An Efficient PbD Framework for Fast Deployment of Bi-manual Assembly Tasks

Bojan Nemec, Leon Žlajpah, Sebastjan Šlajpah[1], Jožica Piškur[1], Aleš Ude

Abstract— **We propose a two-phase programming by demonstration (PbD) framework, which enables fast deployment of complex bi-manual assembly tasks. The first phase is a pre-learning phase, where the robot observes multiple task demonstrations performed by humans. Applying motion segmentation, it builds a rough plan of the task to be accomplished. Next phase is the policy refinement with incremental learning, performed by the kinesthetic guidance of the robot. In this phase, the robot already knows the rough task plan, so it can actively follow the pre-learned trajectories. The operator can arbitrarily modify the execution speed by simply pushing the robot along the demonstrated trajectory. Moreover, it can drive the robot forward and backward, and incrementally modify only those parts of the trajectory that need the refinement. During this phase, the robot estimates also the interaction forces and environmental compliance, which is needed for a robust and stable accomplished of assembly tasks in the exploitation phase. The benefit of this framework is in improved learning efficiency since the operator can concentrate only on the fine adjustment of the pre-learned trajectory. The robot optimizes its configuration from the data obtained in the pre-learning phase, which substantially facilitates the learning of kinematic redundant mechanisms and learning of bi-manual robot mechanisms. The proposed scheme was validated in a task where a bi-manual robot composed of two Kuka LWR-4 robot arms performs an assembly task.**

I. Introduction

Fast deployment of robot programs is one of the challenges of contemporary robotics, which addresses both robots in production lines as well as the service and humanoid robots in our home environments [1]. The ultimate goal is to enable an inexperienced operator to efficiently generate complex robot actions in a human-friendly way. One of the most successful paradigms towards this goal is the Programming by Demonstration (PbD), referred to also as the imitation learning [2]. It involves many aspects beyond a simple copying of human demonstrated motions, such as context understanding, generalization to different contexts, capturing and interpretation of interaction forces and compliance, etc. [3], [4]. Due to its simplicity, predictability, and ability to learn difficult dynamic tasks in reasonable amounts of time, this is a promising route to equip robots with necessary functionalities for performing a large variety of tasks, ranging from the classical industrial production tasks to the applications in households, stores, hospitals, etc. This is why PbD is still one of the most intense research topics in the field of robotics.

Our research focuses on PbD of assembly operations, where the robot passes from an unrestricted movement to the movement in a contact with the environment. For the stable operation, the robot should fulfill position and force constraints arising from the environment [5]. The impedance control is generally accepted paradigm for providing intrinsically safe robot actions while interacting with the environment [6]. However, it requires at least an approximate model of the environment in order to appropriately set the control parameters. In order to accomplish an assembly operation, we have to define robot poses, forces and torques and determine the required impedance parameters. During the kinesthetic guidance, forces and torques can be accurately obtained by reading robot sensors, while the determination of the impedance parameters is more challenging. The problem with kinesthetic guidance is how to accurately demonstrate the required task since the operator has to manually guide gravity compensated robot arm to perform the desired task while avoiding singularities, joint limits and resolving kinematic redundancies. Even with multiple demonstrations, it is not trivial to accurately define complicated motion parameters at the required speed. Some of the above-mentioned problems can be solved from multiple visual observations of humans performing the desired task. After capturing the human motion, the robot builds the model of the task and optimizes its motion taking into account the joint limits, singular configurations, and kinematic redundancies [7]. However, the problem of how to model the task with the desired accuracy required for the assembly operations and how to capture interaction forces and torques beside the task impedance parameters remains. By joining both approaches, we can benefit from advantages of both worlds, which are: the ability to globally model and optimize the task from visual observations and the ability to precisely capture the motion and force/torque parameters during the kinesthetic guidance. This joined approach was applied in the work of Lee and Ott [8], where the robot incrementally refines the initially demonstrated motion by kinesthetic guidance using the concept of the refinement motion tube [9]. The motion tube determines the range of allowed deviations from the nominal path within which the operator can modify the path. From previous demonstrations, they calculated the covariance matrix of the spatial part of the nominal trajectory and associated it with the robot controller impedance parameters. The Gaussian mixture model (GMM)

Humanoid & Cognitive Robotics Lab, Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia, [1]RoboLab, Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, Slovenia, (bojan.nemec, leon.zlajpah,ales.ude)@ijs.si, (sebastjan.slajpah, jozica.piskur)@robo.fe.uni-lj.si

and the Gaussian process regression (GPR) was used to decouple the spatial part from the temporal part of the trajectory. The idea of the refinement tube was used also for the bi-manual human-robot cooperation scheme, where the Dynamic motion primitives (DMP) were used to encode the nominal trajectory [10]. Motion refinement tube was defined regarding the Frenet-Serret (FS) frame and the decoupling from the synchronization between the spatial and temporal part of the trajectory was accomplished by projecting the actual velocity to the FS frame and adjusting the DMP speed scale factor.

In this work, we apply previously presented framework to robot learning and we propose modifications, that allow an efficient and simple PbD of assembly tasks. We demonstrate the benefits of the proposed approach on a representative bi-manual assembly operation.

The paper is organized into 6 sections. In the next section, we briefly describe learning from observation (LfO) framework, where we generate nominal bi-manual trajectory from multiple demonstrations and encode it with DMPs. Our approach to the refinement of the nominal trajectory with kinesthetic guidance is presented in section III. In section IV we tackle few issues related to the bi-manual execution of assembly tasks. The whole framework was experimentally verified in a generic bi-manual assembly task, described in section V. Final conclusions and possible future extensions are discussed in section VI.

II. Learning from Observation

In this section, we describe the first phase of the overall PbD procedure. The movement of both hands of the demonstrator during the execution of the desired task was captured by Kinect motion tracker, as shown in Fig. 1. Next, joint angles $\Theta_{i,j,k} \in \mathbb{R}^{4\times 2\times k}$ were mapped to the

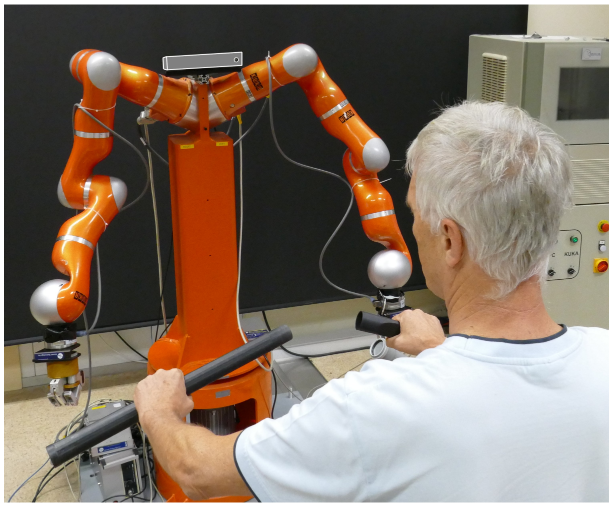

Fig. 1. The robot observes human wile performing a bi-manual operation.

Cartesian coordinates $\begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix}_{j,k} \in \mathbb{R}^{7\times 2\times k}$ expressed in a common base coordinate system using the human kinematic model [11], where index i denotes the joint of the human arm, $j = \{1, 2\}$ the left or the right arm, and index k the time sample. Vector $\mathbf{p} \in \mathbb{R}^3$ denotes the position vector and $\mathbf{q} \in \mathbf{S}^3$ describes the orientation quaternion. Cartesian positions of the right and the left robot were further mapped to the relative and the absolute coordinates of the bi-manual robot using the following transformations

$$\begin{aligned}
\mathbf{p}_a &= \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2), && (1)\\
\mathbf{q}_a &= \mathbf{q}_1 * \mathbf{q}_{\mathbf{k}_{21}}(\vartheta_{21}/2), && (2)\\
(0, \mathbf{p}_r) &= \bar{\mathbf{q}}_a * (0, \mathbf{p}_2 - \mathbf{p}_1) * \mathbf{q}_a && (3)\\
\mathbf{q}_r &= \bar{\mathbf{q}}_1 * \mathbf{q}_2, && (4)
\end{aligned}$$

where $\mathbf{p}_a, \mathbf{p}_r \in \mathbb{R}^3$ are the absolute and relative position vectors and $\mathbf{q}_a,\ \mathbf{q}_r \in S^3$ are the absolute and relative orientation quaternions. $\bar{\mathbf{q}}$ denotes conjugate quaternion and operator $*$ denotes quaternion product. $\mathbf{q}_{\mathbf{k}_{21}}$ is the unit quaternion calculated as

$$\mathbf{q}_{\mathbf{k}_{21}} = \left(\cos\left(\frac{\vartheta_{21}}{4}\right), \mathbf{k}_{21}\sin\left(\frac{\vartheta_{21}}{4}\right)\right), \qquad (5)$$

where $\mathbf{k}_{21}$ and ϑ_{21} are the axis and angle that realize the rotation $\mathbf{q}_1$ to $\mathbf{q}_2$.

Our framework applies the control in the task coordinates. This is necessary because we have to provide the desired robot compliance in task coordinates, which is more appropriate for performing precise assembly operations. Therefore, the singularity and joint limit avoidance will be performed in real time during the task execution, exploiting the kinematic redundancy of bi-manual robot. However, it is very important to choose an appropriate initial robot configuration, which assures that the local optimization will result in a feasible trajectory. The appropriate initial robot configuration is determined iteratively by a global optimization procedure [12].

We encode absolute and relative coordinates (1)-(4) in a more compact parametric representation, which allows us to store lengthy demonstration with a limited set of parameters. For this, we rely on the motion representation with dynamical motion primitives (DMPs) [13], extended for Cartesian space movements [14]. For the accomplishment of our PbD framework, we need to variate the speed of movement in a non-uniform way without changing the course of the movement. A suitable representation is Speed-Scaled Dynamic Motion Primitives (SS-DMPs), which we originally developed in [15]. Positions $\mathbf{p}$ and orientations $\mathbf{q}$ are generated by the following system of nonlinear differential equations [1]

$$\begin{aligned}
\nu(x)\tau\dot{\mathbf{z}} &= \alpha_z(\beta_z(\boldsymbol{g}_p - \mathbf{p}) - \boldsymbol{z}) + \mathbf{f}_p(x), && (6)\\
\nu(x)\tau\dot{\mathbf{p}} &= \boldsymbol{z}, && (7)\\
\nu(x)\tau\dot{\boldsymbol{\eta}} &= \alpha_z\left(\beta_z 2\log\left(\boldsymbol{g}_o * \overline{\mathbf{q}}\right) - \boldsymbol{\eta}\right) + \mathbf{f}_o(x), && (8)\\
\nu(x)\tau\dot{\mathbf{q}} &= \frac{1}{2}\boldsymbol{\eta} * \mathbf{q}, && (9)\\
\nu(x)\tau\dot{x} &= -\alpha_x x. && (10)
\end{aligned}$$

where x is the phase variable and $\boldsymbol{z}$ and $\boldsymbol{\eta}$ are auxiliary variables. The system (6) – (10) converges to the unique

[1]for the sake of simplicity we omit the subscripts $(.)_a$ and $(.)_r$

equilibrium point at $\mathbf{p} = \boldsymbol{g}_p$, $\boldsymbol{z} = 0$, $\mathbf{q} = \boldsymbol{g}_o$, $\boldsymbol{\eta} = 0$, and $x = 0$ providing properly chosen α_z and β_z [14]. By setting $\alpha_z = 4\beta_z > 0$ and $\alpha_x > 0$, the underlying second order linear dynamic system becomes critically damped.

The quaternion logarithm $log : \mathbf{S}^3 \mapsto \mathbb{R}^3$ defined as

$$\log(\mathbf{q}) = \log(v, \mathbf{u}) = \begin{cases} \arccos(v)\dfrac{\mathbf{u}}{\|\mathbf{u}\|}, \ \mathbf{u} \neq 0 \\ [0,0,0]^{\mathrm{T}}, \ \text{otherwise} \end{cases} . \quad (11)$$

maps the quaternion describing the rotation between the goal and current pose to the rotation error vector. Its inverse transformation is defined as

$$\exp(\mathbf{r}) = \begin{cases} \cos\left(\|\mathbf{r}\|\right) + \sin\left(\|\mathbf{r}\|\right)\dfrac{\mathbf{r}}{\|\mathbf{r}\|}, \ \mathbf{r} \neq 0 \\ 1 + [0,0,0]^{\mathrm{T}}, \ \text{otherwise.} \end{cases} \quad (12)$$

The nonlinear forcing terms $\mathbf{f}_p(x)$ and $\mathbf{f}_o(x)$ are formed in such a way that the response of the second-order differential equation system (6) – (10) can approximate any smooth point-to-point trajectory from the initial position $\boldsymbol{p}_0$ and orientation $\boldsymbol{q}_0$ to the final position $\boldsymbol{g}_p$ and orientation $\boldsymbol{g}_o$. They are defined as linear combinations of radial basis functions (RBFs)

$$\mathbf{f}_p(x) = \frac{\sum_{i=1}^{N} \mathbf{w}_{i,p}\Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \quad (13)$$

$$\mathbf{f}_o(x) = \frac{\sum_{i=1}^{N} \mathbf{w}_{i,o}\Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \quad (14)$$

$$\Psi_i(x) = \exp\left(-h_i\left(x - c_i\right)^2\right), \quad (15)$$

where free parameters $\mathbf{w}_{i,p}$, $\mathbf{w}_{i,o}$ determine the shape of the position and the orientation trajectory. Centers c_i of RBFs with widths h_i are evenly distributed along the trajectory.

The temporal scaling function $\nu(x)$ determines variations form the demonstrated speed profile and allows to specify non-uniform speed changes along the demonstrated trajectory. Similarly to the forcing terms (13) and (14), it is encoded as a linear combination of RBFs

$$\nu(x) = \frac{\sum_{j=1}^{M} v_j \Psi_j(x)}{\sum_{j=1}^{M} \Psi_j(x)}, \quad (16)$$

where v_j are the corresponding weights.

In order to parameterize the demonstrated control policy with a DMP, the weights $\boldsymbol{w}_{i,p}$ $\boldsymbol{w}_{i,o}$ and v_j need to be calculated. The shape weights $\boldsymbol{w}_{i,p}$ and $\boldsymbol{w}_{i,o}$ are calculated by applying standard regression techniques [14] using the demonstrated trajectory, which was previously transformed to absolute and relative coordinates. For ν we initially set such v_j that $\nu(x) = 1 \ \forall x$, meaning that the demonstrated speed profile remains as demonstrated.

III. Incremental Task Refinement by Kinestetic Guidance

Kinesthetic guidance involves guiding the robot in the active gravity compensation mode [16]. The essence of our approach is to allow a free motion of the robot backward and forward along the nominal trajectory and to modify only a chosen part of the trajectory. The motion back and forward along the nominal trajectory can be easily performed applying DMP framework and changing the speed scaling factor ν. Namely, when $\nu < 0$ the robot moves backward along the nominal trajectory[2], stands steel when $\nu = 0$, and moves forward along the nominal trajectory when $\nu > 0$. The most intuitive guidance is to apply the force in the desired direction of the motion along the nominal trajectory. Let assume that our robot is impedance controlled in Cartesian coordinates, where $\mathbf{K} \in \mathbb{R}^{6\times6}$ and $\mathbf{D} \in \mathbb{R}^{6\times6}$ are the proportional and derivative controller gain matrices, not necessarily diagonal. Then, the Cartesian force and torque applied to the robot end-effector can be calculated simply by

$$\begin{aligned} \begin{bmatrix} \mathbf{F}(x) \\ \mathbf{M}(x) \end{bmatrix} &= \mathbf{K}\begin{bmatrix} \mathbf{p}_n(x) - \mathbf{p}(x) \\ \log(\mathbf{q}_n(x) * \bar{\mathbf{q}}(x)) \end{bmatrix} + \\ & \quad \mathbf{D}\begin{bmatrix} \dot{\mathbf{p}}_n(x) - \dot{\mathbf{p}}(x) \\ \boldsymbol{\omega}_n(x) - \boldsymbol{\omega}(x) \end{bmatrix}, \end{aligned} \quad (17)$$

where $\boldsymbol{\omega}$ denotes angular velocity, calculated from two subsequent quaternions. Subscript $(.)_n$ denotes nominal trajectory, obtained with LfO as described in section II. Now, define the direction of the translational motion as the tangential axis of the Frenet-Serret frame [17], (see Fig. 2), $\mathbf{t}(x) = \frac{\dot{\mathbf{p}}_n(x)}{\|\dot{\mathbf{p}}_n(x)\|}$, and the robot tool rotation vector $\mathbf{s}(x) = \frac{\boldsymbol{\omega}_n(x)}{\|\boldsymbol{\omega}_n(x)\|}$, where $\boldsymbol{\omega}$ are current robot commanded velocities. Speed scaling factor can be then determined as

$$\nu(x) = k_1(\mathbf{F}(x) \cdot \mathbf{t}(x)) + k_2(\mathbf{M}(x) \cdot \mathbf{s}(x)), \quad (18)$$

where $(\cdot)$ denotes dot product and k_1, k_2 are positive scalars, which scale the velocity of the motion along the nominal trajectory. If we apply this ν to the set of equations (6–10), the robot moves in the tube along the nominal trajectory, as illustrated in Fig. 2. The axes of the ellipse, which define the refinement tube, depend only on applied force/torque and the robot impedance along the normal $\mathbf{n}$ and binormal $\mathbf{b}$ axes of the Frennet-Serret frame. New robot positions and orientations have to be sampled at exactly the same phase as the nominal trajectory and saved as new modified trajectory. In the work of Lee and Ott [8] and in our previous work [10] the robot impedance was calculated according to the covariance matrix that describes the expected deviations around the nominal trajectory. In this work, we relax this requirement and keep impedance (defined with the robot control gains) fixed during the kinestetic guidance. It might happen that this fixed impedance does not allow to modify the trajectory for the desired displacement. We override this limitation by letting the user modify the nominal trajectory in multiple passes. One can simply push the robot back and at the next change of the motion direction, new nominal

[2]Note that DMP is a dynamical system, which becomes unstable for negative ν. In this case we apply another DMP, learned for the time reversed trajectory $\mathbf{p}(t) = \mathbf{p}(\tau - t)$ and $\mathbf{q}(t) = \mathbf{q}(\tau - t)$

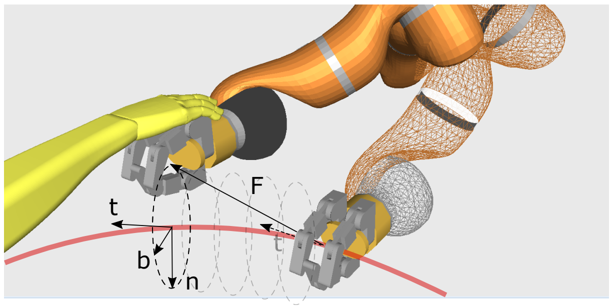

Fig. 2. The figure illustrates the refinement tube (denoted with ellipses) during the kinestetic guidance around the nominal trajectory (denoted with red curve). Applied force moves the robot along the nominal trajectory, while the displacement depends on forces orthogonal to $\mathbf{t}$ and robot stiffness in these directions

trajectory is calculated as

$$\begin{aligned}\mathbf{p}_{n,l+1}(x) &= \zeta\Delta\mathbf{p}(x) + \mathbf{p}_{n,l}(x), &(19)\\ \Delta\mathbf{p}(x) &= \mathbf{p}_{n,l}(x) - \mathbf{p}(x),\\ \mathbf{q}_{n,l+1}(x) &= \exp\left(\zeta\frac{\boldsymbol{\omega}(x)}{2}\right) * \mathbf{q}_{n,l}(x), &(20)\\ \boldsymbol{\omega}(x) &= 2\log(\mathbf{q}_{n,l}(x)) * \overline{\mathbf{q}}(x),\end{aligned}$$

where subscript l denotes the learning cycle index and $0 \le \zeta \le 1$ is the weighting factor that defines the update rate. If we set $\zeta = 1$, new nominal trajectory $\mathbf{p}_{n,l+1}$ is equal to the measured trajectory $\mathbf{p}$. On the other hand, if we set $\zeta = 0$, the nominal trajectory $\mathbf{p}_{n,l+1}$ does not change. After each learning cycle as well as after each motion direction change, the updated nominal trajectory $\mathbf{p}_{n,l+1}$ is encoded into SS-DMP. It is used as a command trajectory to control the robot in the next cycle. The cycle counter is incremented whenever the operator changes the direction along the nominal trajectory. The number of cycles in not limited, we can modify any part of the trajectory and the length of the modified trajectory can vary freely from cycle to cycle, which makes this type of learning very intuitive and efficient at the same time.

The above procedure describes the task refinement for a single robot arm. For the dual-arm implementation, we have to assure that both robots are using a common speed scaling factor $\nu(x)$. This means, that when we move one robot along the nominal trajectory, the other has moved as well. The trajectory adjustment is performed regarding the nominal trajectory of each robot independently and not in absolute and relative coordinates because this is more intuitive for the user. Task coordinates of each robot are obtained from the absolute and relative coordinates by inverting relations (1–4). Note that this transformation is unique and always exits. After calculating the update according to (19–21) for each robot arm independently, we calculate the new nominal absolute and relative trajectory at phase x using (1 – 4). In this way, we obtain a very intuitive way of changing absolute and relative part of the task. If we move just one robot or when we move both robots in opposite directions, we modify relative position coordinates. When we move both robot arms in the same direction, we modify absolute positions. Similarly, we modify the absolute and relative orientations. Please note that this refers to the moving of robot orthogonal to the tangential vector $\mathbf{t}$, since movement along the $\mathbf{t}$ immediately changes the phase (10).

The last phase of learning is actual speed learning. Once we are satisfied with the trajectory, we push the robot from the start to the trajectory end with the desired speed, capture $\nu(x)$ and calculate weights v_j, which encode $\nu(x)$ as a linear combination of the RBFs (16). Note that nothing restricts this motion to be only in the direction of the nominal trajectory. We can freely demonstrate such $\nu(x)$ which goes forward and back along the nominal trajectory. In this way, we can efficiently program quasi-periodic motions and encode tasks like sawing, polishing, wiping, material tooling, etc. with a minimal number of parameters.

A. Determination of the Task Constraints

Task constraints relate to the forces acting on each robot as a consequence of a bi-manual operation and interaction with the environment. We assume that the forces and torques can be measured with a force/torque sensor, usually mounted on the robot wrist. Force readings, mapped to the tool center point of the robot and expressed in the common base, are denoted as $\tilde{\mathbf{F}}_1, \tilde{\mathbf{M}}_1, \tilde{\mathbf{F}}_2, \tilde{\mathbf{M}}_2$ for the first and the second robot, respectively. The corresponding forces and torques expressed in the absolute and relative coordinates are

$$\begin{aligned}(0, \mathbf{F}_a) &= \bar{\mathbf{q}}_1 * (0, \tilde{\mathbf{F}}_1) * \mathbf{q}_1 + \bar{\mathbf{q}}_2 * (0, \tilde{\mathbf{F}}_2) * \mathbf{q}_2, &(21)\\ (0, \mathbf{M}_a) &= \bar{\mathbf{q}}_1 * (0, \tilde{\mathbf{M}}_1) * \mathbf{q}_1 + \bar{\mathbf{q}}_2 * (0, \tilde{\mathbf{M}}_2) * \mathbf{q}_2, &(22)\\ (0, \mathbf{F}_r) &= \bar{\mathbf{q}}_a * (0, \frac{\tilde{\mathbf{F}}_1 - \tilde{\mathbf{F}}_2}{2}) * \mathbf{q}_a, &(23)\\ (0, \mathbf{M}_r) &= \bar{\mathbf{q}}_a * (0, \frac{\tilde{\mathbf{M}}_1 - \tilde{\mathbf{M}}_2}{2}) * \mathbf{q}_a, &(24)\end{aligned}$$

We assume that the robot is interacting with the environment only through the manipulated object. Our goal is to determine the effective compliance, which unifies tool and environment compliances, expressed in absolute and relative coordinates of a bi-manual robot, as illustrated in Fig. 3. In this work we assume a simple model of the effective positional compliance [18]

$$\mathbf{F}_a = \boldsymbol{\Sigma}_a(\mathbf{p}_a - \mathbf{p}_{0,a}), \quad \mathbf{F}_r = \boldsymbol{\Sigma}_r(\mathbf{p}_r - \mathbf{p}_{0,r}), \qquad (25)$$

where $\mathbf{p}_{0,a}$ and $\mathbf{p}_{0,r}$ denotes the estimated contact position [19] in absolute and relative coordinates, respectively. $\boldsymbol{\Sigma}_a \in \mathbb{R}^{3\times3}$ and $\boldsymbol{\Sigma}_r \in \mathbb{R}^{3\times3}$ are the effective absolute and relative diagonal stiffness matrices. Note that the compliance in absolute and relative coordinates are robot configuration dependent. For an example, consider the situation shown in Fig. 3. When both tubes are aligned, the relative motion along the tube (denoted with relative y coordinate) is enabled, constrained only by the friction force. According to the model (25), $\Sigma_{r,y}$ changes and becomes low. Obviously, we can model the effective compliance as a phase dependent parameter. It can be estimated from the past L sampled data using weighted least squares, where we assign lower weights

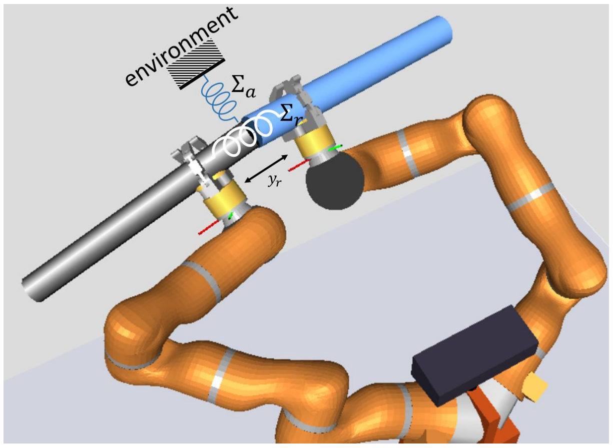

Fig. 3. The relative and absolute effective compliance schematically represented as springs. Relative compliance equals to the manipulated object, while the absolute one is composed of the environment and object compliance.

to the past data samples. i-th component can be estimated form [3]

$$\mathbf{X}=\begin{bmatrix} p_{r,i}(x)-p_{0,r,i} \\ p_{r,i}(x_1)-p_{0,r,i} \\ \cdots \\ p_{r,i}(x_L)-p_{0,r,i}\end{bmatrix}, \mathbf{Y}=\begin{bmatrix} f_{r,i}(x) \\ f_{r,i}(x_1) \\ \cdots \\ f_{r,i}(x_L)\end{bmatrix}$$

$$\begin{bmatrix}\Sigma_{r,p,i}(x)\end{bmatrix}=(\mathbf{X}^T\mathbf{W}_s\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}_s\mathbf{Y},$$

where x denotes the current phase, x_L phase at L-th previous sample and $\mathbf{W}_s\in\mathbb{R}^{L\times L}$ is the diagonal matrix with variable weights $w_s(x_i)=\alpha_s w_s(x_{i-1})$, $\alpha_s<1$. The estimated environment and the tool compliance parameters together with the measured absolute and relative forces and torques are encoded with RBFs after the accomplishment incremental task refinement. The will be used to adapt the impedance control law, described in the next section.

IV. Impedance Control of Bi-manual Assembly Tasks

Impedance control is very appropriate for the assembly operations, as it provides stable operations in contact and not contact motions and transitions between the two [20]. Here, we briefly present impedance control for a bi-manual robot, which provides tracking of the commanded control variables obtained by the proposed kinestetic guidance. Commanded $\mathbf{p}_{a,d}$, $\mathbf{q}_{a,d}$, $\mathbf{p}_{r,d}$, $\mathbf{q}_{r,d}$, $\mathbf{F}_{a,d}$, $\mathbf{M}_{a,d}$, $\mathbf{F}_{r,d}$ and $\mathbf{M}_{r,d}$ are in our scheme obtained with DMP and RBF integration as described in section II

For a bi-manual robot control, we have to map the nominal relative and absolute task coordinates to the corresponding joint coordinates of both robots, denoted with $\boldsymbol{\theta}=[\boldsymbol{\theta}_1\ \boldsymbol{\theta}_2]^T\in\mathbb{R}^{(N_1+N_2)}$, where N_1 and N_2 are the numbers of joints of the first and the second robot, respectively. This transformation is obtained through relative and absolute geometrical Jacobian, which maps the corresponding translational and angular velocities to the joint velocities

$$\begin{bmatrix}\dot{\mathbf{p}}_r\\ \boldsymbol{\omega}_r\end{bmatrix}=\mathbf{J}_r\dot{\boldsymbol{\theta}}\ ,\quad \begin{bmatrix}\dot{\mathbf{p}}_a\\ \boldsymbol{\omega}_a\end{bmatrix}=\mathbf{J}_a\dot{\boldsymbol{\theta}}. \tag{26}$$

Absolute Jacobian is obtained from the time derivative of (1–4), yielding [21]

$$\mathbf{J}_a=\begin{bmatrix}\frac{1}{2}\mathbf{J}_1 & \frac{1}{2}\mathbf{J}_2\end{bmatrix} \tag{27}$$

$$\mathbf{J}_r=\begin{bmatrix}\mathbf{R}_a{}^T & 0\\ 0 & \mathbf{R}_a{}^T\end{bmatrix}\begin{bmatrix}-(\mathbf{J}_{1,p}+\Lambda\frac{\mathbf{J}_{1,\omega}}{2}) & \mathbf{J}_{2,p}+\Lambda\frac{\mathbf{J}_{2,\omega}}{2})\\ -\mathbf{J}_{1,\omega} & \mathbf{J}_{2,\omega}\end{bmatrix},$$

where $\Lambda=\mathbf{S}^T(\mathbf{p}_2-\mathbf{p}_1)$ and $\mathbf{R}_a$ is the rotation matrix representation of the quaternion $\mathbf{q}_a$. $\mathbf{J}_{i,p}$ and $\mathbf{J}_{i,\boldsymbol{\omega}}$ denote the positional and rotational part of the geometrical Jacobian of the i-th arm, respectively. $\mathbf{S}$ represents the well know skew-symmetric matrix.

To control both absolute and relative coordinates, we define extended task coordinates $\mathbf{x}_e=[\mathbf{p}_a^T\ \mathbf{q}_a^T\ \mathbf{p}_r^T\ \mathbf{q}_r^T]^T$ and extended velocities $\mathbf{v}_e=[\dot{\mathbf{p}}_a^T\ \boldsymbol{\omega}_a^T\ \dot{\mathbf{p}}_r^T\ \boldsymbol{\omega}_r^T]^T$ and extended Jacobian $\mathbf{J}_e=[\mathbf{J}_a^T\ \mathbf{J}_r^T]^T$. The impedance control of a bi-manual robot is accomplished with

$$\rho=\mathbf{H}_e\mathbf{J}_e^+(\ddot{\mathbf{x}}_c-\dot{\mathbf{J}}_e\dot{\boldsymbol{\theta}})+\mathbf{h}_e+\mathbf{H}_e\mathbf{N}_e\xi+\mathbf{J}_e\mathbf{F}_e, \tag{28}$$
$$\dot{\mathbf{v}}_c=\dot{\mathbf{v}}_n+\mathbf{D}\dot{\mathbf{e}}_e+\mathbf{K}\mathbf{e}_e+\mathbf{K}_f(\mathbf{F}_{e,d}-\mathbf{F}_e), \tag{29}$$
$$\xi=\mathbf{K}_n(\dot{\boldsymbol{\theta}}_0-\dot{\boldsymbol{\theta}}) \tag{30}$$

where $\rho\in\mathbb{R}^{N_1+N_2}$ are commanded motor torques, $\mathbf{H}_e\in\mathbb{R}^{(N_1+N_2)\times(N_1+N_2)}=\begin{bmatrix}\mathbf{H}_1 & 0\\ 0 & \mathbf{H}_2\end{bmatrix}$ is the extended block diagonal inertia matrix, $\mathbf{H}_1$ and $\mathbf{H}_2$ are inertia matrices of the first and the second robot, $\mathbf{h}_e\in\mathbb{R}^{N_1+N_2}$ are Coriolis, centrifugal and gravity forces, $\mathbf{N}_e\in\mathbb{R}^{(N_1+N_2)\times(N_1+N_2)}$ is the null space projection matrix and $\mathbf{F}_e\in\mathbb{R}^{12}=[\mathbf{F}_a^T\ \mathbf{M}_a^T\ \mathbf{F}_r^T\ \mathbf{M}_r^T]^T$ is the vector of absolute and relative forces and torques. $\mathbf{F}_e$ actually describes the coupling forces of both manipulators when acting on the same object. $\mathbf{K},\mathbf{D}\in\mathbb{R}^{12\times12}$ are positive definite block diagonal matrices of proportional and derivative gains, respectively[4], and $\mathbf{K}_n\in\mathbb{R}^{(N_1+N_2)\times(N_1+N_2)}$ is the positive definite diagonal matrix containing gains of the simplified null-space velocity controller [22] and $\mathbf{K}_f\in\mathbb{R}^{12\times12}$ is the force feedback gain matrix. $\mathbf{J}_e^+$ is the Moore-Penrose pseudo-inverse of the extended Jacobian $\mathbf{J}_e$, $\mathbf{e}_e\in\mathbb{R}^{12}$ is the error between the nominal and actual measured extended task coordinates, calculated as

$$\mathbf{e_e}=\begin{bmatrix}\mathbf{p}_{a,n}-\mathbf{p}_a\\ \log(\mathbf{q}_{a,n}*\bar{\mathbf{q}}_a)\\ \mathbf{p}_{r,n}-\mathbf{p}_r\\ \log(\mathbf{q}_{r,n}*\bar{\mathbf{q}}_r)\end{bmatrix}, \tag{31}$$

$\dot{\mathbf{v}}_n\in\mathbb{R}^{12}$ are the nominal (desired) extended accelerations and $\dot{\mathbf{e}}_e\in\mathbb{R}^{12}$ are errors between the nominal and actual extended velocities. Vector $\dot{\boldsymbol{\theta}}_0\in\mathbb{R}^{(N_1+N_2)}$ is an arbitrary vector of joint velocities that is projected in the null-space

[3]For the sake of simplicity, the estimation procedure is explained only for the relative coordinates. Absolute are estimated in exactly the same way.

[4]composed of blocks of 3×3 matrices, which are not necessarily diagonal. Non-diagonal gain matrices enable to specify the direction of the robot stiffens and damping.

of the primary task. It is chosen in such a way that the robot avoids joint limits and singular configurations [23]. Note that the dimension of the extended task defined with $\mathbf{x}_e$ can be ≤ 12, which allows exploiting the additional degrees-of-redundancy for the secondary task(s). In many cases, particularly for the bi-manual assembly, absolute coordinates might not be relevant. In such a case, the robot exploits the additional degrees-of-redundancy, which generally results in a more optimal policy in the configuration space. However, for the assembly, there are often situations where absolute coordinates are not important only for a certain part of the task. The problem is how to smoothly transit between the parts, where both absolute and relative coordinates are relevant and the parts, where only the relative coordinates are relevant. The appropriate solution was inspired by the work of Chiaverini et. al [24], where they introduced the user-defined accuracy based on the weighted damped least-squares. This allows discriminating between directions in the end-effector space where higher accuracy is desired and directions where lower accuracy can be tolerated. A similar procedure was applied to the absolute and relative coordinates by redefining the extended Jacobian as $\mathbf{J}_e = [\mathbf{W}_a\mathbf{J}_a \;\; \mathbf{J}_r]^T$ and calculating $\mathbf{J}_e^+$ as the inertia weighted damped pseudo-inverse of the Jacobian in the form

$$\mathbf{J}_e^+ = \mathbf{H}_e^{-1}\mathbf{J}_e^T(\mathbf{J}_e\mathbf{H}_e^{-1}\mathbf{J}_e^T + \mathbf{I}\lambda)^{-1}, \qquad (32)$$

where $\mathbf{W}_a \in \mathbb{R}^{6\times 6}$ is a diagonal weighting matrix with weights w_a defining the importance of the corresponding part of the absolute task, $\mathbf{I} \in \mathbb{R}^{12\times 12}$ is the identity matrix and λ is a positive scalar close to 0, which prevents to generate high velocities in near-singular configurations. Setting all weights in $\mathbf{W}_a$ to zero results in the execution of the relative task only. Note that very low values of weights are needed to enable execution of the absolute task also. A numerical problem might arise at weights values close to zero. We achieved numerical stability by weighting also the absolute part of the tracking error in (31) with w_a.

During the assembly process, the robot should automatically adjust its compliance according to the task constraints gathered during PbD. Let consider a simple one-dimensional case as shown in Fig. 4. From the force equilibrium we can

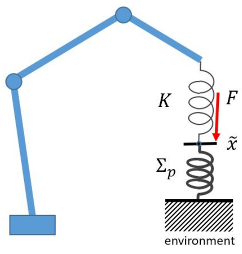

Fig. 4. A simple model of the robot interacting with the environment. F is the disturbance force acting at the robot end effector, $\tilde{x}$ is the displacement due to this force, and K and Σ_p are the robots and the environment stiffness, respectively.

get

$$(K + \Sigma_p) = \frac{F}{\tilde{x}} = \kappa. \qquad (33)$$

The term κ denotes target stiffness, which defines how much will the robot deviate from the commanded position when we apply a disturbance force F. This target stiffness is a design parameter, that we will set and keep constant for the entire task. It enables to set the controller parameters $\mathbf{K}$ from the estimated environment stiffness as

$$K_{a,i} = \kappa - \Sigma_{a,i}$$
$$K_{r,i} = \kappa - \Sigma_{r,i},$$

assuming that the estimated environment stiffness is below the desired target stiffness. If this is not the case, Eq. 34 results in negative gains K; we set the $K_{a,i}$, $K_{r,i}$ to a small positive value making the robot very complaint. According to this simple method, the robot will be compliant in the directions constrained by the environment and vice versa, stiff in unconstrained directions.

V. Experimental Verification with a Generic Bi-manual Assembly Operation

The proposed framework was verified on a bi-manual setup consisting of 1 d.o.f torso, two Kuka LWR-4 7 d.o.f robot arms equipped with Barret hands and ATI force-torque sensors and Kinect-2 RGBD camera, as seen in the Fig. 5. The control algorithms were implemented on xPCTarget Simulink running at a frequency of 2KHz, which communicated with the LWR-4 control computer at 500Hz via Fast Research Interface. The kinesthetic learning algorithms were implemented in Matlab at 50Hz. For the performance

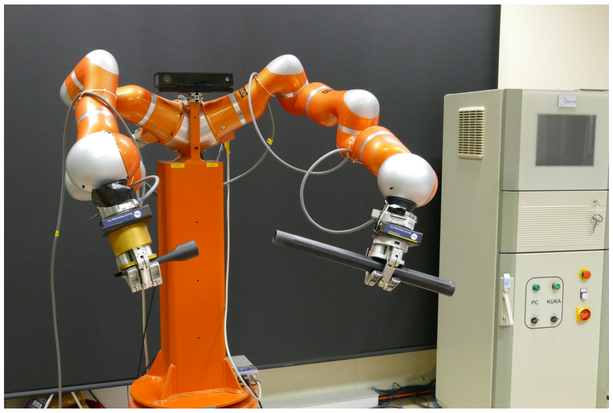

Fig. 5. Experimental setup

evaluation, we selected a generic assembly task, composed of two Peg-In-Hole (PiH) operations, one in relative and the other in absolute coordinates. The robot had to assemble two parts of the vacuum cleaner tube and release it on the pedestal. The initial task demonstration was performed by a human, where the joint angles of both human arms were tracked with Kinect-2. Joint angles were mapped to the Cartesian coordinates and further to the absolute and relative task of the bi-manual setup, as explained in section II. As the Kinect tracker is not capable to track the demonstrator's motion with the precision required for the assembly operation, the demonstrated policy needed additional refinement.

For this, we applied the framework presented in section III, where we iteratively refined primarily both PiH operations. The scalar ζ in Eq. (19), (21) was set to 0.8 during the refinement. The robot stiffness during the task refinement was set to $500N/m$ and $300Nm/rd$ for positional and rotational task coordinates of both robot arms, respectively. The last procedure of the refinement was the demonstration of the desired velocity profile. In this phase, we were pushing the left robot arm along the refined nominal trajectory in the relative and absolute coordinates with the desired velocity profile, shown in Fig. 6. In this phase, we estimated also the effective environment stiffness, as described in subsection III-A. The estimated effective stiffness parameters during the relative peg-in-hole part of the task are shown in Fig. 7. Note that it is important to push at the robot at the wrist and not at the robot end effector in order to get non-biased force reading of the ATI force sensors.

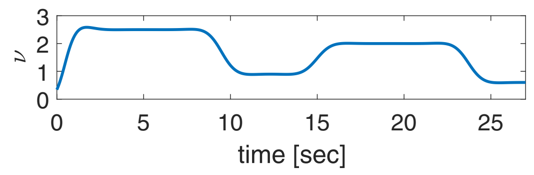

Fig. 6. Smoothed velocity scaling factor, which was demonstrated at the end of the incremental task refinement.

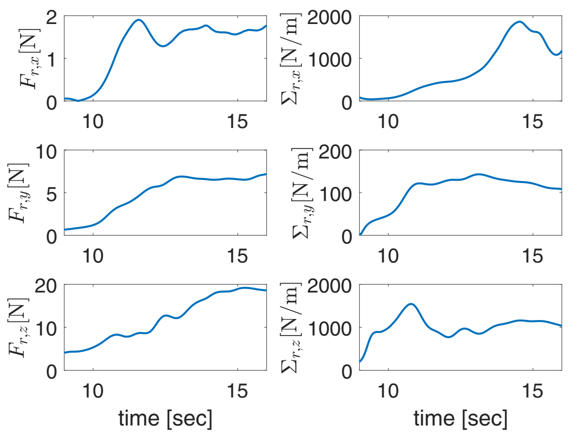

Fig. 7. Measure relative forces and estimated relative stiffness during the relative peg-in-hole operation. Relative direction y is in the direction of the peg insertion, where the robot estimated low stiffness. Note also that the initially estimated stiffness in x coordinate was also low, as the interaction forces in this direction were initially very low.

For playback, we switched to the bi-manual impedance control, as described in section IV. We set the target stiffness parameter κ to $1000N/m$ and limited the minimal robot stiffness to $100N/m$ for all positional coordinates. In this experiment, we adjusted only the positional part of the robot stiffness in relative and absolute coordinates according to (34). Initial environment stiffness was set to zero. Therefore, the robot was initially stiff in all coordinates, which assured the precise tracking of the demonstrated trajectory. During the PiH in relative coordinates, the robot lowered the stiffness to the minimal stiffness in x and z relative coordinates, while the stiffness along the axis of insertion y remained at a relatively high value of approx. $750N/m$. We set also the initial values of the $\mathbf{W}_a$ matrix to 0, which means, that the robot executed only the relative part of the task, while the redundancy resolution minimized kinetic energy during the motion. Therefore, the actual robot motion in absolute coordinates was different from the one demonstrated. Intuitively, we increased the weights of $\mathbf{W}_a$ prior an absolute force was detected, indicating that the robot interacts with the environment and that the absolute coordinates become important. The robot performed a smooth transition from the part of the task, where only the relative coordinates were relevant to the part of the task, where the robot precisely tracked also the absolute coordinates. This can be seen from the measured joint velocities of both robots, plotted in Fig. 8. From Fig. 6 we can see that in most parts of the trajectory we could increase the velocity of the task execution. It was necessary to decreased it only for the peg insertion in order to increase the reliability of the whole assembly task.

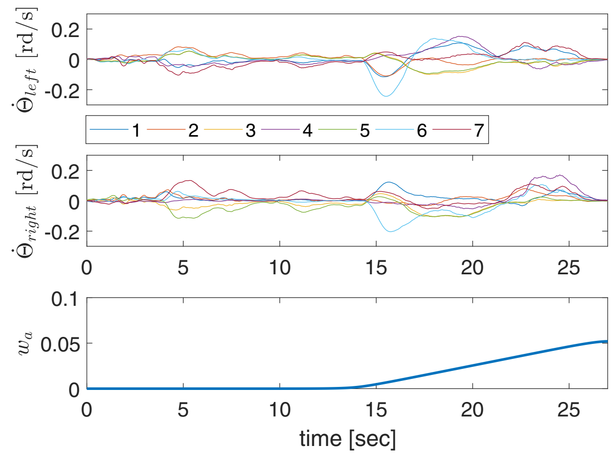

Fig. 8. Measured joint velocities of the left and the right robot, respectively. Note smooth joint velocities during the transition between the execution of the task using relative coordinates only to the task where both relative and absolute coordinates matter.

VI. Conclusions

In the paper, we presented an effective framework for the fast deployment of robot assembly tasks. It is particularly effective for bi-manual learning. However, it can be efficiently used also for single arm tasks, as substantially decrease the time for preparing a new robot application with less experienced robot operators. Therefore, we see possible applications of the proposed learning framework in low-batches SME production lines and for humanoid and service robots performing in our home environments. The framework exploits the impedance properties of the robot manipulator. It can be adapted also to the traditional stiff robot arms equipped with a force/torque sensor mounted in the robot wrist. However, the performance would be degraded while interacting with the stiff environment. Future work involves testing in more complex assembly tasks and implementation of the proposed framework for the full-sized humanoid robot Talos.

Acknowledgment

The research leading to these results has received funding from the Horizon 2020 FoF Research & Innovation Action no.723909, AUTOWARE. and by the Slovenian Research Agency grant J2-7360, "Learning and autonomous adaptation of dual arm assembly and service tasks".

References

[1] E. Dean-Leon, K. Ramirez-Amaro, F. Bergner, I. Dianov, P. Lanillos, and G. Cheng, "Robotic technologies for fast deployment of industrial robot systems," *IECON Proceedings (Industrial Electronics Conference)*, pp. 6900–6907, 2016.

[2] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, jun 2004.

[3] A. Billard and S. Calinon, "Chapter 59: Robot Programming by Demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008.

[4] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," *2011 IEEE International Conference on Robotics and Automation*, pp. 3828–3834, 2011.

[5] S. Chiaverini, B. Siciliano, and L. Villani, "A survey of robot interaction control schemes with experimental comparison," *IEEE/ASME Trans. on Mechatronics*, vol. 4, no. 3, pp. 273–285, 1999.

[6] N. Hogan and S. Buerger, "Impedance and Interaction Control," *Robotics and automation handbook*, pp. 19:1–24, 2005.

[7] A. Vakanski and F. Janabi-Sharifi, *Robot learning by visual observation*. John Wiley & Sons, 2017.

[8] D. Lee and C. Ott, "Incremental Motion Primitive Learning by Physical Coaching Using Impedance Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 4133–4140.

[9] ——, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, no. 2, pp. 115–131, Oct 2011.

[10] B. Nemec, N. Likar, A. Gams, and A. Ude, "Human robot cooperation with compliance adaptation along the motion trajectory," *Autonomous Robots*, vol. 42, no. 5, pp. 1023–1035, 2018.

[11] T. Petrič and L. Žlajpah, "Smooth Transition Between Tasks on a Kinematic Control Level: Application to Self Collision Avoidance for Two Kuka LWR Robots," in *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011, pp. 162–167.

[12] M. Do, P. Azad, T. Asfour, and R. Dillmann, "Imitation of Human Motion on a Humanoid Robot using Non-Linear Optimization," in *2008 8th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2008, pp. 545–552.

[13] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–73, 2013.

[14] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014, pp. 2997–3004.

[15] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, USA, 2013, pp. 423–428.

[16] K. Kronander and A. Billard, "Learning Compliant Manipulation through Kinesthetic and Tactile Human-Robot Interaction," *IEEE Transactions on Haptics*, vol. 7, no. 3, pp. 367 – 380, 2014.

[17] R. Ravani and A. Meghdari, "Velocity distribution profile for robot arm motion using rational Frenet-Serret curves," *Informatica*, vol. 17, no. 1, pp. 69–84, 2006.

[18] L. Rozo, S. Calinon, D. Caldwell, P. Jimnez, and C. Torras, "Learning collaborative impedance-based robot behaviors," in *AAAI'13 Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013, pp. 1422–1428.

[19] A. Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision Detection and Safe Reaction with the DLR-III Lightweight Manipulator Arm," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1623–1630, oct 2006.

[20] N. Hogan, "Stable Execution of Contact Tasks Using Impedance Control," in *1987 IEEE International Conference on Robotics and Automation*, 1987, pp. 1047–1054.

[21] B. Nemec, N. Likar, A. Gams, and A. Ude, "Adaptive human robot cooperation scheme for bimanual robots," in *Advances in Robot Kinematics 2016*, J. Lenarcic and J. Merlet, Eds. Springer Proceedings in Advanced Robotics, 2018, vol. 4.

[22] B. Nemec, L. Žlajpah, and D. Omrčen, "Comparison of null-space and minimal null-space control algorithms," *Robotica*, vol. 25, no. 05, pp. 511–520, feb 2007.

[23] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Trans. on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[24] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the Damped Least-Squares Inverse Kinematics with Experiments on an Industrial Robot Manipulator," *IEEE Transactions on Control Systems Technology*, vol. 2, no. 2, pp. 123–134, 1994.