# Robust Execution of Assembly Policies Using a Pose Invariant Task Representation

Bojan Nemec[1], Matevž Majcen Hrovat[1], Mihael Simonič[1], Suhan Shetty[2], Sylvain Calinon[2], and Aleš Ude[1]

***Abstract*— This paper discusses the robustness of executing robot tasks in contact with the environment. For example, in assembly, even the slightest error in the initial pose of the assembled object or grasp uncertainties can lead to large contact forces and, consequently, failure of the assembly operation. Force control can help to improve the robustness only to a certain extent. In this work, we propose using the position and orientation invariant task representation to increase the robustness of assembly and other tasks in continuous contact with the environment. We developed a variable compliance controller which constantly adapts the policy to environmental changes, such as positional and rotational displacements and deviations in the geometry of the assembled part. In addition, we combined ergodic control and vision processing to improve the detection of the assembled object's initial pose. The proposed framework has been experimentally validated in two challenging tasks; The first example is a mock-up of an assembly operation, where the object moves along a rigid wire, and the second is the insertion of a car light bayonet bulb into the housing.**

## I. Introduction

Assembly is one of the most common tasks in industrial robotics. However, assembly operations are not necessary only in industrial environments but also in our homes, as many of the daily tasks we perform are assembly tasks. One of the problems in assembly is accurate calibration. Consider, for example, the peg-in-a-hole (PiH) task depicted in Fig 1. Even the smallest errors in the orientation of the assembled object can result in large positional errors and, consequently, large forces during the assembly using a predefined assembly sequence. In automated robot assembly, complex calibration procedures and specialized hardware are usually applied to determine the position and orientation of workpieces. On the contrary, this is often hard to achieve in a domestic environment. The same applies to small series and craft production. Therefore, developing procedures that automatically adapt assembly tasks to environmental changes is one of the main challenges for the faster introduction of robotics in our homes and small-scale production [1].

This problem is not new and has been under investigation since the beginning of robotics. Early approaches tend to solve the problem by force control [2]. However, force control is often slow in adaptation and can become unstable, especially in admittance-based control schemes. Therefore, force control is often combined with various learning procedures that can effectively eliminate the stability problems but are unsuccessful with stochastic error sources [3], [4], [5]. One of the most successful approaches turned out to be compliant control, which can naturally adapt to small environmental deviations [6].

Alternatively, one can exploit the arising contact forces and torques to create more robust assembly policies. Based on this paradigm, we aim to develop an algorithm that can adapt to larger deviations in object position where compliance and force control become unsuccessful. The basic idea of our approach is that while adapting to the environment, the robot also updates the task according to the current position and orientation. To adapt to the environment in real-time, we use two techniques: firstly, the task formulation using a pose invariant description and, secondly, the execution of the task with a controller, which inherently adapts to the variable environmental constraints. We have developed a new position and orientation invariant trajectories method, distinguished by its compact form and computational robustness. They are specially adapted to describe assembly operations. The proposed approach allows us to avoid lengthy and demanding calibration procedures usually required in robotic assembly.

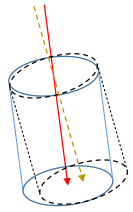

Fig. 1. The solid and the dashed outline show the real and the estimated cylinder, which is subject to peg-in-hole operation. Arrows show the corresponding true and estimated insertion trajectories.

However, the proposed framework requires at least a coarse knowledge of the robot's initial pose when assembling. To this end, we propose a multimodal approach that combines robotic vision and measured contact forces. The appropriate initial pose is then explored with an ergodic controller.

This paper consists of five sections. In Section II we introduce an incremental policy representation for assembly tasks based on position and orientation invariant descriptors. Next, in Section III, we introduce Adaptive Impedance Controller (AIC) as a basic enabler of our approach. It enables following the desired policy and adaptation to the unknown and variable environmental constraints. The search for the

[1] The authors are with the Humanoid & Cognitive Robotics Lab, Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia, email: bojan.nemec@ijs.si, matevz.majcen@ijs.si, mihael.simonic@ijs.si, ales.ude@ijs.si.

[2] The authors are with the Idiap Research Institute, Martigny, Switzerland, email: suhan.shetty@idiap.ch, sylvain.calinon@idiap.ch.

initial assembly pose, which combines vision processing and ergodic control, is explained in Section IV. Experimental evaluation in Section V considers the validation using a mockup, which comprises all assembly challenges considered in our approach, and, finally, the challenging task of bayonet bulb insertion. A short discussion and conclusions are provided in Section VI.

## II. Describing Contact Policy Using Pose-Invariant Trajectory Notation

Our approach aims to encode the task incrementally, i.e., at any time, we specify the motion relative to the robot's current pose. The incremental task formulation is provided using a task description where the position and orientation of the resulting motion trajectory depend only on the initial pose of the task. We refer to this formulation as *pose invariant* task definition. This section proposes a novel approach for formulating invariant trajectories applicable to assembly tasks.

Different formulations have been developed for a position and orientation invariant task description [7], [8]. They are mainly used to recognize and understand human intentions and actions in collaborative robotics. Some approaches aim to compute invariant motion trajectories directly from the video stream [9]. Others apply invariant geometrical properties derived from the definition of curvature and torsion [8], [10] and Frenet-Serret frames [11]. However, most approaches based on differential geometrical properties suffer from the limitations that they can only be applied to non-degenerate trajectories. Roughly speaking, these are trajectories in which the curvature is different from zero everywhere. As straight-line motions are common in robot assembly, many assembly trajectories are degenerate in this sense.

Our approach differs from all previously presented ones in that it does not deal with the general 6-dimensional motion of a rigid body. We consider tasks that allow only partial positional and orientation freedom of motion. The border between the region where the robot motion is constrained by the environment and the region where motion is free is called a *C-surface* [12]. The motion is possible along the tangential direction of the *C-surface* and is constrained in orthogonal directions. The dimension of the C-surface determines the number of degrees of freedom of robot motion in contact with the environment. A typical assembly task is characterized by an at most two-dimensional *C-surface*. Take, for example, a round peg in a hole problem, where the hole is aligned with the $x$ axis of the robot. Here, only two coordinates are freely definable, $x$ and the rotation around $x$ axis. Orientation often changes with position, such as when screwing or assembling a BNC connector. Tasks can be composed as a sequence of such movements, such as inserting a bayonet bulb, composed of a translation followed by a rotation. In such tasks, a maximum of two degrees of freedom change simultaneously, one translational and one rotational. On the other hand, during the execution of the task, the robot follows environmental constraints. The robot can follow them more accurately with prior knowledge of these constraints.

For this reason, we have chosen a four-dimensional task formulation. This formulation can describe most assembly operations and operations where the robot is in physical contact with the environment. In our approach, these degrees of freedom are aligned with the $x$-axis of the local coordinate system, named the *object* coordinate system. This way, one can describe the motion as sliding a ring on a rigid wire, as shown in Fig. 2. This problem comprises all assembly cases considered in our research.

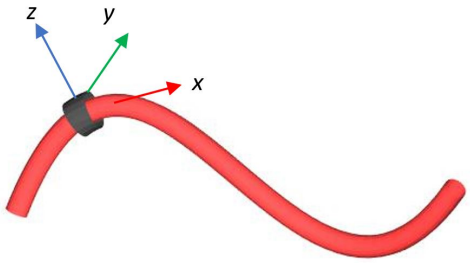


Fig. 2. A movement of a ring on a rigid wire. The coordinate system attached to the ring is referred to as the *object* coordinate system.

Consider that a trajectory of an object (in our case, sliding a ring on a rigid wire) can be represented by a series of trajectory points, which are obtained as a sequence of homogeneous transformation matrices applied to an initial transformation matrix. Each point on the trajectory is characterized as:

$$\mathbf{T}(k) = \mathbf{T}(0)\Delta\mathbf{T}(1)\cdots\Delta\mathbf{T}(k) = \mathbf{T}(0)\prod_{i=1}^{k}\Delta\mathbf{T}(i), \qquad (1)$$

where the homogeneous transformation matrices are calculated as

$$\mathbf{T}(0) = \begin{bmatrix} \mathbf{R}(0) & \begin{bmatrix} x(0) \\ y(0) \\ z(0) \end{bmatrix} \\ \mathbf{0} & 1 \end{bmatrix} \qquad (2)$$

and

$$\begin{aligned} \Delta\mathbf{T}(i) &= \begin{bmatrix} \Delta\mathbf{R}(i) & \begin{bmatrix} \Delta x(i) \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}^T(i\text{-}1)\,\mathbf{R}(i) & \begin{bmatrix} x(i)\text{-}x(i\text{-}1) \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{0} & 1 \end{bmatrix} \end{aligned} \qquad (3)$$

The trajectory is thus parameterized with the local coordinate system $\mathbf{R}(k)$ and the displacement in the direction of the $x$ axis of $\mathbf{R}(k)$. Variable $k$ denotes discrete time. It is related to the continuous time by $t = k\Delta t$, $\Delta t$ being the sampling interval. The main advantage of using the above formulation is an incremental computation of the policy, which enables adaptation to environmental changes. The next trajectory

sample depends only on the previous, as can be seen from the relation

$$\mathbf{T}(k) = \mathbf{T}(k\text{-}1)\Delta\mathbf{T}(k). \tag{4}$$

This notation, however, is highly redundant, as it requires 10 variables to describe a 4 d.o.f motion. A more compact representation can be achieved by utilizing the following transformations

$$\Delta x(k) = \nu(k)\Delta t \tag{5}$$
$$\Delta\mathbf{R}(k) = \exp(\boldsymbol{\omega}(k)\Delta t) \tag{6}$$

where the exponential mapping ($\exp : \mathbb{R}^3 \mapsto \mathbf{S}^3, \forall\boldsymbol{\omega} \in \mathbb{R}^3$, $\|\boldsymbol{\omega}\| < 2\pi$), which gives the rotation matrix, obtained using the Rodrigues formula

$$\exp(\boldsymbol{\omega}\Delta t) = \mathbf{I} + \sin(\|\boldsymbol{\omega}\|\Delta t)\frac{[\boldsymbol{\omega}]_\times}{\|\boldsymbol{\omega}\|} + (1-\cos(\|\boldsymbol{\omega}\|\Delta t)\frac{[\boldsymbol{\omega}]_\times^2}{\|\boldsymbol{\omega}\|^2}, \tag{7}$$

where $[\boldsymbol{\omega}]_\times$ denotes a skew-symmetric matrix composed of elements of vector $\boldsymbol{\omega}$. Scalar $\nu(k)$ and vector $\boldsymbol{\omega}(k)$ constitute a unique differential time-based description of a space curve describing the assembled object position and orientation. Moreover, the spatial position and orientation of the curve depends only on the initial pose, given by $\mathbf{T}(0)$. For this reason, we call this formulation a pose-invariant notation of motion.

A more compact representation can be obtained using unit quaternions. The desired pose passed to the robot controller is often given with a generalized vector in a form

$$\chi(k) = \begin{bmatrix}\boldsymbol{p}^T(k) & \boldsymbol{q}(k)\end{bmatrix} \tag{8}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position vector composed of spatial coordinates $x$, $y$ and $z$, while $\boldsymbol{q} \in \mathbb{R}^4$ is a unit quaternion, describing object orientation. We use the notation $\boldsymbol{q} = v + \boldsymbol{u}$, where $v$ and $\boldsymbol{u}$ are the scalar and vector part of the quaternion. The Eq. (4) can be represented in a form

$$\chi(k) = \Big[\boldsymbol{p}(k\text{-}1) + \nu(k)\Delta t\Delta\boldsymbol{q}(k) * [0 + [1\ 0\ 0]^T] * \overline{\Delta\boldsymbol{q}}(k), \Delta\boldsymbol{q}(k) * \boldsymbol{q}(k\text{-}1)\Big] \tag{9}$$

Operator $\overline{(.)}$ stands for a conjugate quaternion. Applying the well-known relation

$$\Delta\boldsymbol{q}(k) = \exp(\boldsymbol{\omega}(k)\Delta t) \tag{10}$$

we obtain

$$\chi(k) = \Big[\boldsymbol{p}(k\text{-}1) + \nu(k)\Delta t\exp(\boldsymbol{\omega}(k)\Delta t) * [0 + [1\ 0\ 0]^T] * \exp\overline{(\boldsymbol{\omega}(k)\Delta t(k))},\ \exp(\boldsymbol{\omega}(k)\Delta t) * \boldsymbol{q}(k\text{-}1)\Big]. \tag{11}$$

For quaternions, the exponential map is defined as

$$\exp(\boldsymbol{\omega}) = \begin{cases} \cos(\|\boldsymbol{\omega}\|) + \sin(\|\boldsymbol{\omega}\|)\dfrac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}, & \boldsymbol{\omega} \neq 0 \\ 1 + [0,0,0]^T, & \text{otherwise} \end{cases} \tag{12}$$

The new trajectory sample depends only on the previous generalized vector $\chi(k-1)$ and four properties, calculated from $\nu(k)$ and $\boldsymbol{\omega}(k)$, that are pose invariant. In the remainder of the text we refer to them as invariances.

### A. Learning of pose-invariant trajectory representation

Unlike other invariant trajectory representations, which allow describing unconstrained 6 d.o.f motion, calculating invariances is straightforward. Moreover, the solution always exists, and it is unique. From a set of sampled trajectory tuples $[\boldsymbol{p}(k)\ \boldsymbol{q}(k)], k = 1 \cdots T$, translational and angular velocities of the object frame are calculated as

$$\nu(k) = \|\boldsymbol{p}(k) - \boldsymbol{p}(k-1)\|/\Delta t \tag{13}$$
$$\boldsymbol{\omega}(k) = 2\log(\boldsymbol{q}(k) * \overline{\boldsymbol{q}}(k-1))/\Delta t \tag{14}$$

The quaternion logarithm $\log : \mathbf{S}^3 \mapsto \mathbb{R}^3$, that maps the quaternion $\boldsymbol{q}$ to the angular velocity $\boldsymbol{\omega}$ , is defined as

$$\log(\mathbf{q}) = \log(v, \mathbf{u}) = \begin{cases} \arccos(v)\dfrac{\mathbf{u}}{\|\mathbf{u}\|}, & \mathbf{u} \neq 0 \\ [0,0,0]^{\mathrm{T}}, & \text{otherwise} \end{cases} \tag{15}$$

Next, we define a path variable $s$ as a weighted arc length

$$s(t) = \int_0^{t_{max}} (\nu(t) + \gamma\|\boldsymbol{\omega}(t)\|)dt. \tag{16}$$

Scalar $\gamma$ compensates for the different translational and rotational motion metrics. For the discrete-time, it turns to

$$s(k) = \sum_1^T \|\boldsymbol{p}(k) - \boldsymbol{p}(k\text{-}1)\| + 2\gamma\|\log(\boldsymbol{q}(k) * \overline{\boldsymbol{q}}(k\text{-}1))\|, \tag{17}$$

$t_{max} = T\Delta t$. Using $s(k)$ as a phase variable, we encode $\nu(k)$ and $\boldsymbol{\omega}(k)$ with a sum of $M$ Gaussian radial base functions $\Psi$ in the form

$$\hat{\nu}(s) = \mathbf{x}(s)\boldsymbol{w}_\nu, \tag{18}$$
$$\hat{\boldsymbol{\omega}}(s) = \mathbf{X}(s)\mathbf{W}_\omega \tag{19}$$
$$\mathbf{x}(s) = \frac{[\Psi_1(s), \ldots, \Psi_N(s)]}{\sum_{j=1}^N \Psi_j(s)}, \tag{20}$$
$$\mathbf{X}(s) = \begin{bmatrix}\mathbf{x}(s) \\ \mathbf{x}(s) \\ \mathbf{x}(s)\end{bmatrix} \tag{21}$$
$$\Psi_j(s) = \exp\left(-h_j\left(s - c_j\right)^2\right),\ j = 1, \ldots, M. \tag{22}$$

The vector $\boldsymbol{w}_\nu \in \mathbb{R}^M$ and matrices $\mathbf{W}_\omega \in \mathbb{R}^{3\times M}$ contain free parameters that determine the shape of the encoded variables $\hat{\nu}(s)$ and $\hat{\boldsymbol{\omega}}(s)$, $c_j$ are the centers of RBFs and $h_j$ their widths. Usually, they are selected so that RBFs are evenly distributed along the trajectory. RBF weights are learned after the trajectory demonstration by regression [13].

### B. Reconstruction of robot policy from pose-invariant trajectory representation

Invariant representations are very suitable for motion recognition, as we can directly compare invariances of two translated and rotated trajectories [11], [10], [14]. However, invariant policy descriptions can also be used to control a robot. The advantage of such a description is that the policy is the same no matter how the trajectory has to be translated and rotated in space. This is because the rotation

and translation are determined relative to the previous robot pose. Another, perhaps more important benefit is that the assembled object can move freely (with some limitations) during the assembly. At the same time, the policy inherently adapts to the geometric changes of the environment, e.g, the base of an assembled object.

For proper execution of the invariant policies, it is extremely important to estimate proper values of invariances at the robot's current pose and, consequently, precise estimation of the phase variable $s$. As we will see in the next section, we allow the robot to adapt to environmental changes by setting the controller compliant in all axes except in the tangential direction of motion. For this reason, we can not apply Eq. 17 for phase calculation anymore. Let's consider, for example, the simplest possible case, where the robot has to follow a straight rod (or insert a round peg into a hole, which is the equivalent case). We allow the rod to change its position and orientation constantly. In such a case, the robot performs substantial position and rotational motion. On the other hand, all that matters is the relative path along the rod. Relative paths on the rod can be obtained by projecting the robot's motion to the vector aligned with the current rod orientation. This vector can be calculated from the current object pose and the predicted object pose, given by Eqs. (4–6). During the online reconstruction of the robot policy, the phase variable is thus calculated as

$$s(k) = \sum_{1}^{T} (\boldsymbol{p}(k)\text{-}\boldsymbol{p}(k\text{-}1) + 2\gamma \log(\boldsymbol{q}(k) * \overline{\boldsymbol{q}}(k\text{-}1)) \cdot \boldsymbol{a}(k). \quad (23)$$

In the above equation, the operator $(\cdot)$ denotes the scalar product and $\boldsymbol{a}(k) = \mathbf{R}_o(k)[1\ 0\ 0]^T$ is the $x$ axis of the object frame. The calculation of the phase variable also reveals the main limitation of our approach to robust assembly. Namely, the assembled object must never move in the direction of the current tangent of the assembly trajectory, as this results in improper phase determination.

When controlling the robot, the robot's actual position lags behind the reference position due to uncompensated friction in the joints and friction with the robot environment. Therefore, in the incremental generation of the trajectory, where we feed an offset from the current position, the robot often may not move at all. To overcome the above-mentioned problem, we propose adaptive lag compensation. The phase error is defined as

$$e_s(s) = \|\dot{\boldsymbol{p}}_d(s) - \dot{\boldsymbol{p}}(s)\| + \gamma \|\boldsymbol{\omega}_d(s) - \boldsymbol{\omega}(s)\| \quad (24)$$

Based on the estimated phase error, we calculate a new phase variable

$$s^* = s + \delta \int e_s(s) dt, \quad (25)$$

which is used to encode $\hat{\nu}(k)$ and $\hat{\boldsymbol{\omega}}(k)$ that are feed to the robot controller

$$\hat{\nu}(s) = \mathbf{x}(s^*)\boldsymbol{w}_\nu \quad (26)$$

$$\hat{\boldsymbol{\omega}}(s) = \mathbf{X}(s^*)\mathbf{W}_\omega. \quad (27)$$

$\delta$ is a suitably chosen or learned positive constant. The proposed lag compensation algorithm effectively compensates for the unknown friction and assures the robot follows the desired path with a given velocity.

Summarizing, during the reconstruction, the phase is calculated by Eq. (23), translational and angular velocities $\hat{\nu}(s)$ and $\hat{\boldsymbol{\omega}}(s)$ are calculated using Eq. (26, 27) and the next trajectory sample, which is passed to the robot controller, is obtained using Eq. (9).

## III. Adaptive Impedance Controller (AIC)

This section presents a controller that can autonomously adapt to environmental constraints. We assume that the environment constrains the robot's motion so that only one direction is possible at any given time. Rotations are also constrained so that the robot's orientation can only change around this direction. It is referred to as the tangential direction of the assembly trajectory. We formalize this motion control by utilizing the *object frame* $\mathbf{R}_o$ introduced in the previous section, which is attached to the robot tool center point (TCP). Given the object frame, we need a control law enabling arbitrary compliance application along the frame axes. In our experiments, we used the passivity-based impedance control designed for manipulators with flexible joints [15]. However, it was necessary to modify the control law to freely set compliance in the object frame. The commanded torque $\boldsymbol{\rho_c} \in \mathbb{R}^N$, which is passed to the robot motors, is calculated as [1]

$$\begin{aligned} \boldsymbol{\rho_c} &= \mathbf{B}\mathbf{B}_\Theta^{-1}\boldsymbol{u} + (\mathbf{I} - \mathbf{B}\mathbf{B}_\Theta^{-1})\boldsymbol{\rho}, \\ \boldsymbol{u} &= \mathbf{J}^{\mathrm{T}}(\boldsymbol{\theta}) \left( \begin{bmatrix} \boldsymbol{f}_c \\ \boldsymbol{m}_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{f}_a \\ \boldsymbol{m}_a \end{bmatrix} \right) + \mathbf{g}(\boldsymbol{\theta}) + \mathbf{N}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}_0, \end{aligned} \quad (28)$$

where $N$ is the number of robot joints, $\boldsymbol{\theta} \in \mathbb{R}^N$ is the vector of joint angles estimated from the corresponding motor angles $\Theta \in \mathbb{R}^N$ [16], $\mathbf{J} \in \mathbb{R}^{N \times 6}$ is the manipulator Jacobian, while $\mathbf{B}$, $\mathbf{B}_\Theta \in \mathbb{R}^{6 \times 6}$ denote the positive definite diagonal matrices of the actual and the desired joint inertia, respectively. The aim of the term $\mathbf{B}\mathbf{B}_\Theta^{-1}$ is to reduce the joint inertia. $\boldsymbol{\rho}$ are the measured joint torques, and $\mathbf{g}(\boldsymbol{\theta})$ is the gravity vector [17]. $\mathbf{N}(\boldsymbol{\theta}) = (\mathbf{I} - \mathbf{J}^+(\boldsymbol{\theta})\mathbf{J}(\boldsymbol{\theta})) \in \mathbb{R}^{N \times N}$ is the null space projection operator, $\mathbf{J}^+(\boldsymbol{\theta})$ denotes the Moore-Penrose pseudo-inverse of the Jacobian and $\dot{\boldsymbol{\theta}}_0 \in \mathbb{R}^N$ is the null space velocity vector. $\boldsymbol{f}_a$ and $\boldsymbol{m}_a$ are additional forces and torque vectors in task coordinates, which are used to override the effect of friction forces during the assembly. The motor torque controller (28) reduces the motor inertia and compensates for the robot's non-linear dynamics. In contrast, the second part of Eq. (28) provides for the desired impedance and damping, additional task force, and null space motion. The task command input $[\boldsymbol{f}_c^{\mathrm{T}}, \boldsymbol{m}_c^{\mathrm{T}}]^{\mathrm{T}}$ is chosen as

$$\boldsymbol{f}_c = -\mathbf{R}_o\mathbf{D}_p\mathbf{R}_o^{\mathrm{T}}\dot{\boldsymbol{p}} + \mathbf{R}_o\mathbf{K}_p\mathbf{R}_o^{\mathrm{T}}\boldsymbol{e}_p, \quad (29)$$

$$\boldsymbol{m}_c = -\mathbf{R}_o\mathbf{D}_o\mathbf{R}_o^{\mathrm{T}}\boldsymbol{\omega} + \mathbf{R}_o\mathbf{K}_o\mathbf{R}_o^{\mathrm{T}}\boldsymbol{e}_q, \quad (30)$$

where position and orientation tracking errors are defined as $\boldsymbol{e}_p = \boldsymbol{p}_d - \boldsymbol{p}$ and $\boldsymbol{e}_q = 2\log(\overline{\boldsymbol{q}} * \boldsymbol{q}_d)$. $\mathbf{K}_p$ and $\mathbf{K}_o \in$

[1] for the sake of simplicity, we omitted discrete time index $k$ in the following equations

$\mathbb{R}^{3\times3}$ are the diagonal matrices that define the positional and rotational stiffness along and around coordinate axes. $\mathbf{D}_p$ and $\mathbf{D}_o \in \mathbb{R}^{3\times3}$ are diagonal damping matrices, which are set as diagonal elements of the block diagonal matrix calculated so that the overall system is critically damped. The null space velocity has to be controlled to prevent non-conservative motion. An appropriate solution is to set the desired null space velocities to zero, $\dot{\boldsymbol{\theta}}_0 = -\mathbf{K}_n\dot{\boldsymbol{\theta}}$, which results in an energy dissipation controller [18]. $\mathbf{K}_n \in \mathbb{R}^{N\times N}$ is a positive-definite diagonal gain matrix.

Next, it is essential to provide the necessary controller stiffness when the robot is expected to change its direction of motion, i.e, the controller should adapt its compliance according to the commanded motion. We propose the following formulation of variable compliance:

$$\mathbf{K}_p = \begin{bmatrix} k_{p_s}|\nu| & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{31}$$

$$\boldsymbol{\omega}_o = \mathbf{R}_o\boldsymbol{\omega} \tag{32}$$

$$\mathbf{K}_o = \begin{bmatrix} k_{o_s}|\omega_{o_x}| & 0 & 0 \\ 0 & k_{o_s}|\omega_{o_y}| & 0 \\ 0 & 0 & k_{o_s}|\omega_{o_z}| \end{bmatrix} \tag{33}$$

To preserve stability, the diagonal elements of $\mathbf{K}_p$ and $\mathbf{K}_o$ are limited by an upper bound.

The variable compliance along the object frame ensures the basic adaptation of the regulation law according to the limitations of the environment. However, this adaptation alone may not be enough. Such a control law might fail due to uncompensated friction in the robot joints and/or friction between individual objects during assembly, especially when inserting pegs into very tight holes. Different measures should be taken to avoid the above-mentioned problems. Force control is often used to realize a remote center of compliance [19]

$$\boldsymbol{f}_a = \mathbf{K}_f\mathbf{R}_o(\boldsymbol{f}_d - \boldsymbol{f}), \tag{34}$$

$$\boldsymbol{m}_a = \mathbf{K}_m\mathbf{R}_o(\boldsymbol{m}_d - \boldsymbol{m}). \tag{35}$$

$\mathbf{K}_f$ and $\mathbf{K}_m$ are force controllers gains, $\boldsymbol{f}_d$ and $\boldsymbol{m}_d$ are the desired forces and torques (usually set to 0), and $\boldsymbol{f}$ and $\boldsymbol{m}$ are measured forces and torques from the sensor in tool coordinates. Transformation $\mathbf{R}_o$ maps the measured forces and torques from the tool coordinate system to the robot coordinate system.

Another approach is to learn the appropriate additional forces and torques $\boldsymbol{f}_a, \boldsymbol{m}_a$, as proposed in [20]. In their work, the authors propose to superimpose force and torque oscillations, where the meta parameters of the direction, amplitude, and frequency of oscillating force are learned. Others propose a dithering signal with predefined constant parameters [21], [22]. In our approach, we also chose a constant frequency of oscillations. In contrast, the direction of the oscillations was adapted according to the tangent of the movement, and the amplitude depended on the tracking error. Thus, the superimposed forces are given by

$$\boldsymbol{f}_a(k) = \mathbf{R}_o[1\ 0\ 0]^T\kappa(|\boldsymbol{e}_p(k)|)(sin(k\pi/4)+1) \tag{36}$$

where $\kappa$ is a constant that determines the superimposed force amplitude and vectors $\boldsymbol{p}_d, \boldsymbol{p}$ denote commanded and measured position, respectively.

The corresponding overall control scheme for real-time adaptation of assembly policy is outlined in Fig. 3.

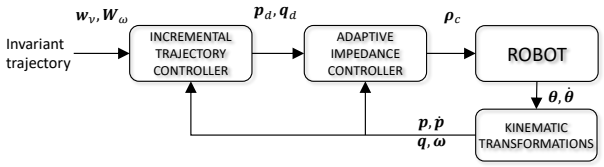


Fig. 3. Incremental controller block scheme.

## IV. Initial pose search with ergodic control

For the robot to follow the environmental constraints imposed by the assembly operation, it must reach the required initial pose with generally very low tolerances. Vision processing alone cannot assure the required accuracy; therefore, we propose combining vision with an active search algorithm. For active search, we apply a framework based on ergodic search [23], [24]. In contrast to the standard control problem, where the goal is to track the robot's pose, the goal of ergodic control is to track a distribution that needs to be covered by the robot. Given the probability distribution, the resulting system has natural exploration behaviors by considering information about the regions that should be explored.

Ergodic search can be viewed as a trajectory generator whose input is the probability distribution. The output is a smooth continuous trajectory that visits spatially distributed locations according to the given probability distribution. The original approach [23] is based on quadratic cost minimization and representation of the trajectories with Fourier series. The Fourier series coefficients are computed using multi-dimensional integration over the spatial domain, which can be computationally costly for high-dimensional state spaces. Moreover, the control loop involves algebraic operations on multi-dimensional arrays, making the original approach too slow for real-time applications in high-dimensional tasks. This problem was addressed in [24] by relying on tensor trains, an approach for low-rank factorization of tensor data, used to compute Fourier coefficients efficiently. Consequently, the ergodic trajectory can be computed in real-time, even for six DOF distributions. The proposed approach was experimentally verified in a peg-in-hole task, demonstrating the robustness of the approach to peg displacements within the robot gripper. Comparison with traditional search methods such as Gaussian Mixture Model (GMM) sampling and spiral search demonstrated that an ergodic controller exceeds other methods in terms of speed (time to find the hole) and success rate. Fig. 4 shows a typical trajectory generated by ergodic control and a trajectory generated with random sampling for a three-DOF problem.

Ergodic control is thus a perfect candidate for an active search for the initial pose in assembly tasks. However, to limit the search space, we apply vision processing. The assembled object is captured with an RGB-D camera. The point cloud $\mathcal{P}_0 = \{\boldsymbol{p}_i\}_{i=1}^{N}$ consisting of $N$ points on the object is associated with a probability distribution for the initial assembly pose. Point cloud and probability distributions are associated by computing a homogeneous transformation $\mathbf{T}_a$ between their centers. We assume that we obtained an assembly policy for the object at a location giving rise to point cloud $\mathcal{P}_0$. Next time we perform the assembly on a possibly translated and rotated object, we capture the point cloud $\mathcal{P}_1$ and compute the transformation matrix $\mathbf{T}_b$ between $\mathcal{P}_0$ and $\mathcal{P}_1$. In our case, transformation matrix $\mathbf{T}_b$ was computed using the *CloudCompare* [25] implementation of the Iterative Closest point (ICP) algorithm [26].

To calculate the probability distribution for the assembled object pose, we apply the transformation $\mathbf{T}_a\mathbf{T}_b\mathbf{T}_a^{-1}$ to the probability distribution at the original pose. Next, we calculate the corresponding ergodic trajectory and execute it. When the robot finds the initial pose[2] We start the assembly by taking the current robot pose as the initial. Since the trajectory is computed incrementally using Eq. (9), we do not need to apply any other transformation.

## V. Experimental evaluation

In this section, we experimentally verify the proposed framework for the robust execution of contact policies, applied to the seven degrees-of-freedom collaborative robot Franka Research 3 equipped with a two fingers gripper. The AIC controller was implemented using the *libfranka* library and *ros_control* framework in C++. The incremental controller, ergodic controller, and vision processing were implemented in Matlab and Python and communicated with the AIC using ROS. Point clouds were captured using the Intel RealSense D435i RGB-D camera.

The first experiments were done on a mockup covering all the essential assembly problems. It consists of a curved rod on which there is a sliding bearing. The sliding bearing is equipped with a screw mechanism, which is used to fix

[2]Usually, we use position and force measurements to detect that the robot has found the initial pose. This detection method is case-specific.

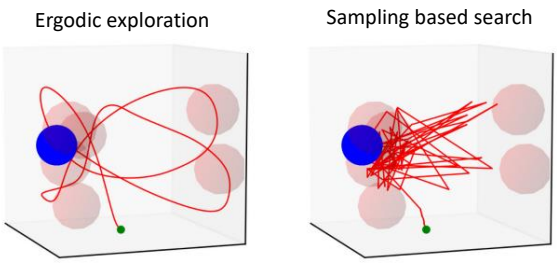


Fig. 4. Example trajectories of ergodic exploration strategy (left) and the sampling-based search (right). The GMM has six equally weighted mixture components (red spheres). Blue sphere is the selected target region within the reference probability distributions. Note that the target region is unknown to the search algorithm.

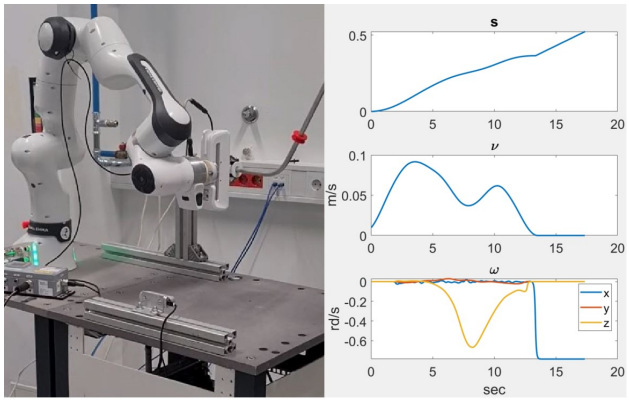


Fig. 5. Left: A mockup for testing assembly operations Right: Phase variable and commanded trajectory invariances.

it to the thread attached to the rod's other end. The rod is clamped at one end only; the other end can be moved freely. The mockup is shown in Figure 5.

The initial policy was learned by kinesthetic teaching with the robot in gravity compensation mode and recalculated in a pose-invariant incremental representation. The corresponding invariant variables are shown in Fig. 5. A simple replay of the learned trajectory failed. The robot got stuck in the curved part of the rod. The results are shown in Fig. 6 left, where we can notice increased force in the local $x$ direction as the robot gets stuck. Next, we applied the AIC proposed in Section III. The robot successfully accomplished the task, as shown in Fig. 6 center. The increased force in $x$ direction is due to the contact with the thread attached at the rod end during the screwing. However, it could not complete the task if we randomly moved the other end of the rod extensively during the task. Finally, we performed the same experiment by applying incremental policy updates from pose invariances (Subsection II-B) and AIC. The robot accomplished the task successfully despite extensive disturbances induced by randomly moving the free side of the rod, as long as the robot joints stayed within physical limitations. The robot trajectory, disturbed by the operator and the corresponding forces and torques, are shown in Fig. 6 right. Note also that the contact forces didn't increase substantially despite the disturbances, as AIC suppressed them effectively.

The next experiment was the bayonet bulb insertion for experimental evaluation. The first part of the task is a peg-in-hole task, where a gap in the bayonet bulb casing determines the bulb's orientation. The next part is the rotation of the bulb to lock it in the final position. The car bulb and the casing (see Fig. 8 left) were 3D printed and fit each other well [3]. The assembly policy was learned by demonstration. After learning, we captured the bulb casing point cloud $\mathcal{P}_0$. Next, we captured 32 images of the bulb casing from 32 perspectives and calculated the centers and principal axes of the gathered point clouds using the CloudCompare framework. The point clouds and the probability distributions for positions in a

[3]Please note that due to the low tolerances, assembly of the printed bulb was more challenging than the assembly of the real bulb, considered in our previous research [27]

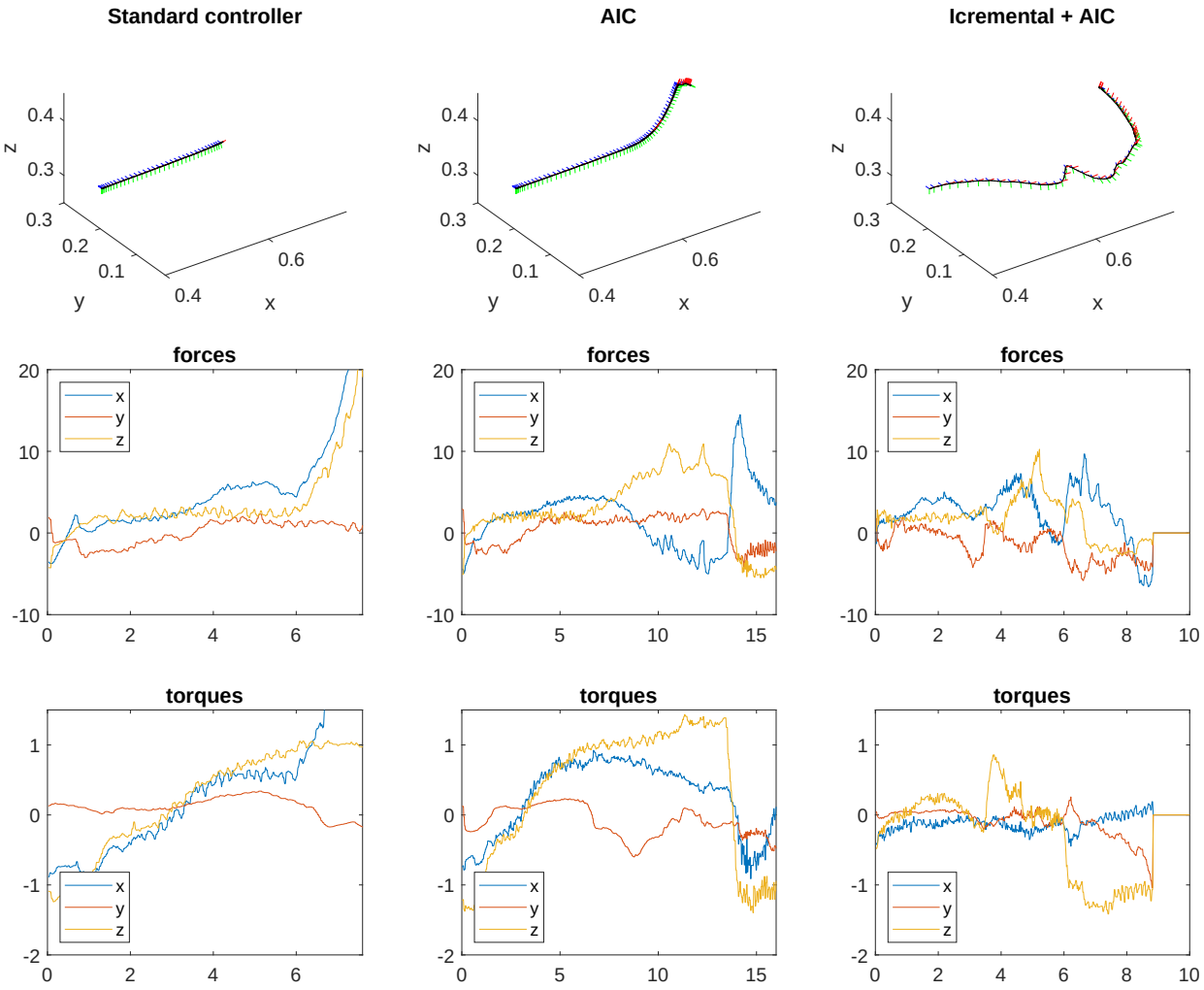


Fig. 6. Robot trajectories, contact forces, and torques obtained with different control algorithms and trajectory generation schemes.

GMM form are shown in Fig. 7, left. Finally, we encoded the assembly policy in a position/orientation invariant form, as described in Section II.

To test the robustness of assembly regarding the unknown position and orientation of the bulb casing, we mounted it on an inclined plate whose position and rotation were unknown to the robot (see Fig. 8). We recorded the point cloud $\mathcal{P}_1$ and displaced the GMM distribution associated with $\mathcal{P}_0$ to the new pose (Fig. 7 right). Based on the displaced distribution, the ergodic controller found a new initial pose for the assembly in 9 seconds on average. After that, the robot successfully inserted the bulb in a casing by incremental policy updates from pose invariances and the proposed AIC. The phase variable and invariances are shown in Fig. 8. Finally, we repeated the same experiment, but this time the case was held by a human performing random displacements of the housing. Nevertheless, the robot successfully inserted the bulb into the housing using the proposed framework. The corresponding videos of all experiments illustrate the robust assembly using invariant policy representation.

## VI. Conclusion

In this research, we considered the problem of robust execution of the assembly tasks, where the pose of the assembled object is not known in advance. Regardless of whether we can precisely determine the starting point of the assembly, even minimal changes in the orientation assessment can cause large deviations, especially for extended objects. In addition, the usual assembly procedures fail if the object's posture changes during assembly. Our main challenge was developing a methodology for assembly policy generation that seamlessly adapts to environmental changes.

We addressed two problems, a) how to generate an environmentally adaptable assembly policy and b) determine the starting point for implementing an assembly policy. To solve the first problem, we proposed incremental execution of the trajectory using AIC and pose-invariant formulation of the policy. We combined point clouds and an ergodic controller to solve the second problem.

The proposed approach is suitable for assembly operations and can be used for many other tasks where the robot is in

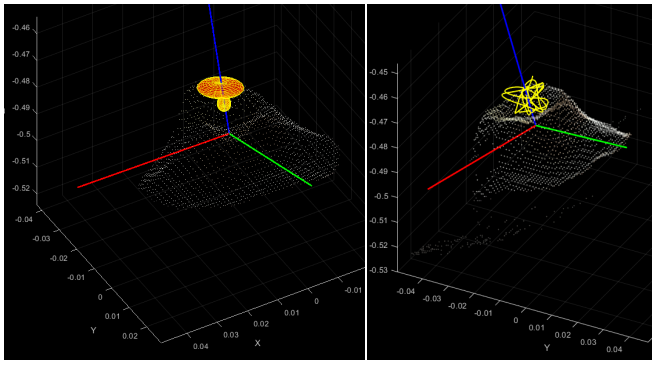

Fig. 7. Left: Point cloud of the bulb casing $\mathcal{P}_0$ (grey) and the GMM for the positional distribution used in ergodic controller (yellow-red); Right: Displaced point cloud $\mathcal{P}_1$ (grey) and a positional trajectory for the ergodic search (yellow).

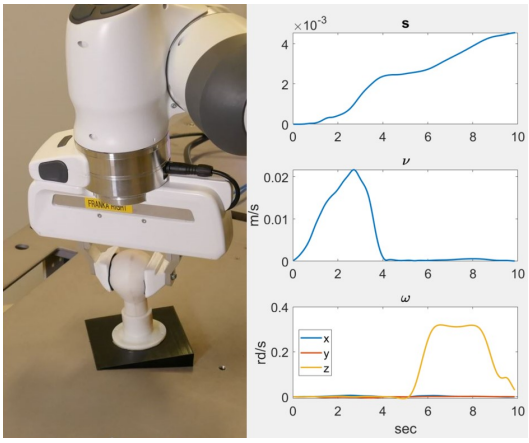


Fig. 8. Left: Franka robot during the execution of the position and orientation invariant policy for the bayonet bulb insertion into the casing. Right: Phase and commanded trajectory invariances for bayonet bulb insertion.

constant contact with the environment. The requirement that the assembled object never moves in the direction of the current tangent of the assembly trajectory can be overridden by additional sensors, e.g., vision. Further steps in our research will extend our framework to assembly cases where a component carried by the robot may shift in the gripper due to imperfect grasping.

## ACKNOWLEDGMENT

The research leading to these results has received funding from the Horizon 2020 Research and Innovation Action ReconCycle, GA no. 871352.

## REFERENCES

[1] L. Evjemo, T. Gjerstad, E. Grøtli, and G. Sziebig, "Trends in smart manufacturing: Role of humans and industrial robots in smart factories," *Current Robotics Reports*, vol. 1, 06 2020.

[2] I.-W. Kim, D.-J. Lim, and K.-I. Kim, "Active peg-in-hole of chamferless parts using force/moment sensor," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, vol. 2, 1999, pp. 948–953 vol.2.

[3] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, no. 2, pp. 199–217, 2015.

[4] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," *Applied Sciences*, vol. 10, no. 19, p. 6923, 2020.

[5] M. Suomalainen, Y. Karayiannidis, and V. Kyrki, "A survey of robot manipulation in contact," *Robotics and Autonomous Systems*, vol. 156, p. 104224, 2022.

[6] S. K. Yun, "Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion?" *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1647–1652, 2008.

[7] D. Aarno and D. Kragic, "Motion intention recognition in robot assisted applications," *Robotics and Autonomous Systems*, vol. 56, no. 8, pp. 692–705, 2008.

[8] M. Saveriano and D. Lee, "Invariant representation for user independent motion recognition," in *2013 IEEE RO-MAN*, 2013, pp. 650–655.

[9] Y. Piao, K. Hayakawa, and J. Sato, "Space-time invariants for recognizing 3D motions from arbitrary viewpoints under perspective projection," *Third International Conference on Image and Graphics (ICIG'04)*, pp. 200–203, 2004.

[10] M. Vochten, T. D. Laet, and J. De Schutter, "Generalizing demonstrated motion trajectories using coordinate-free shape descriptors," *Robotics and Autonomous Systems*, vol. 122, p. 103291, 2019.

[11] R. Soloperto, M. Saveriano, and D. Lee, "A bidirectional invariant representation of motion for gesture recognition and reproduction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6146–6152.

[12] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 418–432, 1981.

[13] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, oct 2010.

[14] Y. Li, R. Xia, and X. Liu, "Learning shape and motion representations for view invariant skeleton-based action recognition," *Pattern Recognition*, vol. 103, p. 107293, 2020.

[15] A. Albu-Schaffer, C. Ott, and G. Hirzinger, "A unified passivity-based control framework for position, torque and impedance control of flexible joint robots," *The International Journal of Robotics Research*, vol. 26, no. 1, pp. 23–39, 2007.

[16] C. Ott, A. Albu-Schaffer, A. Kugi, and G. Hirzinger, "On the Passivity-Based Impedance Control of Flexible Joint Robots," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 416–429, 2008.

[17] C. Ott, A. Albu-Schaffer, A. Kugi, S. Stramigioli, and G. Hirzinger, "A passivity based Cartesian impedance controller for flexible joint robots - part I: torque feedback and gravity compensation," in *IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, 2004, pp. 2659–2665.

[18] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Robotics and Automation, IEEE Journal of*, vol. 3, pp. 43–53, 1987.

[19] J. Jiang, Z. Huang, Z. Bi, X. Ma, and G. Yu, "State-of-the-art control strategies for robotic PiH assembly," *Robotics and Computer-Integrated Manufacturing*, vol. 65, pp. 1–26, 2020.

[20] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A framework for robot manipulation: Skill formalism, meta learning and adaptive control," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5844–5850, 2019.

[21] S. Ipri and H. Asada, "Tuned dither for friction suppression during force-guided robotic assembly," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 1, 1995, pp. 310–315 vol.1.

[22] A. Stolt, A. Robertsson, and R. Johansson, "Robotic force estimation using dithering to decrease the low velocity friction uncertainties," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3896–3902.

[23] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4, pp. 432–442, 2011.

[24] S. Shetty, J. Silvério, and S. Calinon, "Ergodic exploration using tensor train: Applications in insertion tasks," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 906–921, 2022.

[25] "Cloud compare v2.13 GPL software," http://www.cloudcompare.org/, 2023.

[26] Z. Zhang, *Iterative Closest Point (ICP)*. Boston, MA: Springer US, 2014, pp. 433–434.

[27] M. Simonič, A. Ude, and B. Nemec, "Hierarchical learning of robotic contact policies," *submitted to Robotics and Computer-Integrated Manufacturing*, 2023.