Tolerance-Guided Policy Learning for Adaptable and Transferrable Delicate Industrial Insertion

Boshen Niu[*], **Chenxi Wang**[*], **Changliu Liu**
Carnegie Mellon University, USA
bniu, chenxiwa, cliu6@andrew.cmu.edu

Abstract: Policy learning for delicate industrial insertion tasks (e.g., PC board assembly) is challenging. This paper considers two major problems: how to learn a diversified policy (instead of just one average policy) that can efficiently handle different workpieces with minimum amount of training data, and how to handle defects of workpieces during insertion. To address the problems, we propose tolerance-guided policy learning. To encourage transferability of the learned policy to different workpieces, we add a task embedding to the policy's input space using the insertion tolerance. Then we train the policy using generative adversarial imitation learning with reward shaping (RS-GAIL) on a variety of representative situations. To encourage adaptability of the learned policy to handle defects, we build a probabilistic inference model that can output the best inserting pose based on failed insertions using the tolerance model. The best inserting pose is then used as a reference to the learned policy. This proposed method is validated on a sequence of IC socket insertion tasks in simulation. The results show that 1) RS-GAIL can efficiently learn optimal policies under sparse rewards; 2) the tolerance embedding can enhance the transferability of the learned policy; 3) the probabilistic inference makes the policy robust to defects on the workpieces.

Keywords: Insertion, Tolerance, Reinforcement Learning

1 Introduction

Although robotics have been widely applied in manufacturing, it remains challenging for robots to handle delicate industrial insertion [1, 2, 3]. This paper mainly considers insertion tasks on a small scale with multiple contact points. For example, the assembly of an IC socket that has multiple pins on a printed circuit board, the plug of a USB port, etc. Conventional model-based methods for insertion are able to achieve high performance on modeled tasks, but require case-by-case tuning and have poor generalizability [4]. As the emphasis of electronic assembly shifts from massive production to massive customization, the assembly lines are made more and more flexible, which requires robots to be equipped with general insertion skills that cover a wide variety of tasks. Learning-based methods are promising for robots to learn those skills [5, 6, 7, 8]. Nonetheless, there are two major challenges for policy learning in delicate industrial insertion tasks. The first is how to learn a diversified policy (instead of just one average policy) that can efficiently handle different workpieces with minimum amount of training data. The second is how to handle defects of workpieces during insertion, e.g., bent pins.

This paper proposes a tolerance-guided policy learning method to address the problems as a way to leverage models (i.e., tolerance) in the policy learning. Tolerance refers to the amount of misalignment error and force error that is admissible so that the insertion can still succeed. Every insertion task has a tolerance. Different insertion tasks may have the same amount of tolerance. We can explicitly explore the similarities in the tolerance to transfer learned policies to different insertion tasks. The (nominal) tolerance can be directly computed from the CAD models and the specs of the insertion tasks. We call the mathematical description of the tolerance as a *tolerance model*. To encourage transferability of the policy, we augment the policy input space to include the tolerance model and train the policy on a variety of representative tasks. To remove redundant information

[*]These authors contributed equally to this article. The order of authorship was determined by a coin flip.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

in the tolerance model, we encode the tolerance model into a low dimensional vector using auto-encoder. With the tolerance embedding, the next problem is how to efficiently learn the policy as the insertion tasks are with sparse rewards. By leveraging imitation learning and reinforcement learning, we propose generative adversarial imitation learning with reward shaping (RS-GAIL) that starts the learning by mimicking a suboptimal expert and gradually converges to a policy that is optimal with respect to the environment reward. The expert effectively guides the initial exploration. In addition, to encourage adaptability of the learned policy to handle defects, we build a probabilistic inference model that can output the best inserting pose based on failed insertions. The probabilistic inference starts from the nominal tolerance model (computed from CADs) and will gradually learn the true tolerance model (with defects). The best inserting pose based on the true tolerance model is then used as a reference to the learned policy. The contributions of the paper include:

- introduction of RS-GAIL to efficiently train insertion policies in sparse reward environments;
- policy generalization to different workpieces (with different number of pins and different geometry of the pins) using tolerance embedding;
- introduction of probabilistic inference of the defects on workpieces and the consequent optimal insertion points to make the policy robust to defects.

2 Related Work

Learning Framework for Insertion An insertion policy takes the state (e.g., pin/hole relative pose, contact force) as input, and outputs an action for the robot to move the workpiece. Inoue et al [6] used LSTM based policy networks to map the current contact force and peg position to desired force and peg position. Luo et al [7] used model-based RL algorithm and iLQG to train a torque controller and used a neural network to process force/torque readings as reference inputs to the controller. These works focus on single task without workpiece variations, hence the learned policy cannot generalize to new workpieces. Using meta-reinforcement learning [8, 9], the robot can learn skills on new tasks with a small amount of demonstrations. However, since the specifications of workpieces (e.g., number of pins, shape of pins) have large variations, it is intractable to get demonstrations for all new workpieces. Our proposed method, by encoding task specifications into low dimensional parameters and embedding these parameters as another input modality of the policy network, can generalize to new workpieces without training or finetuning using new data.

Learning Algorithm To learn a policy, one can use either imitation learning (IL) or reinforcement learning (RL). In IL, the agent learns the policy by mimicking expert's behavior. Among common IL algorithms, behavior cloning (BC) [10] requires large amount of expert demonstration data; dataset aggregation (DAGGER) [11] requires access to the expert during rollout. Generative adversarial imitation learning (GAIL) [12] can overcome these problems. It includes a discriminator to measure similarity between the student policy and the expert policy, and uses the similarity as criteria to improve the student policy (similar to a RL problem). However, since IL training drives the state-action distribution of student policy towards that of the expert, it is impossible for the student to outperform the expert. In RL, the agent learns the policy by interacting with the environment according to a reward function which needs to be carefully engineered. For tasks with sparse rewards (such as the insertion task considered in this paper), RL algorithms (e.g., DDPG [13]) converge slowly and are not data-efficient due to the difficulty to explore non-zero rewards. To address the problem, we may add demonstration data into the replay buffer [14, 15, 16, 17]. However, these algorithms either need a large amount of demonstration data to balance the data distribution, or may still diverge due to the difficulty in exploration. On the other hand, adding priors to current reward function, namely reward shaping (RS) [18, 19, 20, 21] can guide the policy towards the desired behaviors indicated by the priors. By adding RS to IL, the student policy can explore more efficiently due to the guidance provided by the expert, and keep the potential to capture the desired behaviors to outperform the expert. Our proposed policy learning method leverages IL, RL, and RS, which is data-efficient, converges fast, and can outperform the suboptimal expert demonstrator in sparse reward environments.

3 Problem Formulation

This paper considers the delicate industrial insertion tasks. Let ε be a task (characterized by the workpiece) and $\mathcal{E}$ be the distribution of tasks. The goal is to efficiently learn a policy π_ε, which

varies according to the tasks, that achieves the highest reward over the distribution of tasks. Let τ be the trajectory generated by π_ε and $R(\tau)$ be the reward function on τ, which evaluates the completion of the task within a given time frame, the duration of the task, and the number of collisions during the task. Then the policy learning problem can be written as the following optimization:

$$\max_{\pi_\varepsilon} \ \mathbb{E}_{\varepsilon\sim\mathcal{E}}[\ \mathbb{E}_{\tau\sim\pi_\varepsilon}[R(\tau)]\ ]. \tag{1}$$

It is worth noting that every task has a nominal model that is characterized by the design parameters of the workpiece (e.g., CAD model), while the real workpiece may deviates from the design parameters due to defects, which are difficult to be perceived before insertion. We define a mapping from a real workpiece ε to its nominal model $\bar{\varepsilon}$ as $f : \varepsilon \mapsto \bar{\varepsilon}$. It is assumed that before insertion, only the nominal model $\bar{\varepsilon}$ is available, while we can infer the actual model ε based on the insertion performance. In the following discussion, we introduce the task embedding using tolerance.

Tolerance The performance of policy depends on the system dynamics, which then depends on the task. It is natural to use the peculiar dynamic properties of different workpieces to encode various insertion tasks. Here we use tolerance as the fundamental property to construct task encoding. Tolerance $\mathcal{T}(\varepsilon)$ of task ε is defined as the set of states $\bar{s}$ of the workpiece so that the pins (denoted as the set $\mathcal{C}_{p,\varepsilon}(\bar{s})$) can be inserted into the holes (denoted as the set $\mathcal{C}_{h,\varepsilon}$):

$$\mathcal{T}(\varepsilon) = \{\ \bar{s} \mid \mathcal{C}_{p,\varepsilon}(\bar{s}) \subset \mathcal{C}_{h,\varepsilon}\ \}. \tag{2}$$

Figures 2b and 2d show examples for tolerance. Workpieces with different specifications may result in the same tolerance. Tasks with similar tolerances should have similar policies. However, since $\mathcal{T}(\varepsilon)$ is excessively redundant as task encoding, we will introduce an encoder to transfer tolerances to low-dimensional representations $\psi(\varepsilon)$ to be discussed in section 5.1. We then use $\psi(\varepsilon)$ to guide the policy in order to generate diversified strategies for different workpieces. Since the tolerance of defective realistic workpieces is inaccessible before insertion, we can only parameterize the policy using its nominal model $\bar{\varepsilon} = f(\varepsilon)$. Therefore, the policy learning problem in (1) is decomposed into two problems: policy learning with respect to the nominal model formulated in (3) and inference of the actual task ε.

$$\max_{\pi(\psi(\bar{\varepsilon}))} \ \mathbb{E}_{\varepsilon\sim\mathcal{E}}[\ \mathbb{E}_{\tau\sim\pi(\psi(\bar{\varepsilon}))}[R(\tau)]\ ], \tag{3}$$

Assumptions and Notations This paper focuses on the insertion tasks of circle-shaped multiple pin-hole pairs and polygon-shaped single pin-hole pairs. We use n*m to denote that there are n rows and m pins per row. For example, Fig. 1a shows a 1*3 case. The x and y axes span the horizontal plane and the z axis is the vertical axis that aligns with the insertion direction. The origin is attached to the center of the board on its upper surface. It is assumed that pins and holes are rigid during insertion, and the rotations w.r.t. the x or y axis are negligible given that the robot controller is robust along these two axes. We denote $[x, y, \theta, z]$ as the pose of the workpiece, $[F_x, F_y, F_z]$ as the forces on the workpiece along each axis, q_θ as the torque on the workpiece along θ, $s = [x, y, \theta, z, F_x, F_y, q_\theta, F_z]$ as the state of workpiece, and $a = [u_x, u_y, u_\theta, u_z]$ as velocities along each axis and also the action of the robot. $\bar{s} = [x, y, \theta]$ denotes the truncated state used to determine the tolerance and the success of insertions, i.e., if $\bar{s} \in \mathcal{T}(\varepsilon)$ and $z < 0$, then the pins are inside the holes. The target insertion pose is denoted as $\bar{s}^* = [x^*, y^*, \theta^*] \in \mathcal{T}(\varepsilon)$. The coordinate system is defined such that the target insertion poses are $[0, 0, 0]$ for all nominal tasks. For real tasks, the target insertion pose may be set to a non-zero value to maximize the chance of successful insertion with defective workpieces. The measurement $\hat{s} = [\hat{x}, \hat{y}, \hat{\theta}, \hat{z}, \hat{F}_x, \hat{F}_y, \hat{q}_\theta, \hat{F}_z]$ of the state s follows a Gaussian distribution with mean s. Define $\hat{s}^* := [\hat{x} - x^*, \hat{y} - y^*, \hat{\theta} - \theta^*, \hat{z}, \hat{F}_x, \hat{F}_y, \hat{q}_\theta, \hat{F}_z]$ as the shifted measurement, which will be used as the input to the policy. We also define $r(s, a)$ as the reward function for state-action pairs, while the cumulative $r(s, a)$ over a trajectory τ is $R(\tau)$. The discount factor is set to be 1.

System Architecture The proposed policy π_ε consists of two components as shown in Fig. 1b: the policy $\pi(\psi(\bar{\varepsilon}))$ (shown as the policy network) and the probabilistic inference on the actual task ε (shown as the outer loop). The policy $\pi(\psi(\bar{\varepsilon}))$ also contains two parts: the nominal policy (shown in blue) and the tolerance-guided adaptation (shown in green). The nominal policy $\pi_n : \hat{s}^* \to a$ considers only the measurement feedback and does not consider the diversity of the workpieces. It is trained on a representative workpiece ε_0. The nominal policy learns the common features of

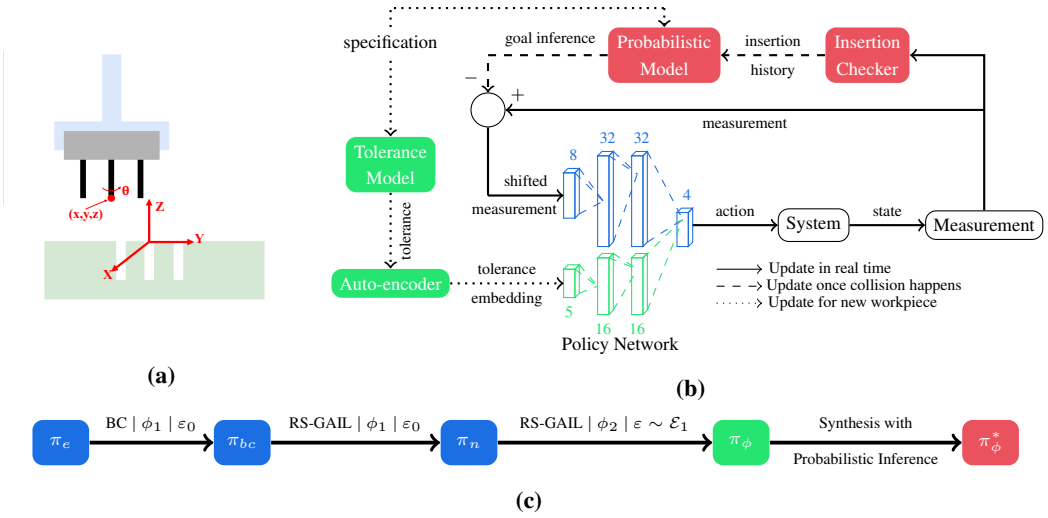


Figure 1: System diagram. (a) Coordination system. (b) System architecture for tolerance-guided insertion. (c) Training pipeline for tolerance-guided policy.

insertion tasks. The tolerance-guided adaptation is designed to handle distinctions among workpieces, serving similarly as learning residual [22, 23]. It takes the task encoding $\psi(\bar{\varepsilon})$ as input and provides modifications to the nominal policy. We call the policy combining the nominal policy and the tolerance-guided adaptation as a diversified policy and parameterize the policy by ϕ, i.e., $\pi_\phi(\psi(\bar{\varepsilon})) : \hat{s}^* \times \psi(\bar{\varepsilon}) \mapsto a$. The parameter contains two parts $\phi = [\phi_1; \phi_2]$ where ϕ_1 corresponds to the nominal policy and ϕ_2 corresponds to the tolerance-guided adaptation. The training of π_ϕ is discussed in section 4 and section 5.1. Probabilistic inference then exploits the insertion history of the current workpiece and infers the actual task ε and the optimal reference point $\bar{s}^*$. The optimal reference for $\bar{s}^*$ will then be compensated to the input vector of π_ϕ. The new policy is denoted π^*_ϕ, to be discussed in section 5.2.

4 Learning

The learning algorithm will be used to train the nominal policy π_n as well as the diversified policy π_ϕ. The learning algorithm essentially solves the following problem

$$\max_{\pi} \mathbb{E}_{\tau\sim\pi,\varepsilon}[R(\tau)]. \tag{4}$$

Since the workpieces have small tolerance in our application, positive rewards (i.e., successful insertion) are sparse. As a result, the exploration of RL algorithm becomes expensive. Imitation learning algorithms provide efficient guidance for exploration, but the performance of the learned policy is constrained by the optimality of the expert policy. We propose an integrated imitation learning and reinforcement learning framework, generative adversarial imitation learning with reward shaping (RS-GAIL), which leverages the effective exploration of IL and optimality of RL. Denote the expert policy as π_e which may be generated by simple PID control or human demonstration. The original GAIL [12] solves the following minimax problem

$$\min_{\pi}\max_{D} \mathbb{E}_{\pi_e}[D(s,a)] - \mathbb{E}_{\pi}[D(s,a)], \tag{5}$$

where D is the discriminator that tries to differentiate the learned policy from the expert policy. s and a are the states and actions generated by the policy. However, since the original GAIL does not contain the environment reward, the learned policy may be suboptimal. We then combine the two objective (4) and (5) through a weighting factor $\alpha \in [0, 1]$:

$$\min_{\pi}\max_{D} (1-\alpha)\left[\mathbb{E}_{\pi_e}[D(s,a)] - \mathbb{E}_{\pi}[D(s,a)]\right] - \alpha\mathbb{E}_{\pi}[r(s,a)]. \tag{6}$$

When $\alpha = 0$, the problem reduces to GAIL, where π goes to π_e. When $\alpha = 1$, the problem is a pure reinforcement learning problem, where π optimizes the environment reward. In practice, we can start with a small α so that the policy π can mimic π_e to avoid useless exploration in the beginning, then gradually increase α in order to achieve optimality. The detailed execution of the RS-GAIL algorithm is summarized in algorithm 1.

Algorithm 1 The Generative Adversarial Imitation Learning with Reward Shaping (RS-GAIL).

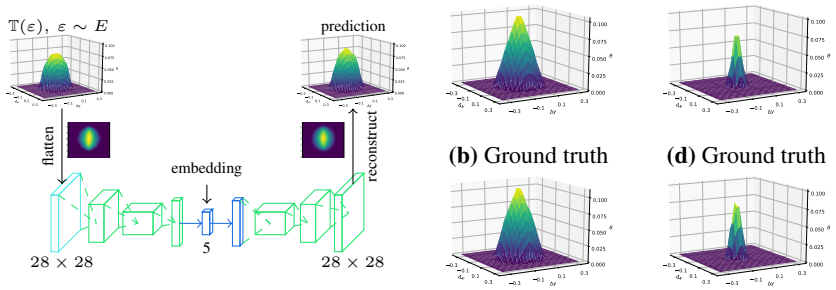
1: **Input:** Expert trajectory $\tau_e \sim \pi_e$, initial policy parameters w_{init}, reward shaping factor α
2: Train $\pi_{w_{init}}$ on τ_e using Behavior Cloning, get $\pi_{w_{BC}}$
3: Initialize the policy parameters in GAIL w_0 with w_{BC}, initialize discriminator parameters d_0
4: **for** $i = 0, 1, \ldots, N-1$ **do**
5: Rollout student trajectory $\tau_i \sim \pi_{\theta_i}$, updated the discriminator parameters from d_i to d_{i+1} using gradient descent with cost function

$$\max_{d_{i+1}} \mathbb{E}_{\pi_e}[D_{d_{i+1}}(s,a)] - \mathbb{E}_{\pi_{w_i}}[D_{d_{i+1}}(s,a)]. \tag{7}$$

6: Update the policy parameter from w_i to w_{i+1} using CMA-ES optimization [24] on

$$\max_{w_{i+1}} (1-\alpha)\mathbb{E}_{\pi_{w_{i+1}}}[D_{d_{i+1}}(s,a)] + \alpha\mathbb{E}_{\pi_{w_{i+1}}}[r(s,a)]. \tag{8}$$

7: **end for**
8: **return** optimal policy π_{θ_N}



(a) Auto-encoder architecture

(b) Ground truth

(c) Prediction

(d) Ground truth

(e) Prediction

Figure 2: Tolerance embedding. (a) Architecture of the tolerance auto-encoder where cyan, green, and blue layers denote input, convolutional, and fully-connected layers. (b) Tolerance of a circle-shaped 2*2 workpiece with 0.3mm radius of pins, 0.5mm radius of holes, and 2.5mm as row/column intervals. (d) Tolerance of a regular-polygon-shaped 1*1 workpiece with 1.5mm circumradius of pins and 1.575mm circumradius of holes. (c) and (e) are the corresponding predictions. The units of x and y axes are millimeter. The unit of θ axis is radius. Only the upper parts of tolerance are shown here since the difference between the upper part ($\theta > 0$) and the lower part ($\theta < 0$) is negligible for small θ. Meanwhile, the upper part is already distinctive enough to distinguish different workpieces.

5 Adaptation

5.1 Tolerance-guided Policy

Convolutional auto-encoder [25, 26] is introduced for task encoding $\psi(\varepsilon)$ to reduce the redundancy of tolerance $\mathcal{T}(\varepsilon)$ for $\varepsilon \sim \mathcal{E}$ as shown in Fig. 2. Though we may directly encode the three-dimensional tolerance [27, 28], it is more efficient to do two-dimensional encoding by 'flattening' the surface of the tolerance, i.e., using a value table to store θ on the tolerance surface and turning the volume of $\mathcal{T}(\varepsilon)$ to an image. In this way, we can use a solid simple-structured convolutional auto-encoders for two-dimensional image processing [29]. Encoders for circle-shaped workpieces and polygon-shaped workpieces are trained separately with corresponding training data. Training data for circle-shaped workpieces is sampled from 2*1 to 2*20 sockets with different radius of pins and radius of holes. Training data for polygon-shaped workpieces is sampled from 1*1 socket with shapes of triangle, rectangle, pentagon, and hexagon in different sizes. As shown in Fig. 2, we can successfully reduce the dimension of tolerance to 5.

The overall training pipeline of the tolerance-guided policy is shown in Fig. 1c. It can be viewed as a form of curriculum learning [30]. We first start with the nominal part of the diversified policy $\pi(\psi(\varepsilon))$ on a representative task ε_0 to learn the general features of insertions. In this phase, only the parameters related to the measurements, i.e., ϕ_1, are updated. According to algorithm 1, we first initialize $\pi_{bc} : \hat{s}^* \mapsto a$ by cloning an expert policy π_e. Then we obtain the nominal policy $\pi_n : \hat{s}^* \mapsto a$ by applying RS-GAIL on top of π_{bc}. In the second phase, we enable policy adaptation

with tolerance embedding to learn the distinction among various tasks. The parameters in ϕ_1 are copied from π_n and do not change during the training. Parameters that are related to tolerance embedding, i.e., ϕ_2, are updated from zero initiation using RS-GAIL. Tasks ε in this phase are sampled from a training task set $\mathcal{E}_1$. The resulting policy is denoted $\pi_\phi : \hat{s}^* \times \psi(\varepsilon) \mapsto a$.

5.2 Probabilistic Inference

Since the tolerance of realistic workpieces $\mathcal{T}(\varepsilon)$ may not align with its nominal form $\mathcal{T}(\bar{\varepsilon})$, the diversified policy $\pi(\psi(\bar{\varepsilon}))$ may deviate from the desired policy $\pi(\psi(\varepsilon))$. Therefore, we introduce probabilistic model to narrow the gap between the two policies. We leverage the information contained in previous insertions to infer the actual task ε and the optimal insertion point $\bar{s}^*$. Denote $I_1, \ldots, I_m$ as the results of previous insertions and $\bar{s}_1, \ldots, \bar{s}_m$ as the corresponding insertion states. m is the number of previous insertions. $I = 1$ indicates successful insertion while $I = 0$ indicates the opposite. Now we consider the problem of maximizing the expectation of I_{m+1} over the next goal insertion state $\bar{s}^*$ given insertion histories: $\max_{\bar{s}^*} \; g_{m+1} = \mathbb{E}[I_{m+1}|I_1, \ldots, I_m, \bar{s}_1, \ldots, \bar{s}_m, \bar{s}^*]$. We assume that the prior distribution of the realistic workpieces ε given its nominal model $\bar{\varepsilon}$ is normal $\varepsilon \sim \mathcal{N}(\bar{\varepsilon}, \Sigma)$. The conditional probability $p(I_j \mid \bar{s}_j, \varepsilon)$ for the jth insertion attempt (denoted as I_j) equals to I_j if $\mathcal{C}_{p,\varepsilon}(\bar{s}_j) \subset \mathcal{C}_{h,\varepsilon}$ or $1 - I_j$ if $\mathcal{C}_{p,\varepsilon}(\bar{s}_j) \not\subset \mathcal{C}_{h,\varepsilon}$ Then we can infer the actual workpiece ε using the previous insertion results and insertion states by applying the Bayesian rule:

$$p(\varepsilon \mid I_1, \ldots, I_m, \bar{s}_1, \ldots, \bar{s}_m) = \frac{\Pi_{j=1}^m p(I_j \mid \bar{s}_j, \varepsilon)\, p(\varepsilon)}{\int \Pi_{j=1}^m p(I_j \mid \bar{s}_j, \varepsilon)\, p(\varepsilon)\, \mathrm{d}\varepsilon}. \tag{9}$$

Then we can rewrite the objective g_{m+1} in (10), which is conditioned on true insertion states. In practice, true states of workpieces are inaccessible. An altered objective g'_{m+1} conditioned on measured states given the noise model $p(\bar{s}|\hat{\bar{s}})$ is shown in (11), where $\hat{\bar{s}}$ are the observed truncated states.

$$g_{m+1} = \int p(I_{m+1} \mid \bar{s}^*, \varepsilon)\, p(\varepsilon \mid I_1, \ldots, I_m, \bar{s}_1, \ldots, \bar{s}_m)\, \mathrm{d}\varepsilon, \tag{10}$$

$$g^{'}_{m+1} = \int g_{m+1}\, p(\bar{s}_1, \ldots, \bar{s}_m \mid \hat{\bar{s}}_1, \ldots, \hat{\bar{s}}_m)\, \mathrm{d}\bar{s}_1 \ldots \mathrm{d}\bar{s}_m. \tag{11}$$

For real applications, both integrals in (10) and (11) are challenging to compute, which require complex slicing to transfer the multi-dimensional integral to practicable repeated integral. Therefore, we solve the problem numerically by sampling ε from its distribution $N(\bar{\varepsilon}, \Sigma)$ and $\bar{s}$ from the measurement noise model. Various optimization methods can be applied to maximize g_{m+1}. Here we adopt the covariance matrix adaptation evolutionary method (CMA-ES) [24]. At each iteration, we sample various $\bar{s}_{m+1}$ and evaluate them with the expectation in (10) or (11). For a consecutive insertion task, we apply probabilistic inference whenever a collision occurs. With the knowledge of the insertion histories, the probabilistic inference utilizes CMA-ES to output the optimal goal insertion state $\bar{s}^*$. A workpiece can be considered as immoderately defected and better to be discarded if the optimal g_{m+1} falls below a certain threshold, which can increase the robustness of industrial insertion production lines.

An numerical study of the probabilistic inference is conducted on workpieces with 2*1 circle-shaped pins and random defects: stochastic horizontal translation of each pin. The workpiece has 0.3mm pin radius, 0.5mm hole radius, and 5mm nominal interval between two pins. This experiment only tests probabilistic inference and assumes perfect control to the goal position and perfect measurement of states. 500 samples are generated for the given distribution of defects where only around 60% of samples can be inserted with unadapted goal states and 0.8% of samples are impossible to insert regardless of goal states. We compare the probabilistic model with a random policy that randomly selects goals from a pre-defined set of goals. The results demonstrate that the probabilistic model outperforms random policy in terms of both the success rate within 10 attempts and the average insertion attempts before success. It increases the former from 60% to 90.2% and maintains an average attempts of 1.836. In contrast, the random policy has a success rate of 78.2% and 2.796 average attempts.

6 Results and Discussions

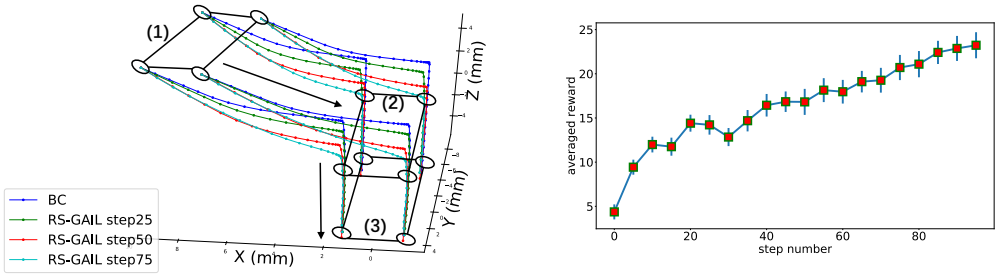
Experiment Setup The proposed method has been tested in OpenAI Gym environment. The dynamics are specified in discrete time with sampling rate $dt = 0.25s$. When there is no col-

lision, the movement follows $s' = s + a \cdot dt$. When there is a collision between two time steps, the robot will stop at the first collision point and receive a contact force (now set as a binary value) at the direction where the collision happens. In addition, the actions are saturated at $[-a_{\max}, a_{\max}]$. At the start of each episode, the workpiece is reset to a random initial pose. The reward function is defined to be $c(1 - \frac{k}{k_{\max}})$ if the insertion is finished before the maximum horizon $k_{\max}$, and -1 if a collision happens. The parameter c adjusts the trade-off between efficiency (fast insertion) and safety (collision minimization). In our experiments, we set $a_{\max} = [2.5\text{mm/s}, 2.5\text{mm/s}, 0.1\text{rad/s}, 2.5\text{mm/s}]$, $k_{\max} = 40$ and $c = 100$. The standard deviation of the measurement noise is set to be $[0.1\text{mm}, 0.1\text{mm}, 0.0\text{rad}, 0.1\text{mm}]$ for $[x, y, \theta, z]$. For RS-GAIL training, the weight α in (6) is set to 0.8 and the maximum iteration is set to 100. The discriminator is updated once every 5 iterations. We stop training when the environment reward of the student policy stops increasing; and the probability that the student state-action pairs are classified by the discriminator as expert is stably around 40-60%.

We designed two expert policies, one emphasizing safety and the other emphasizing efficiency. The safe expert policy π_e^{safe} first moves the workpiece horizontally to align the pins and holes, and then move vertically down to insert the workpiece. The efficient expert policy π_e^{eff} moves in full speed towards the hole. The safe expert has fewer collisions and longer insertion time than the efficient expert. The optimal behavior would be a balance between the two.

Results Effect of different experts. We trained nominal policies from the two experts π_e^{eff} and π_e^{safe}. Comparing experiments C1.1 to C1.3 and C2.1 to C2.3 as shown in table 1, RS-GAIL optimizes the cloned policy π_{bc} by reducing the number of collisions of π_e^{eff} and speeding up the number of steps of π_e^{safe}. Because our RS-GAIL starts by mimicking the expert and then gradually optimizes the policy according to the environment rewards, even starting from experts with different behaviors, the policy can finally converge to similar optimal behavior. As shown in C1.3 and C2.3, we finally get similar π_n with promising number of steps and number of collisions. Effectiveness of RS-GAIL. From Fig. 3 and C2.1 to C2.3, policies trained by behavior cloning π_{bc} having similar behavior with the expert π_e, which first moves in xy plane for pin-hole alignment and then moves down for insertion. This 'L' shaped trajectory is not optimal because longer travel distance induces more steps and less successful rate. As a result, π_e gets low rewards, and π_{bc} even lower. Our RS-GAIL starts from π_{bc} and optimizes it according to our defined reward function. During the training process of RS-GAIL, the learned policy π_n generates smoother trajectories, which have higher rewards and successful rate. We perform the following experiments using the π_n in C2.3. Generalization to new workpieces. Consider C2.3, C3.1, and C3.2. When we introduce new workpiece 2*4 in C3.1, π_n trained in C2.3 is not able to get good performance. However, our tolerance-guilded policy π_ϕ is able to handle this workpiece generalization problem. In C3.2, by training on data of workpiece 2*2 and 2*8 and only updating the parameters of the tolerance embedding networks (ϕ_2), π_ϕ can be generalized to new unseen workpiece 2*4 and get higher performance than π_n. Generalization to defective workpieces. In C4.1 and C4.2, we utilize the same defective 2*1 workpieces in section 5.2 to test synthesized probabilistic inference. Both the reward and the success rate drop significantly due to defects. The policy with probabilistic inference, i.e., π_ϕ^*, mitigates the negative effects of defects and increases both the reward and the success rate. Generalization to different pin geometries. We also create a series of 1*1 workpieces with polygon-shape pins, including rectangle  , pentagon  , and hexagon  . As shown in P1.1 to P1.3, RS-GAIL optimizes the policy similar to C2.1 to C2.3. Regardless of the differences in the shape of pin sections, those workpieces can be handled with similar policies. As a result, our robust nominal policy π_n gets similar performance on  and  , as shown in P1.3 and P2.1. In P2.2, by training on  and  , our π_ϕ slightly outperforms π_n on  .

Discussion The tolerance model can be extended in multiple directions. This paper ignored the rotation outside the xy plane and constructed $\mathcal{T}$ only using the truncated state $[x, y, \theta]$. It is possible to relax the assumption and include more degrees of freedom in the truncated states. Meanwhile, although this paper only considers insertion with spacial tolerances, the proposed method can apply to insertions with tight fit using a tolerance model on force and torque, e.g., $\bar{s} = [x, y, \theta, F_x, F_y, \tau_\theta]$. In addition, there are several limitations of the method and will be addressed in future work. First, the reliability of the task encoding is determined by the prediction error of auto-encoder. The accuracy of prediction drops significantly when the tolerance is big in θ (e.g. over 0.5 rad). We may introduce regularization or normalization of θ to address the problem. Second, to apply the proposed method on real robot hardware, we need to resolve the sim-to-real gap. One method is to use a robust



(a) **(b)**

Figure 3: Illustration of the RS-GAIL training on the nominal policy in experiment C2.3. (a) Trajectory change over the training process demonstrated using a 2*2 workpiece, where (1) is the initial pin position; (2) is the pin position when the pins start entering the holes; and (3) is the pin position of successful insertion. (b) Reward (with variance) change over the training process in experiment C2.3.

ID	Policy	Train env	Eval env	reward	successful rate	number of steps	number of collisions
C1.1	π_e^{eff}	2*2	2*2	13.76 ± 1.20	0.75 ± 0.03	29.65 ± 0.23	6.11 ± 0.71
C1.2	π_{bc}	2*2	2*2	6.57 ± 3.07	0.68 ± 0.04	30.91 ± 0.71	8.15 ± 1.31
C1.3	π_n	2*2	2*2	$\mathbf{20.82 \pm 0.76}$	$\mathbf{0.94 \pm 0.03}$	$\mathbf{31.05 \pm 0.24}$	$\mathbf{1.69 \pm 0.28}$
C2.1	π_e^{safe}	2*2	2*2	11.28 ± 0.72	0.77 ± 0.04	35.49 ± 0.29	0.0 ± 0.0
C2.2	π_{bc}	2*2	2*2	4.94 ± 0.96	0.58 ± 0.05	37.69 ± 0.28	0.83 ± 0.31
C2.3	π_n	2*2	2*2	$\mathbf{20.58 \pm 0.71}$	$\mathbf{0.90 \pm 0.02}$	$\mathbf{31.40 \pm 0.20}$	$\mathbf{0.91 \pm 0.24}$
C3.1	π_n	2*2	2*4	16.38 ± 1.53	0.84 ± 0.03	33.37 ± 0.43	2.74 ± 0.49
C3.2	π_ϕ	2*2, 2*8	2*4	$\mathbf{20.47 \pm 0.54}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{31.42 \pm 0.18}$	$\mathbf{0.98 \pm 0.10}$
C4.1	π_ϕ	2*2, 2*8	2*1 defected	6.04 ± 1.28	0.30 ± 0.03	35.03 ± 0.53	6.42 ± 0.11
C4.2	π_ϕ^*	-	2*1 defected	$\mathbf{12.44 \pm 2.38}$	$\mathbf{0.41 \pm 0.02}$	$\mathbf{32.80 \pm 0.82}$	$\mathbf{5.60 \pm 0.37}$
P1.1	π_e^{safe}	□	□	12.36 ± 0.41	0.82 ± 0.03	35.05 ± 0.16	0.0 ± 0.0
P1.2	π_{bc}	□	□	7.46 ± 0.30	0.68 ± 0.02	37.02 ± 0.12	0.0 ± 0.0
P1.3	π_n	□	□	$\mathbf{15.47 \pm 0.54}$	$\mathbf{0.87 \pm 0.01}$	$\mathbf{33.96 \pm 0.18}$	$\mathbf{2.14 \pm 0.17}$
P2.1	π_n	□	⬠	15.82 ± 0.43	0.87 ± 0.01	32.91 ± 0.16	1.90 ± 0.05
P2.2	π_ϕ	□ ⬡	⬠	$\mathbf{16.01 \pm 0.51}$	$\mathbf{0.89 \pm 0.02}$	$\mathbf{32.84 \pm 0.20}$	$\mathbf{1.88 \pm 0.01}$

Table 1: Evaluation results of different policies. **Naming Rule**: IDs started with "C" and "P" means workpieces with 2*N circle pins and 1*1 polygon pin, respectively. **Experiment Groups**: The policies in C1.x all come from the efficient expert π_e^{eff}, while all other experiments are based on the safe expert π_e^{safe}. The experiment groups C1.x, C2.x and P1.x compare the expert policy and the cloned policy and the policy learned by RS-GAIL in different situations. The experiment groups C3.x and P2.x compare the nominal policy and the tolerance guided policy in different situations. C4.x compares the policy with and without probabilistic inference. In particular, the policy in C3.1 is identical to the policy in C2.3; the policy in C4.1 is identical to the policy in C3.2. **Environment Specification**: Our 2*N circle workpieces have 0.3mm pin radius and 0.5mm hole radius, while the interval between 2 rows is 7.62mm, and the intervals between N columns are 2.54mm each. For our 1*1 polygon environments, the pin circumradious and hole circumradious of □, ⬠, ⬡ are (1.5mm, 1.65mm), (1.5mm, 1.65mm), (1.2mm, 1.26mm), respectively.

controller [31] on the robot hardware so that the closed loop dynamics (robust controller plus the hardware) are almost identical to the dynamics used in the simulation.

7 Conclusion

This paper introduced a tolerance-guided policy learning method to generate adaptable and transferable policies for delicate industrial insertion tasks. We addressed two practical challenges: generalization of the learned policy to unseen tasks and adaptation of the policy when the workpiece has defects. The proposed method contains three parts. First, we used tolerance embedding to encourage the generalization of the learned policy to different tasks. Second, to deal with efficient training in sparse reward environments, we proposed the reward shaping generative adversarial imitation learning algorithm that leverages the advantages of imitation learning and reinforcement learning. Lastly, we developed probabilistic inference methods to infer optimal insertion points based on failed insertions to make the policy robust to defects of the workpiece. The proposed method has been extensively validated through simulation. It has been demonstrated that 1) the proposed learning algorithm efficiently learned optimal policies under sparse rewards; 2) the tolerance embedding enhanced the transferability of the learned policy; 3) the probabilistic inference improved the success rate for defective workpieces.

Acknowledgments

This work is sponsored by Efort Intelligent Equipment Co., Ltd.

References

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

[2] S. R. Chhatpar and M. S. Branicky. Search strategies for peg-in-hole assemblies with position uncertainty. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 3, pages 1465–1470. IEEE, 2001.

[3] R. K. Jain, S. Majumder, and A. Dutta. Scara based peg-in-hole assembly using compliant ipmc micro gripper. *Robotics and Autonomous Systems*, 61(3):297–311, 2013.

[4] H. Qiao, B. Dalay, and R. Parkin. Robotic peg-hole insertion operations using a six-component force sensor. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 207(5):289–306, 1993.

[5] Y. Fan, J. Luo, and M. Tomizuka. A learning framework for high precision industrial assembly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 811–817. IEEE, 2019.

[6] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana. Deep reinforcement learning for high precision assembly tasks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 819–825. IEEE, 2017.

[7] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel. Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3080–3087. IEEE, 2019.

[8] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow. Meta-reinforcement learning for robotic industrial insertion tasks. *arXiv preprint arXiv:2004.14404*, 2020.

[9] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340, 2019.

[10] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[11] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[12] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[14] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[15] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[16] B. Kim, A.-m. Farahmand, J. Pineau, and D. Precup. Learning from limited demonstrations. In *Advances in Neural Information Processing Systems*, pages 2859–2867, 2013.

[17] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

[18] U. Syed and R. E. Schapire. Imitation learning with a value-based prior. *arXiv preprint arXiv:1206.5290*, 2012.

[19] K. Judah, A. Fern, P. Tadepalli, and R. Goetschalckx. Imitation learning with demonstrations and shaping rewards. In *AAAI*, pages 1890–1896. Quebec, 2014.

[20] R. P. Bhattacharyya, D. J. Phillips, C. Liu, J. K. Gupta, K. Driggs-Campbell, and M. J. Kochenderfer. Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 789–795. IEEE, 2019.

[21] Y. Wu, M. Mozifian, and F. Shkurti. Shaping Rewards for Reinforcement Learning with Imperfect Demonstrations using Generative Models. *arXiv e-prints arXiv:2011.01298*, 2020.

[22] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[23] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[24] N. Hansen. The CMA evolution strategy: A tutorial. *arXiv:1604.00772*, 2016.

[25] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[26] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[27] Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai. Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 204:41–50, 2016.

[28] J. Ji, S. Mei, J. Hou, X. Li, and Q. Du. Learning sensor-specific features for hyperspectral images via 3-dimensional convolutional autoencoder. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1820–1823. IEEE, 2017.

[29] Y. Bengio, A. C. Courville, and P. Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1:2012, 2012.

[30] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[31] C. Tang, Z. Xu, and M. Tomizuka. Disturbance-observer-based tracking controller for neural network driving policy transfer. *IEEE Transactions on Intelligent Transportation Systems*, 2019.