstm_vs_promp` (this code relies on the ProMP library).

REFERENCES

[41] L. Natale, C. Bartolozzi, D. Pucci, A. Wykowska, and G. Metta, "iCub: The not-yet-finished story of building a robot child," *Science Robotics*, vol. 2, no. 13, p. eaaq1026, 2017.

[42] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[43] L. Penco, E. M. Hoffman, V. Modugno, W. Gomes, J. Mouret, and S. Ivaldi, "Learning robust task priorities and gains for control of redundant robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2626–2633, 2020.

[44] A. Rocchi, E. M. Hoffman, D. G. Caldwell, and N. G. Tsagarakis, "Opensot: a whole-body control library for the compliant humanoid robot coman," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6248–6253.

[45] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[46] L. Penco *et al.*, "Robust real-time whole-body motion retargeting from human to humanoid," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 425–432.

[47] D. Roetenberg, H. Luinge, and P. Slycke, "Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors," *Xsens Motion Technol. BV Tech. Rep.*, vol. 3, 01 2009.

[48] T. L. Foundation, "Iproute2." [Online]. Available: https://wiki.linuxfoundation.org/networking/iproute2

[49] "Kinovea. A microscope for your videos," https://www.kinovea.org/.

[50] "Virtual Desktop. Your PC in VR," https://www.vrdesktop.net/.

[51] O. Gurewitz, I. Cidon, and M. Sidi, "One-way delay estimation using network-wide measurements," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2710–2724, 2006.

[52] M. Garetto and D. Towsley, "Modeling, simulation and measurements of queuing delay under long-tail internet traffic," in *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '03. New York, NY, USA: Association for Computing Machinery, 2003, pp. 47–57. [Online]. Available: https://doi.org/10.1145/781027.781034

[53] A. Alharbi, A. Bahnasse, and M. Talea, "A comparison of voip performance evaluation on different environments over vpn multipoint network," *International Journal of Computer Science and Network Security*, vol. 17, pp. 123–128, 05 2017.

[54] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," Internet Requests for Comments, RFC 5905, June 2010.

[55] D. Veitch, S. Babu, and A. Pàsztor, "Robust synchronization of software clocks across the internet," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 219–232. [Online]. Available: https://doi.org/10.1145/1028788.1028817

[56] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," Internet Requests for Comments, RFC Editor, STD 64, 2003.

[57] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey, "Extended RTP profile for real-time transport control protocol (RTCP)-based feedback (RTP/AVPF)," *RFC*, vol. 4585, pp. 1–51, 2006.

[58] M. Claypool and J. Tanner, "The effects of jitter on the peceptual quality of video," in *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, 1999, pp. 115–118.

[59] B. Oklander and M. Sidi, "Jitter buffer analysis," in *Proc. Int. Conf. on Comp. Comm. Networks, ICCCN*, 09 2008, pp. 1 – 6.

[60] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, p. 529–551, 07 2018.

[61] O. Dermy, A. Paraschos, M. Ewerton, J. Peters, F. Charpillet, and S. Ivaldi, "Prediction of intention during interaction with iCub with probabilistic movement primitives," *Frontiers in Robotics and AI*, vol. 4, pp. 45–45, 2017.

[62] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

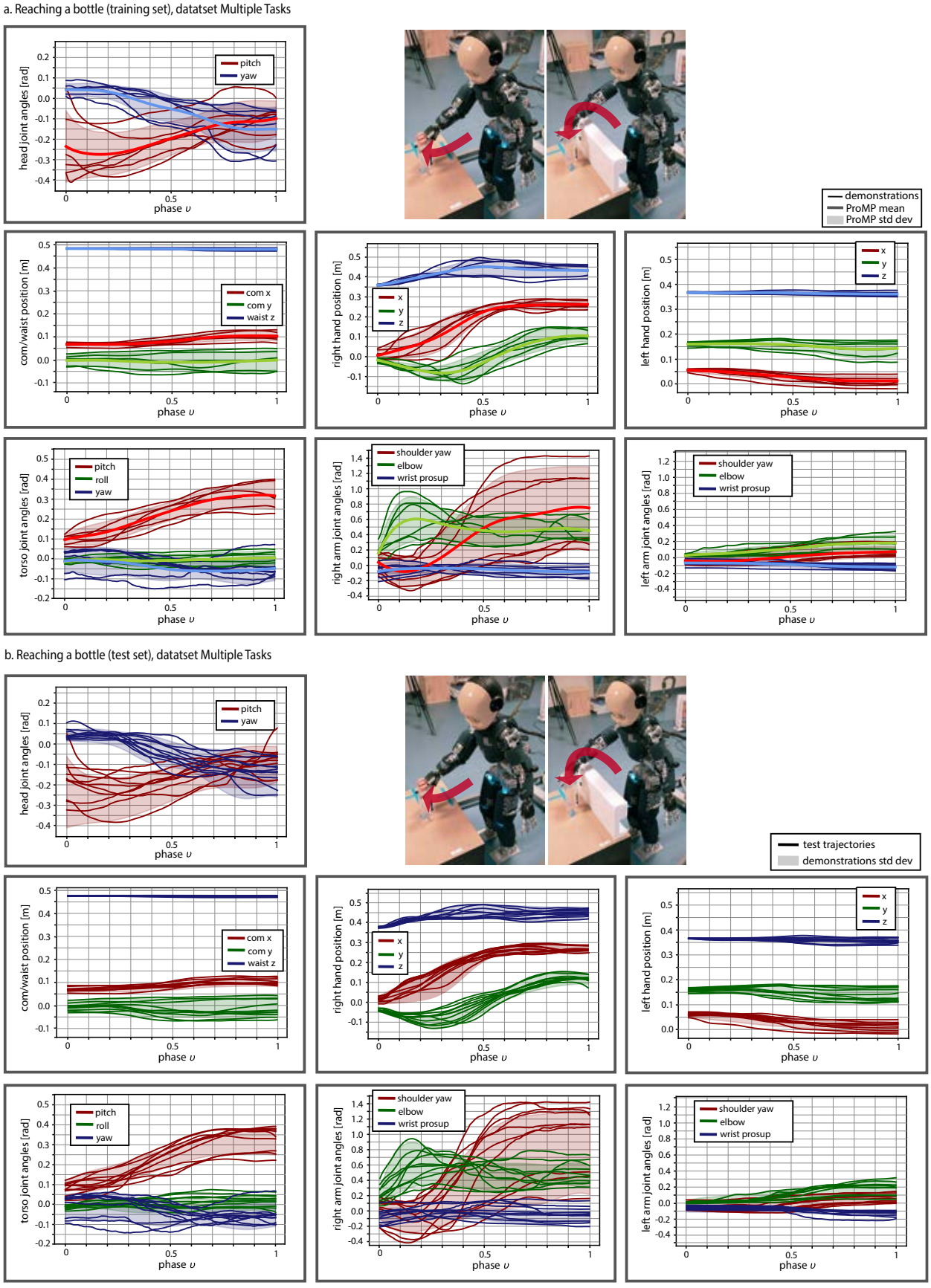

Fig. S1. **Dataset "Multiple Tasks", scenario reaching a bottle on the table. a. Training trajectories and learned ProMPs.** The whole-body motion of the teleoperated robot is obtained by following the reference trajectories retargeted from the human. We learned a ProMP for each of these trajectories, given 6 demonstrations in a local network without any delay (3 with an obstacle in between the robot and the bottle, and 3 without). **b. Test trajectories.** The test trajectories are different and additional repetitions of the training motions. For the task of reaching a bottle 10 different repetitions were recorded (5 with an obstacle in between the robot and the bottle, and 5 without).

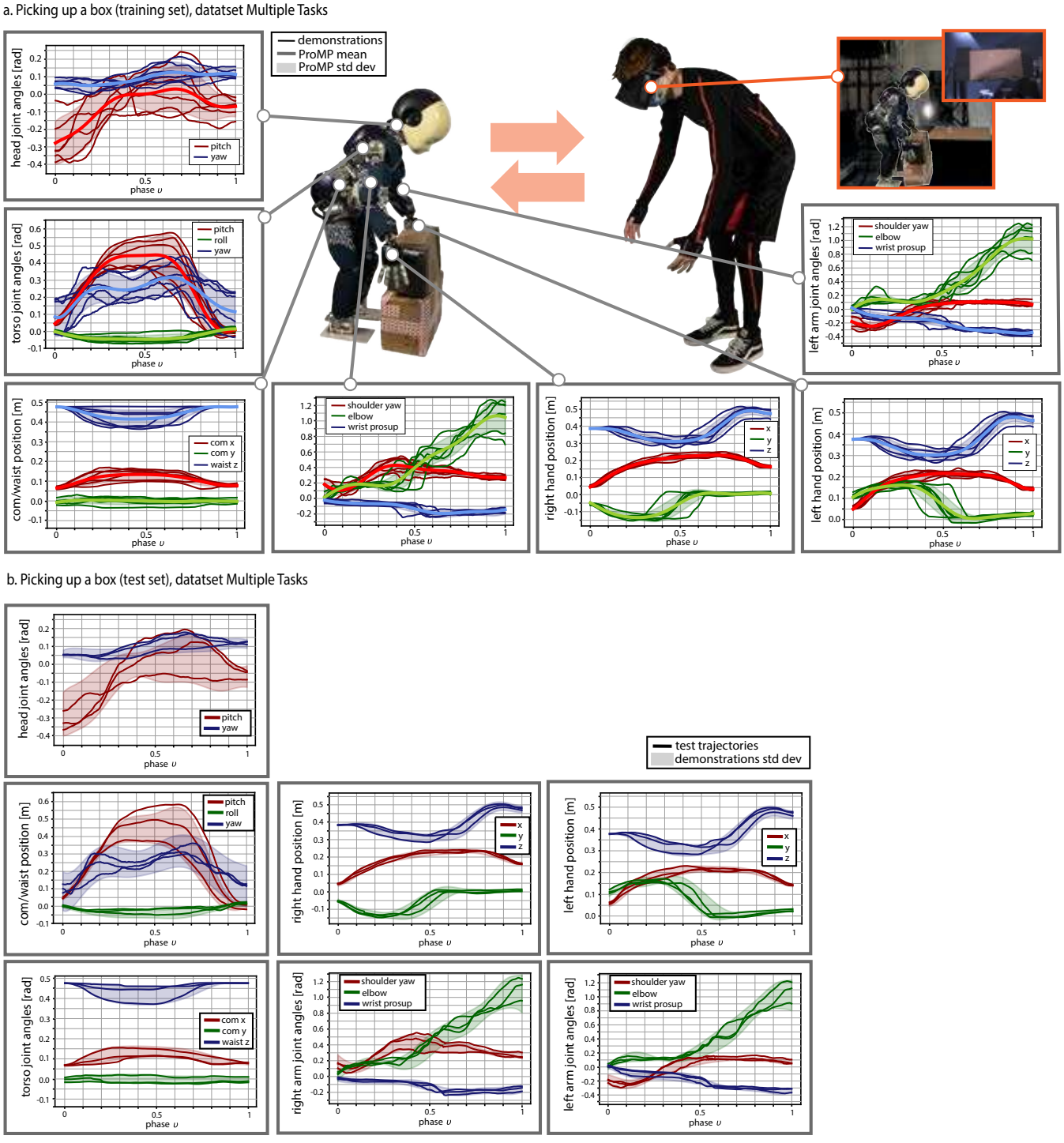

Fig. S2. **Dataset "Multiple Tasks", scenario picking up a box at a low position. a. Training trajectories and learned ProMPs.** For each of the 6 demonstrations, the motion of the operator is first "retargeted" to the robot using the whole-body controller (ignoring delays). The trajectories of each body/joint of the robot is then recorded. From this set of demonstrations (thin lines), a ProMP is fitted for each trajectory; this ProMP is represented here as a thick line (the mean) and a light zone (the standard deviation). The computed mean is a smooth trajectory that averages all the demonstrations and the standard deviation captures the variability of the demonstrations. **b. Test trajectories.** The test trajectories are different and additional repetitions of the training motions. For the tasks of picking up a box 3 different repetitions were recorded.

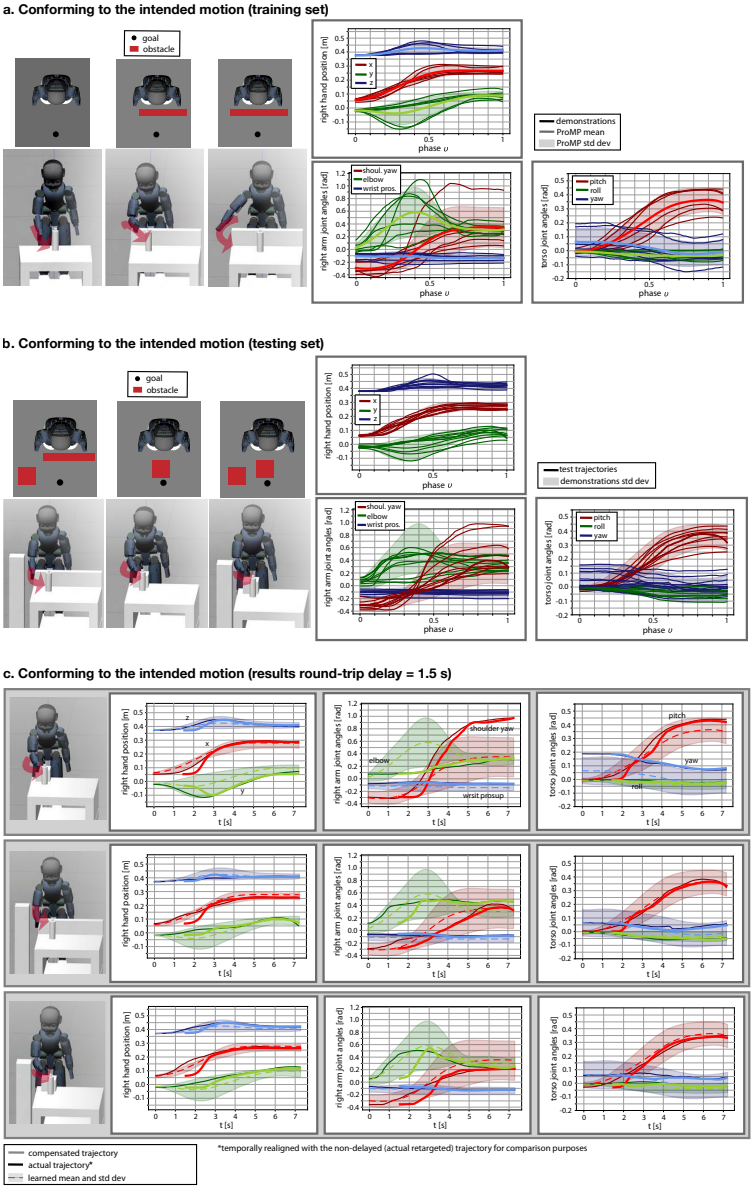

Fig. S3. **Dataset "Obstacles": Conforming the teleoperation to the intended motion. a. Training trajectories and learned ProMPs.** The most relevant learned ProMPs and the associated demonstrations are reported. The 6 demonstrations (2 without obstacles, 2 with an obstacle in between the robot and the bottle and other 2 with a different obstacle) have been recorded while teleoperating the robot in simulation, in a local network without any delay. **b. Test trajectories.** The 9 test trajectories are different from those used for training and consist of 3 repetitions of the bottle reaching motion for each of the 3 distinct simulated scenarios with different obstacles, illustrated on the left. **c. Results: comparison between the compensated trajectory and the ideal (non-delayed) trajectory with a mean round-trip delay of 1.5s.** On the top row, there is an unexpected obstacle (a small box) to avoid and the operator approaches the object by the right; on the second row, the obstacle in front of the robot is different; on the bottom row, there is the same obstacle from the top row with in addition a large obstacle on the right of the robot, which forces the operator to move the hand in between the two obstacles. These situations were not in the training set. After the initial recognition period, our approach makes the robot follow the specific way the human is performing the task despite the delay, and even if the robot is asked to perform the task in a way that has not been demonstrated before (but included in the distribution of the demonstrations). This is not the same as following the mean of previously demonstrated motions (here, the dashed line) or letting the robot replicate previously demonstrated motions. The non-delayed trajectories are some of the test trajectories from panel b, where the robot has to reach the bottle on the table in the presence of different obstacles that were not considered during the training.

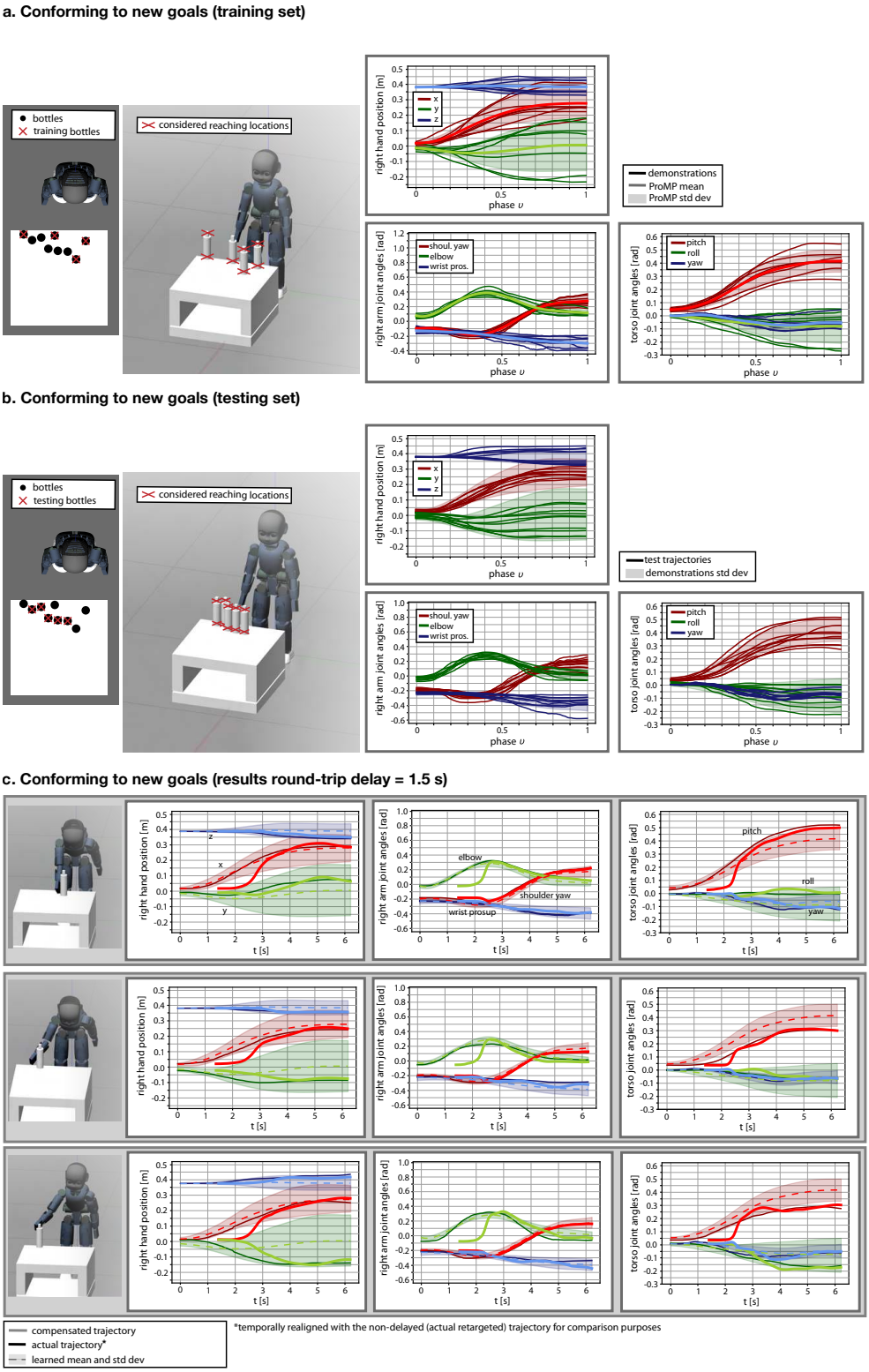

Fig. S4. **Dataset "Goals": Conforming the teleoperation to new goals. a. Training trajectories and learned ProMPs.** The most relevant learned ProMPs and the associated demonstrations are reported. The 7 demonstrations have been recorded while teleoperating the robot in simulation, in a local network without any delay. The different bottle locations are illustrated on the left. **b. Test trajectories.** The 10 test trajectories are different from those used for training. The different bottle locations are illustrated on the left. **c. Results: comparison between the compensated trajectory and the ideal (non-delayed) trajectory with a mean round-trip delay of 1.5s.** The bottles are located on the table in different positions that were not in the training set (but included in the distribution of the demonstrations). The non-delayed trajectories are some of the test trajectories from panel b, where the robot has to reach the bottle on the table in the presence of different obstacles that were not considered during the training.

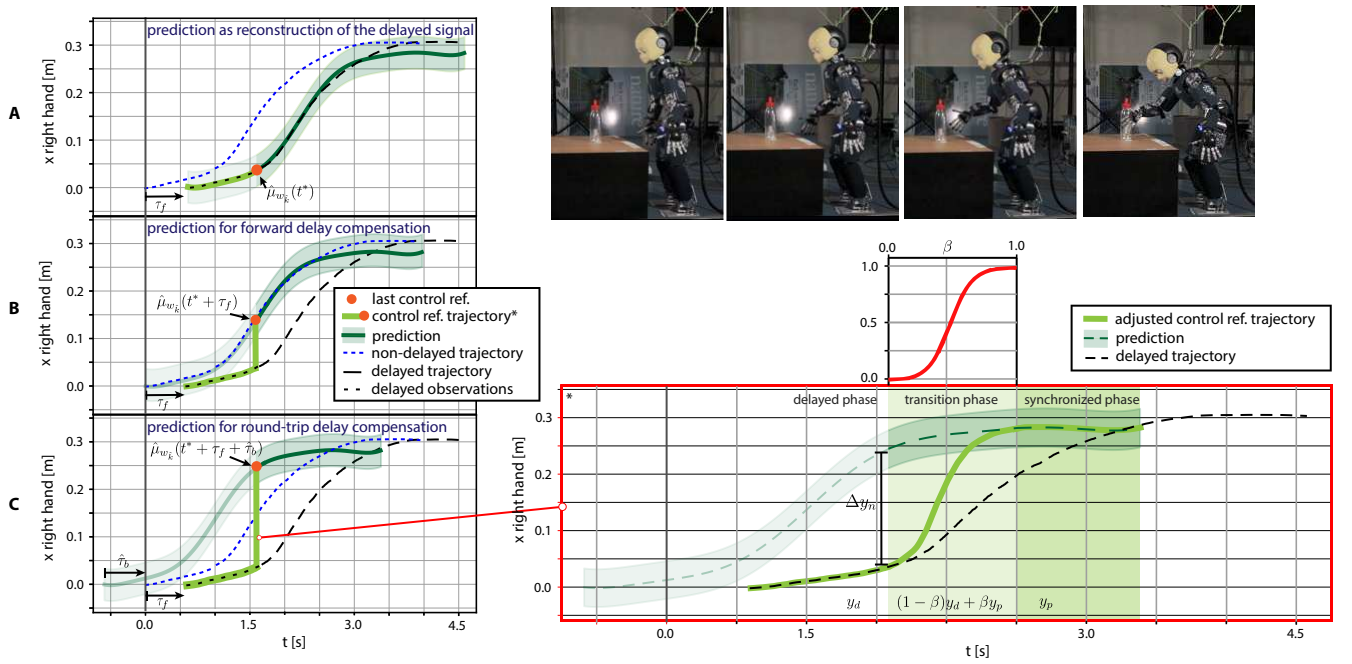

Fig. S6. **Round-trip delay compensation.** Given the past delayed observations, the robot produces at each time a prediction ($\hat{\mu}_{w_k}$) of the current command. To compensate for the delays, the right sample (orange dot) from the prediction has to be selected as reference for the robot controller at each time. (A) The sample corresponding to the last received observation is an estimate of the delayed command. (B) By knowing the forward delay $\tau_f(t)$, a sample from the prediction can be selected to achieve a synchronization between the operator's movement and the robot movements. (C) By knowing the forward $\tau_f(t)$ and backward delay $\tau_b(t)$, the robot can select the right sample from its prediction so as to achieve a synchronization between the operator's movement and the feedback from the robot side. A policy blending arbitrates the delayed observations with the samples selected from the prediction, which guarantees a smooth transition from delayed to compensated teleoperation.

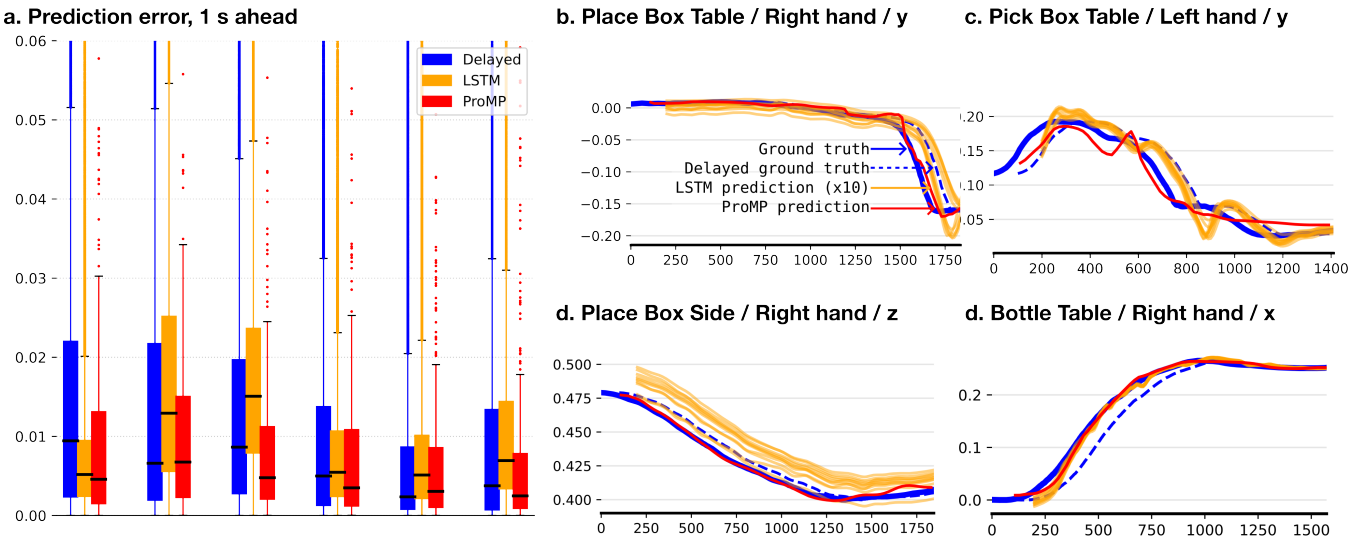

Fig. S7. **Prediction for LSTM and ProMP, 1 second ahead. a. Prediction error, 1-s horizon.** The plot shows the average difference between the point predicted 1 second ahead (100 time-steps) and the ground truth (Methods), for each degree coordinate of each hand, for each trajectory of the dataset "Multiple tasks" (Methods). The box extends from the lower to upper quartile values of the data, with a line at the median. For the LSTM, the 10 replicates of the learning process (with different seeds) are considered as independent data (i.e., the variance of the prediction error comes both from the different trajectories and the different seeds). The "Delayed" trajectory corresponds to the original trajectory delayed by 1 second, that is, to what the robot would have done without any prediction system. All comparisons are statistically significant ($p < 10^{-6}$, Mann-Whitney U-test). **b-d. Examples of predicted trajectories.**

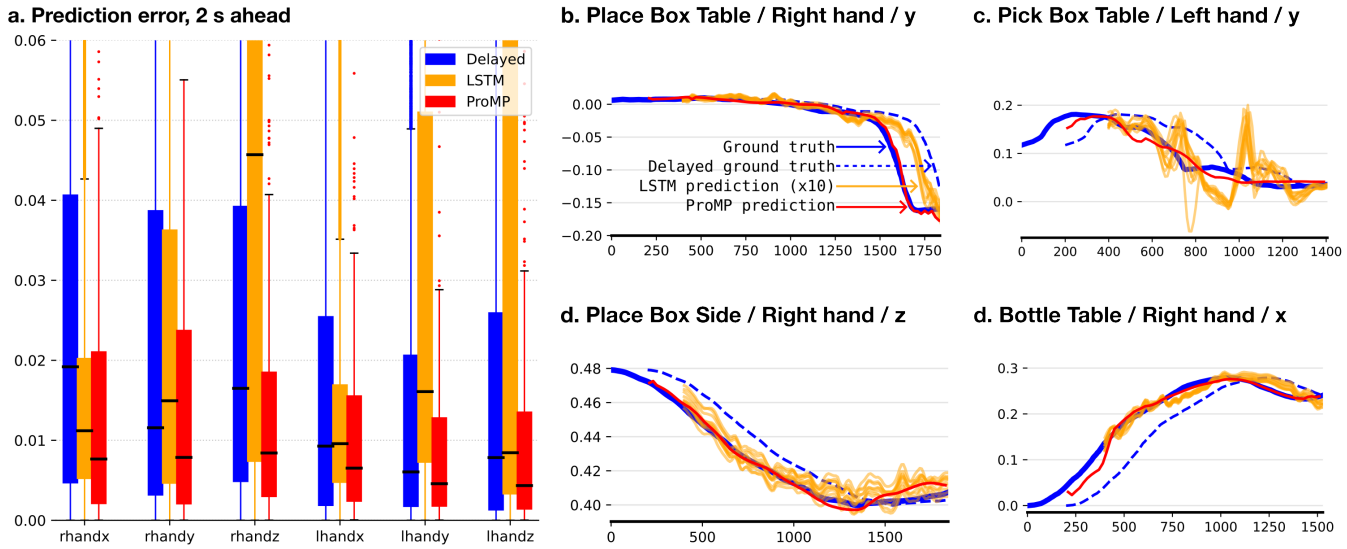

Fig. S8. **Prediction for LSTM and ProMP, 2 seconds ahead. a. Prediction error, 2-s horizon.** The plot shows the average difference between the point predicted 2 seconds ahead (200 time-steps) and the ground truth (Methods), for each degree coordinate of each hand, for each trajectory of the dataset "Multiple tasks" (Methods). The box extends from the lower to upper quartile values of the data, with a line at the median. For the LSTM, the 10 replicates of the learning process (with different seeds) are considered as independent data (i.e., the variance of the prediction error comes both from the different trajectories and the different seeds). The "Delayed" trajectory corresponds to the original trajectory delayed by 2 seconds, that is, to what the robot would have done without any prediction system. All comparisons are statistically significant ($p < 10^{-6}$, Mann-Whitney U-test). All comparisons are statistically significant ($p < 10^{-6}$, Mann-Whitney U-test). **b-d. Examples of predicted trajectories.**

TABLE S2

DIFFERENCE (ROOT MEAN SQUARE ERROR) WITH THE NON-DELAYED TRAJECTORIES, FOR BOTH THE COMPENSATED AND THE NON-COMPENSATED (DELAYED) TRAJECTORIES (AVERAGE DELAY: 1.5 S) THE ERROR IS COMPUTED FOR THE 20 TESTING MOTIONS FROM THE BOTTLE REACHING SCENARIO OF THE DATASET MULTIPLE TASKS (FIG. S1), AND FOR THE 21 TESTING MOTIONS FROM THE BOX HANDLING SCENARIO OF THE DATASET MULTIPLE TASKS (FIG. S2). THE TIME-VARYING FORWARD FOLLOWS A NORMAL DISTRIBUTION WITH 750MS AS MEAN AND 100MS AS STANDARD DEVIATION. THE BACKWARD DELAY IS SET EQUAL TO 750MS.

	Box handling		Bottle reaching	
	RMS error [rad]		**RMS error [rad]**	
	compensation	no compensation	compensation	no compensation
head yaw	0.024±0.011	0.035±0.012	0.013±0.007	0.021±0.011
torso pitch	0.045±0.020	0.136±0.064	0.027±0.012	0.041±0.019
torso roll	0.022±0.011	0.089±0.038	0.015±0.008	0.020±0.010
torso yaw	0.069±0.028	0.129±0.055	0.019±0.009	0.032±0.011
r. shoulder yaw	0.071±0.025	0.145±0.051	0.065±0.018	0.221±0.092
r. elbow	0.062±0.020	0.171±0.067	0.096±0.030	0.194±0.071
r. wrist prosup.	0.025±0.007	0.077±0.033	0.054±0.012	0.091±0.041
	RMS error [cm]		**RMS error [cm]**	
	compensation	no compensation	compensation	no compensation
r. hand x	1.02±0.31	2.95±1.12	1.29±0.33	4.97±1.46
r. hand y	0.90±0.26	3.36±1.21	1.21±0.31	4.33±1.17
r. hand z	0.96±0.30	2.96±0.75	1.11±0.25	4.15±1.13
com x	0.90±0.13	1.24±0.36	0.33±0.07	1.01±0.20
com y	0.79±0.11	1.06±0.39	0.24±0.06	1.01±0.32
waist z	0.88±0.14	2.02±1.22	0.22±0.07	0.61±0.09