Reinforcement Learning of Insertion Tasks: A Comparison Between Policy Structures

Adriaan Mulder [1] **Nick Striebel** [1]

Abstract

This paper presents a comparative study of distinct approaches for learning insertion policies in robotics: the Manifold Stable Vector Field (MSVF), Multi-Layer Perceptron (MLP), and Residual Reinforcement Learning (Residual RL). The MSVF approach, designed to ensure stability by construction, is evaluated against a standard MLP and a Residual RL method that combines a learned policy with a linear attractor. The performance of these approaches is assessed across multiple simulated environments, including 2D and 3D tasks of increasing complexity. Results indicate that while the MSVF consistently achieves stability and efficient learning in simpler environments, the Residual RL approach excels in more complex scenarios, such as 3D L-shaped object insertion. The study highlights the trade-offs between stability guarantees and learning efficiency, offering insights into applying these methods in real-world assembly tasks.

1. Introduction

Insertion tasks are a fundamental problem in robotics (Khan et al., 2014; Suomalainen et al., 2022), often encountered in assembly operations, where an object needs to be precisely inserted into a designated location. Examples of tasks are a peg into a hole or a key into a lock. Such tasks require precise manipulation and accurate alignment. There are two primary difficulties in insertion tasks. The first arises from contact between the object to insert and the environment. This can restrict the robot's freedom of movement or block its movement in the worst case. Additionally, there is often a lack of detailed information about the environment. Properties like friction are hard to estimate. To achieve reliable performance, the insertion process is often manually defined in a time-consuming process. Methods like imitation learning (IL) and reinforcement learning (RL) try to remove the need for this necessity. However, a very likable but hardly achievable property with these methods is stability. Stability during training can prevent the robot from adopting unsafe or sub-optimal policies, thereby improving the overall learning efficiency and effectiveness. Furthermore, in safety-critical environments, stability is mandatory.

A structure that provides stability is the so-called manifold stable vector field (MSVF), which is already applied to IL (Urain et al., 2022) and RL (Hellwig, 2023). RL poses a higher difficulty regarding learning but poses the chance of outperforming IL, especially regarding unseen states. The results of (Hellwig, 2023) do not show the best performance, thus we revisit his work once more.

In addition to the MSVF approach, we also investigate two other policy structures: a Multi-Layer Perceptron (MLP) network with no specific structural assumptions and the widely adopted residual learning approach (Silver et al., 2018). The MLP serves as a baseline to understand the impact of incorporating minimal prior knowledge, while the residual learning approach is explored for its potential to refine and enhance basic policies by leveraging additional information or corrections.

We conduct a comparative study of these three policy categories, focusing on their performance during the learning phase, their final task performance, and their ability to generalize to new initial states. This evaluation is carried out across multiple environments with varying levels of complexity: starting with a simple 2D box insertion, more challenging scenarios such as tight stick insertion and L-shaped object insertion using the Panda Franka Emika robot are performed. By testing in different environments, we aim to provide a comprehensive analysis of how different policy structures impact performance. Executing this in simulation allows for controlled experimentation and consistent evaluation. This study shall provide insights for broader applications in automated assembly and manufacturing.

2. Related Work

Over the years, considerable research has been conducted on automating insertion tasks, leveraging both IL (Wan

[*]Equal contribution [1]Department of Computer Science, Technical University of Darmstadt, Darmstadt, Germany. Correspondence to: Adriaan Mulder <adriaanmariojan.mulder@stud.tu-darmstadt.de>, Nick Striebel <nick.striebel@stud.tu-darmstadt.de>.

Proceedings of the IAS Conference on Robot Learning, TU Darmstadt, Germany. Summer term 24.

et al., 2017; Tang et al., 2015; 2016; Chang et al., 2022; Beltran-Hernandez et al., 2020; Luo et al., 2018) and RL techniques (Inoue et al., 2017; Lee et al., 2018; Thomas et al., 2018; Fan et al., 2018). Imitation learning involves teaching a robot to mimic demonstrations, while reinforcement learning focuses on enabling the robot to learn optimal policies through exploration and exploitation. Methods that combine multiple policies (Lee et al., 2020; Silver et al., 2018; Johannink et al., 2018; Davchev et al., 2020) try to increase insertion performance even further.

A category that falls into the field of multi-policies is residual learning (Silver et al., 2018). In this case, a base policy is combined with a learned one. The hope is that the learned policy makes minor adjustments to the base policy. While this often works, the stability of the base policy can be eliminated, or the learned residual is restricted too much, resulting in reduced performance.

Different studies (Urain et al., 2022; Khader et al., 2020; 2021) address the challenge of stability, but all have their downside. For example, (Khader et al., 2020) only works with Euler angles. Representative orientation representations and stability are jointly achieved by manifold stable vector fields (Urain et al., 2022). We revisit the approach introduced by J. Hellwig (2023) to learn a stable vector field (SVF) using reinforcement learning. This method ensures stability during the evaluation phase (Khader et al., 2021) and throughout the training process.

3. Background

This section provides an overview of the applied robot control principles, policy architectures, and learning algorithms. It uses notations corresponding to the insertion task for a better understanding. The position vector is called $\boldsymbol{p}$, and the rotation can be represented in different ways. For instance, using a rotation matrix $\boldsymbol{R} \in SO(3)$ ('Special Orthogonal group') or a rotation vector $\boldsymbol{r} \in \mathfrak{so}(3)$ in corresponding Lie algebra. In combination, these two variables can describe the state of the end effector $\boldsymbol{x}$, e.g., $\boldsymbol{x} = (\boldsymbol{p}, \boldsymbol{R})$. The output of our policies is the desired change in the state $\Delta\boldsymbol{x}$ or the desired velocity $\dot{\boldsymbol{x}}$.

3.1. Control

3.1.1. Operational Space Control

Operational Space Control (OSC) computes the desired joint torques $\boldsymbol{\tau}$ using the Jacobian $\boldsymbol{J}$ and the control signal in the operational space $\boldsymbol{u}$ (Peters & Schaal, 2006b):

$$\boldsymbol{\tau} = \boldsymbol{J}^T \boldsymbol{u} \,. \tag{1}$$

Specifying the forces and torques in the operational space is intuitive and has the advantage of defining the path of the end effector. Disabling gravity in the simulation and only operating the robot far away from singularities removes the need for further correction terms.

3.1.2. Impedance Control

Using a compliant PD-controller is a widely spread approach in insertion tasks as this reduces the risk of damages if the controller gains $\boldsymbol{K}$ are chosen accordingly (Dong et al., 2020). The controller computes a desired end effector acceleration

$$\ddot{\boldsymbol{x}} = \boldsymbol{K}_{\mathrm{P}} \left(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{DES}} \right) - \boldsymbol{K}_{\mathrm{D}} \dot{\boldsymbol{x}} \tag{2}$$

driving the end effector towards the desired state $\boldsymbol{x}_{\mathrm{DES}}$, which is static in insertion tasks. Using the robot mass matrix in end-effector space $\boldsymbol{M}$, the control signal for Equation 1 is

$$\boldsymbol{u} = \boldsymbol{M} \ddot{\boldsymbol{x}} \,.$$

Please consult (Khatib, 1987) for a background on computing $\boldsymbol{M}$.

3.2. Residual Learning

Residual policy learning (Silver et al., 2018) is summarized by

$$\pi(\boldsymbol{x}) = \pi_{\mathrm{BASE}}(\boldsymbol{x}) + \lambda \pi_{\boldsymbol{\theta}}(\boldsymbol{x}) \,.$$

The base policy π_{BASE} can be any arbitrary policy. Weighted with a scaling factor λ, the learned policy $\pi_{\boldsymbol{\theta}}$ can correct the base policy. This paper implements $\pi_{\boldsymbol{\theta}}$ as an MLP. Learning π resorts to learning $\pi_{\boldsymbol{\theta}}$, since $\nabla_{\boldsymbol{\theta}} \pi = \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}$. This shows that the base policy can be a non-differentiable function and arbitrarily complex.

Yet, the goal is to find policy structures that do not need a lot of manual work. This is why it is favorable to choose a simple base policy that helps the residual policy $\pi_{\boldsymbol{\theta}}$ to learn faster by using π_{BASE} as guidance. Ideally π_{BASE} is shared across tasks and $\pi_{\boldsymbol{\theta}}$ provides task-specific adjustments. Moreover, π_{BASE} allows the use of prior knowledge on solving the task, e.g., it can be learned from demonstrations.

3.3. Manifold Stable Vector Fields

A stable policy by construction is desirable since it provides explainability and safety. This is easy to achieve in the space of positions, $\mathbb{R}^3$. However, this is not the case for the rotations space $SO(3)$. The representation of rotations is ambiguous, and so is it with computing distances between rotations. Here, the Lie Groups come to a rescue. Unless noted otherwise, this section is based on (Hall, 2013).

3.3.1. Manifolds and the Lie Groups

A n-manifold $\mathcal{M}$ is a topological space that, around any given point, resembles the Euclidean space $\mathbb{R}^n$, which allows vector calculus. For instance, a sphere, though a 3-dimensional object, locally resembles a 2-dimensional Euclidean space. This local resemblance is established through

so-called charts, which map neighborhoods of the manifold to $\mathbb{R}^n$. An atlas forms a collection of these charts that cover the manifold. If the transition maps between overlapping charts are differentiable, $\mathcal{M}$ is called a differentiable manifold. Each point $\boldsymbol{x} \in \mathcal{M}$ has an associated tangent space, $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, which is a vector space $\mathbb{R}^n$ containing all tangential directions at point $\boldsymbol{x}$. This tangent space can be represented by using an Euclidean coordinate system $\mathbb{R}^n$.
Between two smooth manifolds $\mathcal{M}$ and $\mathcal{N}$, it is possible to define a diffeomorphism $\varphi : \mathcal{M} \to \mathcal{N}$, which is not only a bijection, but also its inverse is differentiable (Lang, 2001a).
A Lie group is a group $\mathcal{G}$ that also possesses the structure of a differentiable manifold. This means that the group operations are smooth functions. Associated with each Lie group is a Lie algebra $\mathfrak{g}$, which is the tangent space of the group at the identity element. The connection between a Lie group and its Lie algebra is made through the exponential ExpMap and logarithmic map LogMap which link elements of the Lie algebra to elements of the Lie group and vice versa:

$$\text{ExpMap} : \mathfrak{g} \to \mathcal{G}\,, \quad \text{LogMap} : \mathcal{G} \to \mathfrak{g}\,.$$

The orthogonal groups $SO(N)$ are the Lie groups of orthogonal matrices with a determinant of 1. Regarding the insertion task, the groups $SO(N)$ with $O \in 2, 3$ are relevant, as they represent the rotations in 2D and 3D. These rotations can be transformed into the tangent space $\mathfrak{so}(N)$ - the Lie algebra. The Lie groups $SE(N)$ ('special Euclidean group') can be used to include the position information by representing the group of the homogeneous transformation matrix. Here, the rotation matrix and the position vector are combined in the homogeneous transformation matrices. The corresponding Lie algebras/tangent vector spaces are described as $\mathfrak{se}(N)$.

3.3.2. The Stable Vector Field

In the MSVF policy, the stability will be implemented in a latent Lie algebra. The simplest form of an ordinary differential equation (ODE) (Lang, 2001b)

$$\dot{\boldsymbol{z}} = -\boldsymbol{z}$$

will be used. The equilibrium point of this system is easily identified with $\bar{\boldsymbol{z}} = 0$. This ODE can be written as $\dot{\boldsymbol{z}} = V(\boldsymbol{z})$ by defining the vector field $V(\boldsymbol{z}) = -\boldsymbol{z}$.

3.3.3. The MSVF-Policy

To form the MSVF policy the concepts of the previous subsections are combined, following the work of Hellwig and Urain et al.. Figure 1 visualizes the structure, described in the following.
For simplification purposes, we assume that the target position and orientation of the insertion are centered in the reference frame of the robot base. By applying transformations to other frames, this assumption can be surpassed. Furthermore, only the 3D case is considered, which can be simplified easily to 2D.

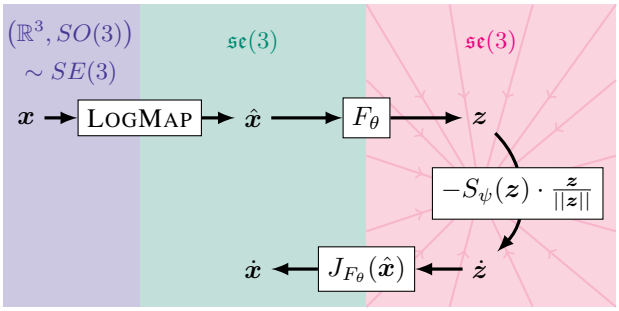

Figure 1. The structure of the MSVF policy. For simplification purposes $\boldsymbol{x}_{\text{GOAL}} = 0$ is assumed.

The input for the policy is the robot's end effector state $\boldsymbol{x} \in \mathcal{X} = \big(\mathbb{R}^3, SO(3)\big)$ or $SE(3)$. This observation gets mapped into the Lie algebra $\hat{\boldsymbol{x}} = \text{LogMap}(\boldsymbol{x}) \in \hat{\mathcal{X}} = \mathfrak{se}(3)$ - the tangent space of the observation space around its origin.
In the next step, the diffeomorphism φ is applied to reach the latent lie algebra $\mathcal{Z} = \mathfrak{se}(3)$. It is implemented as the solution $\varphi = F$ of a neural ODE (NODE) (Chen et al., 2018). The NODE is defined as a neural network H_θ that maps a state $\boldsymbol{s}$, to its time derivative: $H_\theta(\boldsymbol{s}, t) = \dot{\boldsymbol{s}}$. This is applied to the MSVF's structure in the following way. The element $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}$ is defined as the start state $\boldsymbol{s}(0) = \boldsymbol{s}(t = 0)$ and the latent space representation $\boldsymbol{z}$ is defined to be the state $\boldsymbol{s}_1$ at $t = 1$. Integrating out the NODE results in the mapping

$$\boldsymbol{z} = F_\theta(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{x}} + \int_0^1 H_\theta\big(\boldsymbol{s}(t), t\big)\ \mathrm{d}t\,. \tag{3}$$

Due to the structure of the NODE, the inverse mapping can be found by integrating backward in time

$$\hat{\boldsymbol{x}} = F_\theta^{-1}(\boldsymbol{z}) = \boldsymbol{z} + \int_1^0 H_\theta\big(\boldsymbol{s}(t), t\big)\ \mathrm{d}t\,,$$

keeping in mind that $\boldsymbol{s}(1) = \boldsymbol{z}$. Solving the ODE can be done, e.g., with the Euler method (Euler, 1792). This shows the bijectivity of F_θ, even though the inverse mapping is not needed, as it will be seen shortly. The mapping requires some further handling of edge cases. For details, the reader is referred to (Hellwig, 2023).
An important property is not considered at this point. The target point in $\hat{\mathcal{X}}$ is its origin $\hat{\boldsymbol{x}}_H = \mathbf{0}$. To keep track of this and not shift the stable point, the origin of $\hat{\mathcal{X}}$ must get mapped to the origin of $\mathcal{Z}$: $\mathbf{0} = \boldsymbol{z}_H \overset{!}{=} F_\theta(\hat{\boldsymbol{x}}_H)$. This will later be achieved by adding a fix-center-loss $\mathcal{L}_{\text{FCL}}$ while learning.
In the latent Lie-algebra, the stable vector field $\dot{\boldsymbol{z}} =$

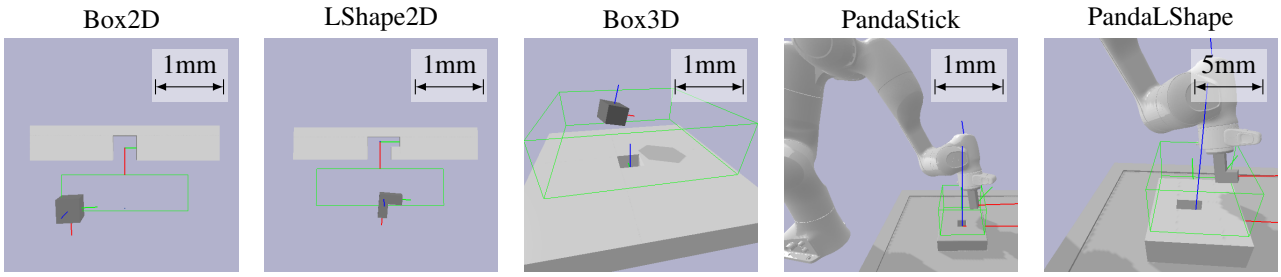

Figure 2. The different environments for the insertion task - starting with minimal complexity on the left towards higher complexities on the right. The green bounding boxes represent the area for possible initial states. In the 3D environments, we sample from the boxes' surfaces uniformly. The rotation is sampled from $\texttt{uniform}[-20°, 20°]$ per axis.

$-V(\boldsymbol{z}) = -\boldsymbol{z}$ is defined. Using this structure would impose very small control signals close to the target position as $\dot{\boldsymbol{z}} \to 0$ for $\boldsymbol{z} \to 0$. To counteract this, a scaling network $s = S_\psi(\boldsymbol{z}) > 0$ is learned, which scales the normalized vector field (Hellwig, 2023):

$$\dot{\boldsymbol{z}} = -s\frac{\boldsymbol{z}}{||\boldsymbol{z}||}\,. \tag{4}$$

$\dot{\boldsymbol{z}}$ needs to be mapped back to the observation space. In the first step, the mapping into $\mathcal{Z}$ is done. Revisiting the mapping from $\hat{\mathcal{X}}$ to $\mathcal{Z}$ shows that $\boldsymbol{z} = F_\theta(\hat{\boldsymbol{x}})$ and hence $\dot{\boldsymbol{z}} = \frac{\mathrm{d}}{\mathrm{d}t}F_\theta(\hat{\boldsymbol{x}})$. Evaluating this derivative results in

$$\dot{\boldsymbol{z}} = \frac{\partial}{\partial\hat{\boldsymbol{x}}}F_\theta(\hat{\boldsymbol{x}}) \cdot \frac{\mathrm{d}}{\mathrm{d}t}\hat{\boldsymbol{x}} = J_{F_\theta}(\hat{\boldsymbol{x}})\dot{\boldsymbol{x}}\,,$$
$$\dot{\boldsymbol{x}} = J_{F_\theta}(\hat{\boldsymbol{x}})^{-1}\dot{\boldsymbol{z}}\,.$$

This mapping out of the latent space can be seen as a controlled deformation process of the vector field $V(\boldsymbol{z})$, which warps the simple structure into an arbitrary complex vector field with the same equilibrium point.
The derived $\dot{\boldsymbol{x}}$ is a velocity, but can also be interpreted as the desired change in position by integrating it for one time step to get $\Delta\boldsymbol{x} = \dot{\boldsymbol{x}}\Delta t$, where Δt can be the policy frequency. The EXPMAP is later applied internally in the operational space controller where the axis–angle representation (Caccavale et al., 1998) is used in combination with the Rodrigues' rotation formula (Mebius, 2007). This implicitly includes a map from the Lie algebra $\mathfrak{se}(3)$ to its Lie group $SE(3)$.

3.4. Reinforcement Learning Algorithms

The goal of a reinforcement learning agent is to solve a Markov Decision Process (MDP). This translates into computing an optimal policy that maximizes the expected sum of discounted rewards. In this work, two deep actor-critic methods are used to obtain an optimal policy.
The first is Proximal Policy Optimization (PPO) (Schulman et al., 2017), which is an on-policy algorithm that combines Policy Gradient methods (Peters & Schaal, 2006a) with Trust Region Policy Optimization (TRPO) (Schulman et al., 2015). Although the PPO approach is relatively simple, it performs quite well in most scenarios.
The second option chosen for this work is the Soft Actor-Critic algorithm (SAC) (Haarnoja et al., 2018). In contrast to PPO, it is an off-policy algorithm. SAC optimizes not only the expected discounted return but also an entropy-regularized objective that encourages the agent to keep exploring.

4. Implementation

The following section outlines the implementation of the environments with varying levels of complexity, along with the policies applied to perform the insertion.

4.1. Environments and Tasks Descriptions

The different learning approaches are compared in 5 simulated insertion tasks with increasing structural complexity, see Figure 2. These are implemented using the PYBULLET (Coumans & Bai, 2016–2021) simulator, while the RL algorithms are implemented using MUSHROOMRL (D'Eramo et al., 2021).

In Box2D and LShape2D environments, movement along the x and y-axes and rotation around the z-axis is possible. The Box2D environment is straightforward. The actor only has to learn to move a square box into a hole with a 1mm tolerance. In the LShape2D environment, the hole and the peg are no longer square but L-shaped, and with identical (1 mm) wiggle-room, the peg has to be moved inside the hole and then to the side.
In 3D space, the Box3D environment is a generalization of the Box2D, where one more dimension is used for translation and two new dimensions are used for rotation. Two simulated environments feature the Franka Panda robot, which additionally constrains the movement of the peg. In the PandaStick task, the robot has to insert a stick, which here is implemented as an attached peg, into a hole, giving 1mm of wiggle room in each dimension. The PandaLShape

task increases the complexity by presenting a 3D version of the L-shaped insertion task with a 5mm tolerance.
In all environments, including the Franka Panda robot, the peg's movement is defined and learned in task space, and the necessary motor commands are calculated through the OSC. Thereby, the robot's size and its joint limits present natural boundaries of the peg's movement.
(The long-term goal is to perform experiments on the real robot. The insertion-specific parts (peg and hole) will be printed in 3D. With this in sight, the parts are modeled as stiff and with a friction coefficient of $\mu = 0.4$ (Perepelkina et al., 2017; Pawlak, 2018).)

4.1.1. The Observation Space

The observation space $\boldsymbol{x} \in \mathcal{S}$ is the same for all tasks. It contains the current peg position $\boldsymbol{p} \in \mathbb{R}^3$ and orientation $\boldsymbol{R} \in SO(3)$ provided as a flattened rotation matrix $\tilde{\boldsymbol{R}} \in \mathbb{R}^9$. Given the structure of our environments and that all matrix entries in $\boldsymbol{R}$ are in the range of $[-1, 1]$, no further scaling of the observations is needed. It also should be mentioned that $\boldsymbol{x} = (\boldsymbol{p}, \tilde{\boldsymbol{R}}) \in \mathbb{R}^{12}$ for all environments, independently of representing a 2D or 3D task.

4.1.2. The Action Space

The action space $\mathcal{A}$ is split into two parts. It comprises the proposed change in position $\Delta\boldsymbol{p}$ and the desired change in orientation $\Delta\boldsymbol{r}$, as $\Delta\boldsymbol{x} = (\Delta\boldsymbol{p}, \Delta\boldsymbol{r})$. $\Delta\boldsymbol{p}$ lives in $\mathbb{R}^2$ or $\mathbb{R}^3$ depending on the dimension of the environment. For 2D environments, $\Delta\boldsymbol{r}$ is a scalar value describing the proposed angle change in orientation around the z-axis. In 3D $\Delta\boldsymbol{r}$ is a rotation vector in $\mathfrak{so}(3)$ that describes the rotation as a rotation vector (the axis-angle representation). Its direction represents the axis of rotation, and its magnitude is the rotation angle in radians.
The actions are clipped so that the change in position is limited to $\pm 0.1m$ and the rotation to $\pm 45°$. Using this action output, the desired location for the operational space controller is updated to $\boldsymbol{x}_{\text{DES}} = \boldsymbol{x} + \Delta\boldsymbol{x}$. The controller tries to reach $\boldsymbol{x}_{\text{DES}}$ following the Equations of the Section 3.1.
The policy updates $\boldsymbol{x}_{\text{DES}}$ at 20 Hz, while the low-level controller runs at 240 Hz, giving the OSC twelve time steps to reach the desired state before it is updated.

4.1.3. The Reward Function

The reward function penalizes any distance between the goal state and the current state $\delta\boldsymbol{x} = (\delta\boldsymbol{p}, \delta\boldsymbol{r}) = \boldsymbol{x}_{\text{GOAL}} \ominus \boldsymbol{x}_{\text{CURRENT}}$ and is identical for each environment. Calculating $\delta\boldsymbol{p}$ for the translation component is done by subtracting vectors. The rotational difference $\delta\boldsymbol{r}$ is computed using the axis-angle-feedback (Caccavale et al., 1998) in combination with Rodrigues' rotation formula.

A typical reward is the negative L2-norm $||\delta\boldsymbol{x}||_2^2$. However, as shown by (Levine et al., 2015), using a funnel-like shaped penalty term is better, which penalizes small errors in position more. This enforces a policy of higher accuracy by defining the positional cost via

$$c_{\text{POS}}(\boldsymbol{p}) = w \cdot ||\delta\boldsymbol{p}||_2^2 + v \cdot \log\left(||\delta\boldsymbol{p}||_2^2 + \alpha\right) .$$

w, v are hyper-parameters to weight the two terms (L2, and Lorentzian ρ) against each other. α defines the funnel-width (Levine et al., 2015).
The total reward function is composed by summing up the adapted positional cost and an L2-penalty for rotational distance as well as an L2-loss for the joint velocities to strive for smoother movements:

$$r(\boldsymbol{x}) = -\omega_p \cdot c_{\text{POS}}(\boldsymbol{x}) - \omega_r \cdot ||\delta\boldsymbol{r}||_2^2 - \omega_{\dot{q}} \cdot ||\dot{\boldsymbol{q}}||_2^2 . \quad (5)$$

The positive weights ω balance the effect of the individual terms. Settings with $\omega_r > \omega_{\dot{q}} \geq \omega_p$ are found to work best for the insertion tasks, but depending on the number of degrees-of-freedom and number of robot joints, individual adaptions are needed.
Furthermore, two additional terms are added to the reward. $r = 1$ for reaching the goal and $r = -100$ for going out of bounds.

4.1.4. The Initial State Distribution

The initial state distribution is similar for the 2D and 3D environments. We draw a box above the holes as shown in Figure 2 and sample the initial state based on these boxes. For the 2D environments, we decided to sample the initial coordinates inside the boundaries of these boxes and rotate the peg by a value between $-20°$ and $+20°$. In 3D the initial states are sampled uniformly from the box's surfaces and a rotation of $\pm 20°$ is added for each axis. This approach works towards the following control strategy after training: Using a simple control signal to guide the end effector towards the region of insertion before the learned policy takes control.

4.2. The Base Policy π_{BASE}

Residual learning, as introduced in Section 3.2, is based on a policy π_{BASE} that encodes some prior knowledge about the task and should provide an initial guidance for the agent.

In the setup for the insertion task, such a simple policy is used as the linear attractor. It computes the action $\Delta\boldsymbol{x}_{\text{BASE}} = \pi_{\text{BASE}}(\boldsymbol{x})$ in such a way that $\boldsymbol{x} + \Delta\boldsymbol{x}_{\text{BASE}} = \boldsymbol{x}_{\text{GOAL}}$ and is then clipped in the same manner as described in Section 4.1.2.

4.3. The Policy Components

PPO and SAC are both Deep Actor-Critic approaches. The actor and the critic functions are modeled using fully connected neural networks (FCNNs) with two hidden layers

with 256 features each. The inner nodes apply a RELU activation (Agarap, 2018). In SAC, the policy is modeled as a Gaussian distribution, with the mean and variance components, each generated by separate FCNNs of identical structure.

4.4. The Manifold Stable Vector Field

As outlined in Section 3.3.3, the MSVF consists of several components, and their implementation is described below.

4.4.1. THE DIFFEOMORPHISM F_θ

The diffeomorphism is implemented as the solution of an NODE (Equation 3). The network H_θ that models the NODE is implemented as FCNN as well. Again, the structure shows two hidden layers with 256 features each in combination with the LEAKYRELU-activation after the first two. The integration from Equation 3 - that performs the mapping to the latent Lie-algebra - is approximated using Euler integration (Euler, 1792).
As described in Section 3.3.3 it is important that the origin $\hat{\boldsymbol{x}}_H$ is mapped to $\boldsymbol{z}_H$. To achieve this, the fix-center-loss $\mathcal{L}_{\text{FCL}}\big(F_\theta(\hat{\boldsymbol{x}}_H)\big) = w \cdot ||F_\theta(\hat{\boldsymbol{x}}_H)||_2^2 + v \cdot \log\left(||F_\theta(\hat{\boldsymbol{x}}_H)||_2^2 + \alpha\right)$ is added to the objectives of SAC and PPO during training. Additionally, this loss is used to pre-train the diffeomorphism before starting to learn in the environment. This is done to guarantee a stable policy from the start.

4.4.2. THE SCALING NETWORK S_ψ

The scaling network is important to prevent vanishing control signals for states close to the goal state. It is again implemented as an FCNN of the already mentioned structures, with the only difference being the activation of the final output: applying the SOFTPLUS guarantees positive scaling values ($s > 0$).

5. Experimental Results

The three different policy approaches - namely *(i)* multilayer perceptron (MLP)/unstructured ●, *(ii)* residual with linear attractor ● and *(iii)* MSVF ● - are evaluated against each other. While running the training with both learning approaches - PPO and SAC - the following evaluation is based solely on the results with PPO, as SAC showed stability problems, especially in environments of higher complexity. As an example of a successfully learned behavior, Figure 4 shows the path of insertion in the Panda-LShape environment.

5.1. General Observations

Each combination of policy and environment is trained across five different seeds. The success rate and the number of steps required to reach the goal are measured by evaluating the performance across ten different starting states for each scenario. The results of these evaluations are presented in Figure 3.

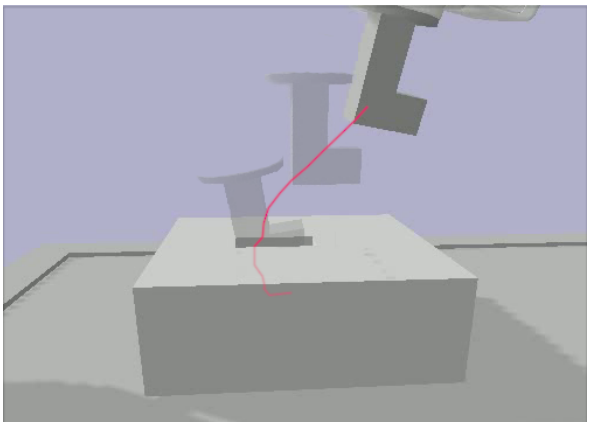

Figure 4. Insertion trajectory of the MSVF policy after training with PPO. The policy moves the peg over the opening first. Once at the bottom, the LShape is pushed forward.

For the simple two-dimensional environments, we can observe a clear order of policies in terms of convergence efficiency. The MSVF converges to a success rate of 100% after 50k steps or less, followed by the residual policy and then the unstructured policy. Due to the pre-training of the MSVF with $\mathcal{L}_{\text{FCL}}$, the origins of the two Lie algebras align from the very first training iteration, enabling successful insertions from the outset. The residual policy provides some beneficial information for the agent compared to the unstructured policy.
In the case of the Box3D environment, too, the MSVF converges early. However, this time, the residual policy outperforms the two other approaches. The latent stable vector field of the MSVF is now defined in six dimensions (instead of three for the 2D case), which makes it more difficult for the diffeomorphism to warp the space.
When analyzing the measured number of steps to terminate, we can see that — after some time - all three policies perform equally, with the residual policy having a slight advantage in all cases.

5.2. Performance in the Panda Environments

For the simple PandaStick environment, the performance ranking is the same as the one for the Box3D insertion. This is expected as the two tasks pose the same structure from the policy's viewpoint. The underlying OSC control handles the computation of the fitting torque signals. Again the MSVF performs slightly slower regarding the two metrics at hand, compared to the residual policy. However, the difference between the two is not significant. Accepting a slightly longer training time in exchange for a guarantee of stability can be

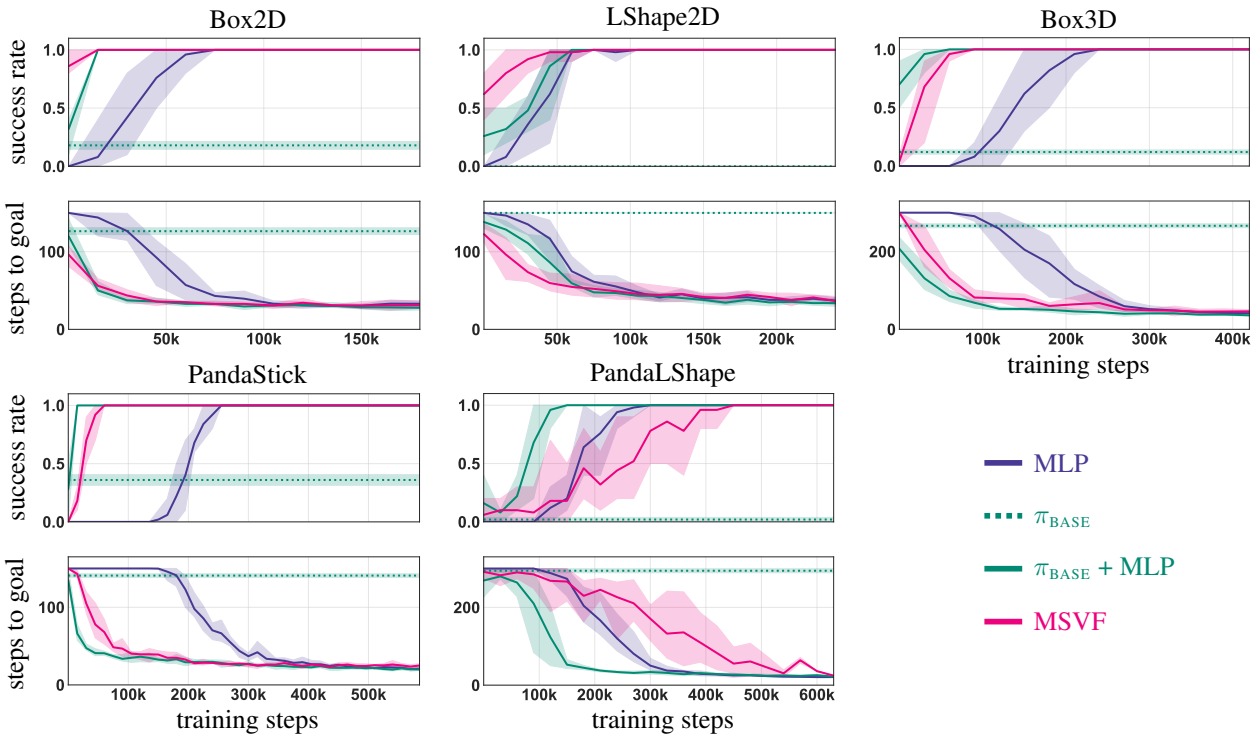

Figure 3. Success rate and steps-to-goal for training the different policies in all 5 environments. To indicate the difficulty of the task, the performance of the linear attractor π_{BASE} is displayed as well. Please notice that the success rate of π_{BASE} is zero for the LShape2D environment and therefor is not visible in the plot.

very useful, depending on the task and application domain. The PandaLShape exposes some difficulties for the MSVF. The distortion of the vector field must be much greater to achieve a successful insertion. In combination with the higher dimension of the latent space, this requires longer training. On top of that, learning is less stable - but this might be resolved with further hyperparameter search. Nevertheless, the guaranteed stability of the MSVF should also be kept in mind here.

5.3. Analyzing the Induced Vector Field

The induced vector fields are analyzed exemplary in the LShape2D environment, see Figure 5.
The MSVF points towards the goal position before the first training step because of the pre-training with the fix-center-loss $\mathcal{L}_{\text{FCL}}$. The residual policy is randomly initialized, but the influence of π_{BASE} is visible: it drives the peg towards the goal. Initially, the unstructured policy vector field even points further away from the goal.
After training, it can be seen that all policies learned a vector field that solves the task. Notice that all policies do not prevent contact with the environment.

If desired, increasing friction or applying a contact penalty in the reward function could prevent this. The unstructured

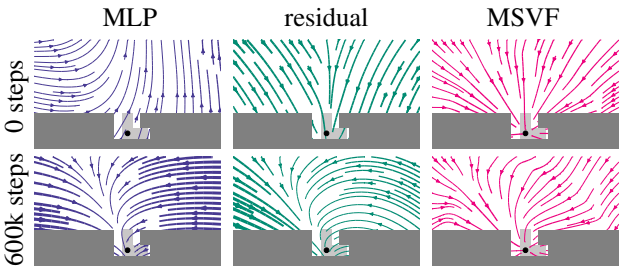

Figure 5. Translational Vector Fields induced by the policies in the LShape2D environment. The line thickness is proportional to the strength of the field. *Top row*: Before training. *Bottom row:* After training.

MLP policy and the residual policy are very identical for the LShape. Both learned to lead the peg correctly into the hole and then push it downwards onto the hole's ground. The MSVF's policy also leads the peg to the hole, but instead of just pushing the peg downwards, the policy learns to converge to the target point inside the hole.

5.4. Generalization

To test generalization, the initial state's region is shifted above the region used during training ($\mathcal{T}$). The orientations for the initial states are sampled - per axis - from the ranges

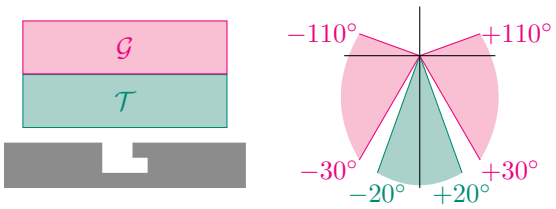

Figure 6. Exemplary visualization of the generalization test. Instead of the green starting ranges $\mathcal{T}$ of the training, the policies are tested starting in regions $\mathcal{G}$ which are placed above $\mathcal{T}$ and are rotated more.

Table 1. Results (success rate) of the generalization experiments. Each setup ran for 100 epochs. The success rate is evaluated for the initial state ranges $\mathcal{T}$ and $\mathcal{G}$.

	Box2D		LShape2D		Box3D		PandaStick		PandaLShape	
	$\mathcal{T}$	$\mathcal{G}$	$\mathcal{T}$	$\mathcal{G}$	$\mathcal{T}$	$\mathcal{G}$	$\mathcal{T}$	$\mathcal{G}$	$\mathcal{T}$	$\mathcal{G}$
MLP	1.0	1.0	1.0	1.0	1.0	1.0	1.0	**0.99**	1.0	**0.97**
residual	1.0	1.0	1.0	1.0	1.0	1.0	1.0	**0.99**	1.0	**0.98**
MSVF	1.0	1.0	1.0	1.0	1.0	1.0	1.0	**0.97**	1.0	**0.94**

$\pm[30°, 110°]$. Figure 6 visualizes the difference between the training $\mathcal{T}$ and testing region $\mathcal{G}$. Table 1 presents the results.

In the Box2D, LShape2D, and Box3D environments, all policy classes show a very good generalization: they achieve a success rate of 1.0 for both regions $\mathcal{T}$ and $\mathcal{G}$. This could be expected, as the policies are continuous mappings and show the good generalization properties of neural networks. In the Panda environments the success rate drops for all policies, but only by a few percentage points. Analyzing the failure cases in detail showed that these cases got all stuck in the joint limits of the robot. This happens because the OSC was kept as simple as possible without any null space task. Overall these runs also manifest the good generalization properties of neural networks in general.

6. Conclusion & Outlook

The paper compares three different policy classes to solve insertion tasks. The residual policy and the MSVF both include prior knowledge about this task and outperform the MLP-policy in all low-complexity environments. In the environment with the highest complexity - the PandaLShape - the residual policy still performs the best. Here, the MSVF policy performance shows problems. The dimension of the latent vector field is large, and the diffeomorphism has to apply a strong distortion to the vector field, but with longer training, it still solves the task.

In contrast to (Hellwig, 2023) it was able to show that the MSVF alone can solve non-linear insertion tasks - whiteout the need for a further residual policy part. This emphasizes the potential of this policy structure further.

In tasks like the insertion of PCB components (Luo et al., 2024), such a strong warping is not needed, and the stability guarantees of the MSVF can be the decisive factor in deciding on it. In these cases, the training time increases minimally compared to the residual policy, and the price paid for stability guarantees is more than acceptable. We believe the MSVF is always worth a try. Only if the insertion task is found to be too complex, other options like the residual policy, should be considered.

Enhancing the adaptability of the impedance controller by integrating force observations and learning its gain parameters (Luo et al., 2024) are exciting research opportunities. This approach may enable the controller to encode richer structural information about the environment, potentially leading to a more nuanced understanding and interaction. Additionally, allowing greater degrees of freedom for the policy could improve its ability to handle complex environments.

Given the substantial amount of contact between peg and base observed (see Figure 5), managing these interactions effectively is crucial. One potential avenue for further refinement involves incorporating a penalty for large forces into the reward function. Especially in scenarios where minimizing force is essential or where excessive force might be undesirable.

References

Agarap, A. F. Deep Learning using Rectified Linear Units (ReLU). *CoRR*, 2018. doi:https://doi.org/10.48550/arXiv.1803.08375.

Beltran-Hernandez, C. C., Petit, D., Ramirez-Alpizar, I. G., and Harada, K. Variable Compliance Control for Robotic Peg-in-Hole Assembly: A Deep Reinforcement Learning Approach. *CoRR*, 2020. doi:https://doi.org/10.3390/app10196923.

Caccavale, F., Natale, C., Siciliano, B., and Luigi, V. Resolved-acceleration Control of Robot Manipulators: A Critical Review with Experiments. *Robotica*, 16:565–573, 1998. doi:10.1017/S0263574798000290.

Chang, C., Haninger, K., Shi, Y., Yuan, C., Chen, Z., and Zhang, J. Impedance Adaptation by Reinforcement Learning with Contact Dynamic Movement Primitives. *cs.RO*, 2022. doi:10.48550/arXiv.2203.07191.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural Ordinary Differential Equations. *CoRR*, 2018. doi:10.48550/arXiv.1806.07366.

Coumans, E. and Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Davchev, T., Luck, K. S., Burke, M., Meier, F., Schaal, S., and Ramamoorthy, S. Residual Learning from Demonstration. *CoRR*, 2020. doi:10.48550/arXiv.2008.07682.

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. Mushroomrl: Simplifying reinforcement learning research. *Journal of Machine Learning Research*, 22(131):1–5, 2021. URL http://jmlr.org/papers/v22/18-056.html.

Dong, Y., Ren, T., Wu, D., and Chen, K. Compliance Control for Robot Manipulation in Contact with a Varied Environment Based on a New Joint Torque Controller. *Journal of Intelligent & Robotic Systems*, 99(1):79–90, 2020. ISSN 1573-0409. doi:10.1007/s10846-019-01109-8.

Euler, L. *Institutiones calculi integralis*. Academia Imperialis Scientiarum, 1792.

Fan, Y., Luo, J., and Tomizuka, M. A Learning Framework for High Precision Industrial Assembly. *CoRR*, 2018. doi:10.48550/arXiv.1809.08548.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *CoRR*, 2018. doi:10.48550/arXiv.1801.01290.

Hall, B. *Lie Groups, Lie Algebras, and Representations*, pp. 333–366. Springer Cham, 2013. ISBN 978-1-4614-7115-8. doi:10.1007/978-1-4614-7116-5_16.

Hellwig, S. Residual Reinforcement Learning with Stable Priors. *TU Darmstadt*, 2023.

Inoue, T., Magistris, G. D., Munawar, A., Yokoya, T., and Tachibana, R. Deep Reinforcement Learning for High Precision Assembly Tasks. *CoRR*, 2017. doi:10.48550/arXiv.1708.04033.

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E., and Levine, S. Residual Reinforcement Learning for Robot Control. *CoRR*, 2018. doi:10.48550/arXiv.1812.03201.

Khader, S. A., Yin, H., Falco, P., and Kragic, D. Stability-Guaranteed Reinforcement Learning for Contact-rich Manipulation. *CoRR*, abs/2004.10886, 2020. doi:10.48550/arXiv.2004.10886.

Khader, S. A., Yin, H., Falco, P., and Kragic, D. Learning Stable Normalizing-Flow Control for Robotic Manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1644–1650. IEEE Press, 2021. doi:10.1109/ICRA48506.2021.9562071.

Khan, S. G., Herrmann, G., Al Grafi, M., Pipe, T., and Melhuish, C. Compliance Control and Human–Robot Interaction: Part 1 — Survey. *International Journal of Humanoid Robotics*, 2014. doi:10.1142/S0219843614300013.

Khatib, O. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3:43–53, 1987. doi:10.1109/JRA.1987.1087068.

Lang, S. *Fundamentals of Differential Geometry*. Graduate Texts in Mathematics. Springer New York, 2001a. ISBN 9780387985930.

Lang, S. *Fundamentals of Differential Geometry*. Graduate Texts in Mathematics. Springer New York, 2001b. ISBN 9780387985930.

Lee, M. A., Zhu, Y., Srinivasan, K., Shah, P., Savarese, S., Fei-Fei, L., Garg, A., and Bohg, J. Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks. *CoRR*, 2018. doi:10.48550/arXiv.1810.10191.

Lee, M. A., Florensa, C., Tremblay, J., Ratliff, N. D., Garg, A., Ramos, F., and Fox, D. Guided Uncertainty-Aware Policy Optimization: Combining Learning and Model-Based Strategies for Sample-Efficient Policy Learning. *CoRR*, 2020. doi:10.48550/arXiv.2005.10872.

Levine, S., Wagener, N., and Abbeel, P. Learning Contact-Rich Manipulation Skills with Guided Policy Search. *CoRR*, 2015. doi:10.48550/arXiv.1501.05611.

Luo, J., Solowjow, E., Wen, C., Ojea, J., and Agogino, A. Deep Reinforcement Learning for Robotic Assembly of Mixed Deformable and Rigid Objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2062–2069, 10 2018. doi:10.1109/IROS.2018.8594353.

Luo, J., Hu, Z., Xu, C., Tan, Y. L., Berg, J., Sharma, A., Schaal, S., Finn, C., Gupta, A., and Levine, S. Serl: A software suite for sample-efficient robotic reinforcement learning, 2024.

Mebius, J. E. Derivation of the Euler-Rodrigues formula for three-dimensional rotations from the general formula for four-dimensional rotations, 2007.

Pawlak, W. Wear and coefficient of friction of PLA - Graphite composite in 3D printing technology. In *Engineering Mechanics*, 2018. doi:10.21495/91-8-649.

Perepelkina, S., Kovalenko, P., Pechenko, R., and Makhmudova, K. Investigation of Friction Coefficient of Various Polymers Used in Rapid Prototyping Technologies with Different Settings of 3D Printing. *Tribology in Industry*, 39:519–526, 2017. doi:10.24874/ti.2017.39.04.11.

Peters, J. and Schaal, S. Policy Gradient Methods for Robotics. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 2219 – 2225, 11 2006a. doi:10.1109/IROS.2006.282564.

Peters, J. and Schaal, S. Learning operational space control. *Proceedings of Robotics: Science and Systems (RSS), Philadelphia, PA*, 01 2006b.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust Region Policy Optimization. *CoRR*, 2015. doi:10.48550/arXiv.1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. *CoRR*, 2017. doi:10.48550/arXiv.1707.06347.

Silver, T., Allen, K. R., Tenenbaum, J., and Kaelbling, L. P. Residual Policy Learning. *CoRR*, 2018. doi:10.48550/arXiv.1812.06298.

Suomalainen, M., Karayiannidis, Y., and Kyrki, V. A survey of robot manipulation in contact. *Robotics and Autonomous Systems*, 156, 2022. ISSN 0921-8890. doi:10.1016/j.robot.2022.104224.

Tang, T., Lin, H.-C., and Tomizuka, M. A Learning-Based Framework for Robot Peg-Hole-Insertion. In *Dynamics Systems and Control Conference*, 10 2015. doi:10.1115/DSCC2015-9703.

Tang, T., Lin, H.-C., Zhao, Y., Fan, Y., Chen, W., and Tomizuka, M. Teach industrial robots peg-hole-insertion by human demonstration. In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 488–494. IEEE Press, 2016. doi:10.1109/AIM.2016.7576815.

Thomas, G., Chien, M., Tamar, A., Ojea, J. A., and Abbeel, P. Learning Robotic Assembly from CAD. *CoRR*, 2018. doi:10.48550/arXiv.1803.07635.

Urain, J., Tateo, D., and Peters, J. Learning Stable Vector Fields on Lie Groups. *cs.RO*, 2022. doi:10.48550/arXiv.2110.11774.

Wan, A., Xu, J., Chen, H., Zhang, S., and Chen, K. Optimal Path Planning and Control of Assembly Robots for Hard-Measuring Easy-Deformation Assemblies. *IEEE/ASME Transactions on Mechatronics*, 22:1600–1609, 2017. doi:10.1109/TMECH.2017.2671342.