AGH University of Krakow

FIELD OF SCIENCE ENGINEERING AND TECHNOLOGY

SCIENTIFIC DISCIPLINE AUTOMATION, ELECTRONIC, ELECTRICAL ENGINEERING AND SPACE TECHNOLOGIES

DOCTORAL THESIS

Deep reinforcement learning for robotic industrial assembly

Author: Grzegorz Bartyzel

First supervisor: dr hab. inż. Paweł Rotter, prof. AGH
Auxiliary supervisor: dr inż. Marcin Węgrzynowski

Completed in: AGH University of Krakow, Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering

Kraków, 2024

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

DZIEDZINA NAUK INŻYNIERYJNO-TECHNICZNYCH

DYSCYPLINA AUTOMATYKA, ELEKTRONIKA, ELEKTROTECHNIKA I TECHNOLOGIE KOSMICZNE

ROZPRAWA DOKTORSKA

Zastosowanie głębokiego uczenia ze wzmocnieniem w robotycznym montażu przemysłowym

Autor: Grzegorz Bartyzel

Promotor rozprawy: dr hab. inż. Paweł Rotter, prof. AGH
Promotor pomocniczy: dr inż. Marcin Węgrzynowski

Praca wykonana: Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Kraków, 2024

Foremost, I would like to express my sincere appreciation to my colleagues from the Chi Team – Maciej Aleksandrowicz, Paweł Irzyk, Mariusz Jagosz, Jakub Motak, Magdalena Połetek, Marcin Rosenhof, Adam Serafin, and Jakub Turaj – for their professionalism, collaboration in developing the robotic platform, and the positive working environment they have fostered. I also wish to extend my gratitude to my senior colleagues, dr inż. Wojeciech Półchłopek and dr inż. Dominik Rzepka, for their invaluable advice and support throughout the project.

I am thankful to my supervisor, dr hab. inż. Paweł Rotter, for his guidance, support, and valuable feedback during the writing of this thesis. Additionally, I would like to acknowledge dr inż. Marcin Węgrzynowski for his assistance and advice in the initial stages of the project.

My appreciation also goes to Jakub Fidelus, the president of Fitech, and the entire Fitech company for funding the research and providing me with the opportunity to work on this exciting project.

Finally, I extend my deepest gratitude to my beloved wife, Justyna, for her unwavering support, patience, and understanding throughout this journey. Without her, this achievement would not have been possible.

I dedicate this dissertation to my deceased father, Zygmunt.

This work was carried out in collaboration with the company Fitech, funded by the Polish National Center for Research and Development through Project “Intelligent robot for autonomous handling of assembly of electronic components based on artificial intelligence and neural networks” Number POIR.01.01.01-00-0123/19.

Abstract

Modern production lines are characterized by a complex industrial process, driven by the demand for increasingly complex products. As a result, there is a growing need for flexibility, which requires these lines to quickly adapt to changing production conditions. This demand is especially evident in assembly processes, where the complexity comes from the need for precise positioning and integration of various components. Although automated stations using industrial robots are becoming more common, they face challenges in quickly adapting to the assembly of different products. Traditional robot control methods, such as following manually programmed trajectories, often fall short in flexible production lines. Furthermore, equipping robotic assembly cells with advanced sensors or tool-changing systems significantly increases the costs of developing and maintaining a production line. As a result, there has been a growing interest in methods based on algorithms using artificial intelligence, particularly reinforcement learning (RL), in recent years.

The aim of the presented dissertation was to develop an RL-based method to effectively assemble various elements on a single production line. Research involved developing an efficient framework to train an RL agent using a real industrial robot and introducing a novel algorithm called *Multimodal Variational DeepMDP* (MVDeepMDP). This algorithm allowed for a quick adaptation of the pre-trained RL agent to new products on production lines by combining the Soft Actor-Critic (SAC) algorithm, which belongs to the family of *off-policy actor-critic algorithms*, with a multimodal with a multimodal variational autoencoder. The proposed method focuses on learning a multimodal latent representation of the dynamics of the environment, allowing the RL agent to receive task-relevant features. Additionally, the framework is characterized by an asynchronous data collection process relative to the learning process, which improves the RL agent's interaction with the environment. Furthermore, the robot arm is controlled using the admittance control system, ensuring a safe and seamless learning process.

The proposed method was evaluated by testing its performance in inserting electronic parts into specific locations on a printed circuit board. This electronic assembly task served as an effective benchmark for assessing the overall effectiveness and generalization capability of the proposed solution, given the diversity and complexity of the components. The initial experiments focused on evaluating the proposed framework using the SAC algorithm. Analysis of results indicated that the presented framework achieved significantly better quality metrics compared to conventional insertion techniques. Afterward, a series of experiments were performed to test the ability of MVDeepMDP to quickly adapt to newly assembled products. The results demonstrated that, in most cases, this method could perform zero-shot transfer to unseen components. In cases where zero-short transfer failed, fine-tuning took a maximum of 5 minutes.

In summary, the presented solution allows for rapid adaptation of the robotic assembly station to constantly changing products in a flexible production environment. This functionality is desirable from the point of view of modern manufacturing standards, which makes it a potential replacement for commonly used solutions.

Streszczenie

Współczesne linie produkcyjne cechują się skomplikowanymi procesami przemysłowymi, które są spowodowane produkcją coraz to bardziej złożonych produktów. Dodatkowo coraz częściej wymaga się od linii produkcyjnych elastyczności, co oznacza, że muszą one być zdolne szybko adaptować się do zmieniających się warunków produkcyjnych. Sytuacja ta jest najbardziej widoczna w przypadku procesów montażu, gdzie złożoność procesu jest związana z koniecznością precyzyjnego pozycjonowania i łączenia różnych elementów. Obecnie coraz częściej stosuje się stanowiska zautomatyzowane, bazujące na robotach przemysłowych, jednak tak wyposażone linie produkcyjne są często niezdolne do szybkiego dostosowania się do montażu często zmieniających się produktów. Dodatkowo tradycyjne metody sterowania robotami (m.in. ręczna definicja trajektorii) często są niewystarczające w przypadku elastycznych linii produkcyjnych. Natomiast uzbrojenie stanowisk montażowych w zaawansowane czujniki, czy też systemy do zmiany narzędzi znacząco podnosi koszty stworzenia i konserwacji linii produkcyjnej. W związku z tym w ostatnich latach coraz większą popularność zdobywają metody oparte na algorytmach wykorzystujących sztuczną inteligencję, a w szczególności uczenie ze wzmocnieniem (ang. *Reinforcement Learning* (RL)).

Celem prezentowanych badań było uzyskanie metody bazującej na uczeniu ze wzmocnieniem, która pozwoli na efektywne rozwiązanie problemu montażu różnorodnych elementów na jednej linii produkcyjnej. W ramach przeprowadzonych badań stworzono wydajną procedurę uczenia agenta RL na rzeczywistym robocie przemysłowym, a także opracowano algorytm *Multimodal Variational DeepMDP* (MVDeepMDP). Algorytm ten pozwolił na szybką adaptację wstępnie wytrenowanego agenta RL do nowych produktów na liniach produkcyjnych. MVDeepMDP łączy algorytm Soft Actor-Critic (SAC), z rodziny algorytmów *off-policy actor-critic*, z multimodalnym wariacyjnym autoenkoderem. Zaproponowana metoda charakteryzuje się uczeniem wielomodalnej reprezentacji dynamiki środowiska, dzięki której agent RL uzyskuje istotne informacje o wykonywanym zadaniu. Dodatkowo wspomniana metodologia charakteryzuje się asynchronicznym procesem zbierania danych w stosunku do procesu uczenia, co znacząco usprawnia interakcję agenta RL ze środowiskiem. Co więcej, ramię robota sterowane jest admitancyjnie, dzięki czemu proces uczenia jest bezpieczny i płynny.

Proponowana metoda została przetestowana na problemie montażu elementów elektronicznych do odpowiednich miejsc na płycie drukowanej. Zadanie montażu elektronicznego stanowiło doskonałe środowisko do oceny ogólnej wydajności proponowanego rozwiązania oraz możliwości jego generalizacji, ze względu na różnorodność i złożoność elementów. Wstępne eksperymenty zakładały ewaluację proponowanej metodologii z wykorzystaniem algorytmu SAC. Analiza uzyskanych wyników sugerowała, że przedstawiona metodologia osiągnęła znacznie wyższe wskaźniki jakości niż powszechne techniki mon-

tażu. W następnej kolejności wykonano serię eksperymentów, których celem było sprawdzenie możliwości szybkiej adaptacji MVDeepMDP do nowych montowanych produktów. Wykazano, że metoda ta w większości przypadków jest w stanie montować nowe elementy bez konieczności dodatkowego trenowania. W pozostałych przypadkach, w których błyskawiczne przeniesienie nie powiodło się, douczenie trwało maksymalnie 5 minut.

Podsumowując, zaprezentowane rozwiązanie pozwala na szybkie dostosowanie się zrobotyzowanego stanowiska montażowego do ciągle zmieniających się produktów w elastycznych środowisku produkcyjnym. Funkcjonalność ta jest pożądana z punktu widzenia współczesnych standardów produkcyjnych, co sprawia, że potencjalnie może zastąpić powszechnie stosowane rozwiązania.

Contents

List of Figures

List of Tables

List of Symbols

Reinforcement Learning and Representation Learning

t	Discrete time $t \in \mathbb{N}$
T	Time horizon of the trajectory $T \in \mathbb{N}$
$\mathbf{x}_t$	State of the environment at time t
$\mathbf{x}_t^d$	Terminal state of the environment
$\mathbf{a}_t$	Action taken by the policy at time t
r_t	Reward received at time t
$\mathbf{o}_t$	Observation received at time t
$\mathcal{X}$	State space
$\mathcal{A}$	Action space
Ω	Observation space
τ	Trajectory of the collected transitions, $\tau = \{(\mathbf{x}_1, \mathbf{a}_1, \mathbf{r}_1), \ldots, (\mathbf{x}_T, \mathbf{a}_T, \mathbf{r}_T)\}$
γ	Discount factor
G_t	Discounted return of the trajectory
$\hat{G}_t$	Reward-to-go of the trajectory
$R(\mathbf{x}_t, \mathbf{a}_t)$	Reward function
$Pr(\mathbf{x}_{t+1}\|\mathbf{x}_t, \mathbf{a}_t)$	Probability transition model in MDP
$O(\mathbf{o}_{t+1}\|\mathbf{x}_t, \mathbf{a}_t)$	Probability observation model in POMDP
$\pi(\mathbf{a}_t\|\mathbf{x}_t)$	Policy function
$Q(\mathbf{x}_t, \mathbf{a}_t)$	Q-function
$V(\mathbf{x}_t)$	Value function
$\boldsymbol{\theta}, \boldsymbol{\phi}, \ldots$	Parameters of function approximators
$\Theta, \Phi, \ldots$	Sets of parameters of function approximators
$\mathcal{D}$	Replay buffer
α	Soft Actor-Critic temperature parameter
$\hat{\mathcal{H}}$	Soft Actor-Critic target entropy
υ	Polyak averaging coefficient
η	Learning rate
$\mathbf{x}$	Neural network input data
$\mathbf{z}$	Latent vector

$\mathbf{e}$	Embedding vector
$\boldsymbol{\mu}$	Mean of the latent distribution
$\boldsymbol{\sigma}$	Standard deviation of the latent distribution
$\boldsymbol{T}$	Precision of the latent distribution, $\boldsymbol{T} = \boldsymbol{\Sigma}^{-1}$
$\mathbf{X}$	Set of neural network input data, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$
$\mathbf{Z}$	Set of latent vectors, $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$
$\mathbf{E}$	Set of embedding vectors, $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_M\}$
β	Beta parameter of the β-VAE
λ	Weights of the generalized Product-of-Experts
$\mathcal{D}_{KL}$	Kullback-Leibler divergence
$\mathcal{N}$	Gaussian distribution
$\mathcal{U}$	Uniform distribution
$\mathcal{J}$	Loss function
$\mathbb{E}$	Expectation
$\mathbb{R}$	Real numbers
$\mathbb{N}$	Natural numbers

Robotics

$\boldsymbol{x}$	Current Cartesian pose of the end-effector
$\dot{\boldsymbol{x}}$	Current Cartesian velocity of the end-effector
$\ddot{\boldsymbol{x}}$	Current Cartesian acceleration of the end-effector
$\boldsymbol{x}_d$	Desired Cartesian pose of the end-effector
$\dot{\boldsymbol{x}}_d$	Desired Cartesian velocity of the end-effector
$\ddot{\boldsymbol{x}}_d$	Desired Cartesian acceleration of the end-effector
$\boldsymbol{x}_r$	Reference Cartesian pose of the end-effector
$\boldsymbol{q}$	Current joint positions
$\boldsymbol{q}_r$	Reference joint positions
$\mathcal{W}_d$	Desired contact wrench
$\mathcal{W}_r$	Reference contact wrench
F_x, F_y, F_z	Force components
τ_x, τ_y, τ_z	Torque components
$\mathcal{W}$	Vector of measurements from the force/torque sensor
$\boldsymbol{\tau}$	Joint torques
$\mathcal{T}$	Homogeneous transformation matrix
$\mathbf{K}_P$	Proportional gain matrix
$\mathbf{K}_I$	Integral gain matrix
$\mathbf{S}$	Selection matrix
$\mathbf{K}$	Stiffness matrix
$\mathbf{D}$	Damping matrix

$\mathbf{M}$	Mass matrix
$\mathbf{J}$	Jacobian matrix of the manipulator

Framework for Industrial Assembly

$\{\mathbf{w}\}, \{\mathbf{r}\}, \{\mathbf{t}\}$	Reference frames of the workspace, robot, and tool
$H \times W$	Image resolution
l^{xy}, l^{ψ}	Perturbation limits for the Cartesian pose and orientation
$\mathcal{P}_{grasp}$	Set of grasp poses of the parts
$\mathcal{P}_{target}$	Set of target poses of the parts
$\boldsymbol{p}_{grasp}$	Grasp pose of the part
$\boldsymbol{p}_{target}$	Target pose of the part
R_{limit}	Radius of the cylindrical workspace
ψ_{limit}	Rotation limit of the workspace

List of Abbreviations

AE	Autoencoder
API	Application Programming Interface
ATC	Augmented Temporal Contrast
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CVAE	Conditional Variational Autoencoder
DDPG	Deep Deterministic Policy Gradient
DDS	Data Distribution Service
DoF	Degrees of Freedom
DP	Dynamic Programming
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
ELBO	Evidence Lower Bound Objective
F/T	Force/Torque
FMS	Flexible Manufacturing System
gPoE	generalized Product-of-Experts
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HMLV	High-Mix Low-Volume
IK	Inverse Kinematics
JMVAE	Joint Multimodal Variational Autoencoder
JS	Jensen-Shannon
KL	Kullback-Leibler
LfD	Learning from Demonstration
LMHV	Low-Mix High-Volume
LR	Learning Rate
LSTM	Long Short-Term Memory
MC	Monte Carlo
MDP	Markov Decision Process
MIB	Multiview Information Bottleneck
MMVAE	Mixtures of Multimodal Variational Autoencoders

MoE	Mixture-of-Experts
MSE	Mean Squared Error
MVAE	Multimodal Variational Autoencoder
NLL	Negative Log-Likelihood
ODE	Ordinary Differential Equation
PCB	Printed Circuit Board
PCBA	Printed Circuit Board Assembly
PD	Proportional-Derivative
PER	Prioritized Experience Replay
PI	Proportional-Integral
PoE	Product-of-Experts
POMDP	Partially Observable Markov Decision Process
RAE	Regularized Autoencoder
RL	Reinforcement Learning
ROS 2	Robot Operating System 2
RSSM	Recurrent State-Space Model
SAC	Soft Actor-Critic
Sim2Real	Simulation to Reality
SLAC	Stochastic Latent Actor-Critic
SVGB	Stochastic Variational Gradient Boosting
TCP	Tool Center Point
TD	Temporal Difference
TD3	Twin Delayed Deep Deterministic Policy Gradient
t-SNE	t-Distributed Stochastic Neighbor Embedding
VAE	Variational Autoencoder
VIB	Variational Information Bottleneck
VS	Visual Servoing
XML-RPC	Extensible Markup Language Remote Procedure Call

Chapter 1

Introduction

1.1 Background

In modern manufacturing systems, industrial robots serve as integral components for automating repetitive and labor-intensive tasks. These systems are designed to perform a wide range of operations, including pick-and-place, welding, and assembly. The majority of existing industrial robotic systems are designed for low-mix, high-volume (LMHV) production lines that are characterized by highly specialized and linear production systems. Such systems are commonly employed in industries such as automotive manufacturing, where dedicated production lines are configured for specific vehicles and operate for extended periods.

However, the emergence of high-mix, low-volume (HMLV) production lines has highlighted the increasing importance of flexible manufacturing systems (FMS). Flexible manufacturing systems characterize production lines capable of quickly adapting to changing production requirements, such as new products or variations in existing products. These systems are prevalent in industries such as electronics manufacturing, where a multitude of devices with lower production volumes, such as specialized devices, are produced. This type of electronic manufacturing epitomizes a high-mix, low-volume production line, which requires a flexible production process to adapt to frequent changes in product design and production volume.

In such systems, the ability to rapidly re-tool automated machines, such as robotic cells, is critical to minimize production downtime. Nevertheless, achieving this level of flexibility in industrial robotic systems is challenging due to the complexity of the tasks involved and the need for precise control of the robot's motion and interaction with the environment. In most cases, tasks requiring high dexterity and precision are still performed manually by human operators, such as the assembly of non-standardized electronic components or the attachment of engine components in the automotive industry.

Although human operators demonstrate high precision and adaptability in performing these tasks, they are also prone to fatigue and errors, leading to lower productivity and quality issues. Consequently, there is a growing interest in developing robotic systems capable of performing these tasks with the same level of precision and adaptability as human operators, while also being able to operate continuously without fatigue or errors.

The field of robotics has made significant advances in recent years, especially in the context of contact-rich manipulation. Contact-rich manipulation refers to tasks that involve physical interaction between robots

and their environment, such as grasping, manipulating, and assembling objects. Achieving effective contact-rich manipulation is challenging, as it necessitates the robot's understanding of the physical characteristics of the environment, including factors such as friction and contact forces.

Conventional approaches to contact-rich manipulation rely on techniques such as hybrid force/motion and impedance control law to control the interaction between the robot end-effector and the surrounding objects. Force-controlled robots execute predefined motions while maintaining a desired force in the given axis or impedance profile. Although these methods have demonstrated success in various applications, they require manual tuning of control parameters, impeding their quick adaptation to new tasks or manipulated objects.

One common strategy to extend the capabilities of force-controlled robots is to integrate tool change systems that enable the robot to switch between different end-effectors to perform various tasks. However, this approach presents drawbacks, such as the need for additional hardware and the need to stop the production cycle to change the end-effector. Moreover, tool change systems may not handle all types of objects present in the production line within a single robotic cell. Therefore, while effective, these systems are costly and require significant space to accommodate all the necessary tools.

An alternative method to improve the flexibility of industrial robotic systems involves utilizing computer vision systems to provide visual feedback to the robot about its environment. Visual feedback can be used to adjust the robot's motion based on observed objects and task requirements. However, these systems often require complex preprocessing and stable lighting conditions to operate effectively. Achieving consistent performance and reusability across different manipulated objects is time-consuming and expensive, leading to configurations tailored for specific sets of objects. Furthermore, for high-precision tasks, the cost of the vision system is significant, as it often necessitates high-resolution cameras and complex image processing algorithms.

Recent advances in deep reinforcement learning (DRL) have introduced new possibilities for developing robotic systems capable of performing high-precision manipulation tasks with adaptability. Reinforcement learning (RL) is a machine learning paradigm in which an agent learns to solve a given task by interacting with an environment and receiving feedback as a reward value that indicates the quality of the made decision. Integrating RL algorithms with robotic systems can provide solutions to tasks that are difficult to solve using traditional vision-based control systems. Moreover, as DRL algorithms rely on deep neural networks to approximate the policy function, they can reduce the need for complex preprocessing and manual tuning of control parameters, thereby reducing system costs and potentially improving reusability across different tasks and manipulated objects. However, it is important to note that "raw" RL algorithms struggle to quickly adapt to new tasks or manipulated objects. Therefore, training a policy from scratch for each new task is time-consuming and impractical in real-world scenarios. Rapid adaptation to new tasks is crucial in the manufacturing industry; hence, this remains an open research problem.

1.2 Industrial Robotic Assembly Tasks

Automation of insertion tasks by industrial robots is a common practice in modern manufacturing systems, where robots play a crucial role in the assembly of objects by inserting one component into another. One of

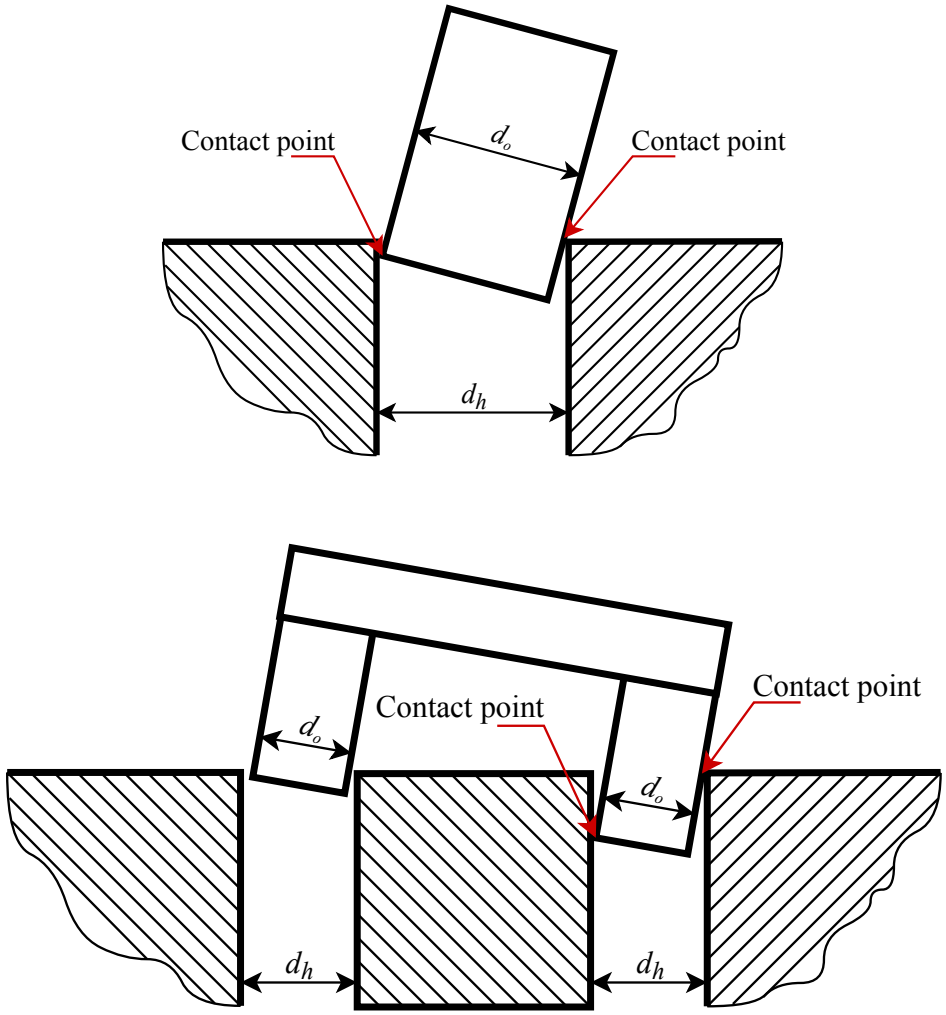


Figure 1.1: Schemas of an industrial robotic assembly task. **Top:** Single peg-in-hole insertion task. **Bottom:** Multiple peg-in-hole insertion task. The object diameter is defined as d_o, while the hole diameter is defined as d_h. The multiple peg-in-hole task is more challenging due to the complex geometry of the object and the need to insert multiple pegs into corresponding holes. In the visualized example, one of the pegs makes contact with the edge of the hole, while the other pegs are still in the process of insertion. Therefore, the robot must adjust its motion to ensure that all pegs are inserted correctly.

the fundamental tasks in this domain is the peg-in-hole insertion, where the robot is tasked with inserting a peg into a corresponding hole, as illustrated in the top schema of Figure 1.1. The task is considered successfully completed when the peg is fully inserted into the hole.

Although the peg-in-hole task may seem relatively straightforward for a human operator, it presents significant challenges when performed by a robot. Complexities arise from various factors, including the complex geometry of objects and the clearance of the hole, both of which have a substantial impact on the control system. Achieving submillimeter precision is often necessary because of the small clearance between the peg and the hole. Additionally, the robot must adapt its motion to accommodate the friction between the peg and the hole, as well as the contact forces that emerge during the insertion process.

The complexity of the insertion task is further intensified when multiple pegs need to be inserted into their corresponding holes, as shown in the lower schema of Figure 1.1. For instance, the assembly of electronic components, such as non-standardized transformers using through-hole technology (THT), involves the insertion of multiple leads into the respective holes on the printed circuit board (PCB). These leads are typically thin and fragile to external forces, making the insertion process difficult due to the risk of damage. Moreover, the tightness of the PCB holes further complicates the insertion process, particularly when coupled with fragile leads.

Let's now explore the general details of the peg-in-hole insertion task. The objective of this task is to insert a peg into a hole, as illustrated in Figure 1.1. The peg and hole are characterized by their respective diameters, d_o and d_h. The insertion process begins with the peg being grasped and moved towards the hole. Subsequently, the robot is required to adjust its motion to align the peg with the hole and execute the insertion. In most cases, aligning the peg with the hole and performing the straight movement on the given axis is sufficient. However, in certain scenarios, the peg may come into contact with the edge of the hole during insertion. In such cases, the robot needs to adjust its tool orientation to ensure proper alignment of the peg with the hole. Furthermore, in high-precision insertion tasks, the robot must not only ensure correct alignment, but also compensate for the contact forces. The task ends when the peg is fully inserted into the designated hole. The same procedure applies to the multiple-peg-in-hole insertion task, where the robot is tasked with inserting multiple pegs into their corresponding holes.

1.3 Research Objectives

The main objective of this dissertation is to demonstrate the feasibility of using deep reinforcement learning in a real-world industrial robotic assembly task, such as electronic component assembly. It is hypothesized that deep reinforcement learning algorithms combined with multimodal representation learning can be used effectively in flexible manufacturing systems to train robotic systems that can quickly adapt to new tasks and manipulated objects. Furthermore, it is expected that the multimodal information fusion mechanism will have an impact on extracting task-relevant features, which, in turn, will improve the generalization capability of the learned policy.

1.4 Thesis Content

The following dissertation is organized as follows. In Section 1.5, a comprehensive review of the literature related to intelligent robotic systems and representation learning is presented. The review begins with an overview of contact-rich manipulation in the field of robotic manipulation tasks (Section 1.5.1), followed by a discussion of methods for multiple peg-in-hole insertion tasks (Section 1.5.2). Subsequently, the review delves into representation learning in reinforcement learning (Section 1.5.3), focusing on methods aimed at improving sample efficiency and policy generalization. Finally, the review concludes with a discussion of multimodal generative learning (Section 1.5.4), presenting methods designed to learn the joint latent representation of multiple modalities. The introduction of the dissertation is concluded with a summary of contributions (Section 1.6), providing details on the research objectives (Section 1.3).

In the following chapter, Chapter 2 provides essential background knowledge for understanding the subsequent chapters. The chapter begins with a comprehensive overview of reinforcement learning (Section 2.1), covering key concepts such as the Markov Decision Process (MDP) [8], as well as value-based and policy-based methods. This section also explores deep reinforcement learning, with a focus on the Soft Actor-Critic (SAC) [31], the core algorithm in this dissertation. The chapter then delves into a discussion of compliance control (Section 2.2) for industrial robots, a critical component in applications for contact-rich manipulation tasks. This section briefly discusses popular compliant control methods, including hybrid position/force control and admittance control. Finally, the chapter concludes with a brief introduction to Variational Autoencoders (VAE) (Section 2.3). The theoretical foundation of VAE is utilized in subsequent chapters as an approach to learning the joint latent representation of multiple modalities.

Chapter 3 introduces the proposed framework for robotic industrial assembly tasks. The chapter begins with a detailed description of the task within the context of the Markov Decision Process in Section 3.1. Then, the control architecture designed for the robotic system is introduced in Section 3.2. Additionally, this section presents the communication software stack for efficient training of RL agents in a real robotic system. The chapter continues with a description of the laboratory setup in Section 3.3, covering the hardware components used in the robotic system, the objects used in the experiments, and the reset procedure for the environment. Furthermore, the implementation of the model architecture used in the experiments is presented in Section 3.4. The chapter concludes with a comprehensive analysis of the experimental results and a discussion of the results obtained in Section 3.5. Initially, a detailed analysis of the influence of visual observation on policy performance is provided. Subsequently, a performance comparison between RL agents and conventional control methods is presented. Finally, qualitative results are discussed at the end of this chapter.

In Chapter 4, the dissertation presents key contributions. The chapter begins with a detailed description of the novel method, called *Multimodal Variational DeepMDP* (MVDeepMDP), which was developed to improve the generalizability of the learned policy deployed in a real-world industrial robotic system. This section provides information about the multimodal fusion mechanism (see Section 4.4), the model architecture (refer to Section 4.1.3), and algorithmic details (refer to Section 4.1.4). Then, the chapter presents the experimental results. The first part of the results section (Section 4.3) covers the quantitative analysis of the learned policy, comparing the transferability of MVDeepMDP to other state-of-the-art methods through a set of benchmark test scenarios. The second part focuses on the qualitative analysis of the proposed approach. The author discusses the impact of the multimodal fusion mechanism on policy performance and analyzes design choices. Finally, the chapter concludes with a discussion of the method for preprocessing state-based observations of different physical quantities.

1.5 Related Work

1.5.1 Contact-rich Manipulation

The field of contact-rich manipulation in robotics is dedicated to studying the interaction between robots and their environment. This field focuses on tasks that involve physical interaction, such as grasping, ma-

nipulating, and assembling objects. Achieving effective contact-rich manipulation poses a challenge, as it requires the robot to comprehend the physical characteristics of the environment, such as friction, compliance, and contact forces. Due to its relevance in practical applications such as manufacturing, contact-rich manipulation has garnered substantial attention within the robotics community over the years [56].

Traditional approaches to contact-rich manipulation rely on force control techniques, such as impedance control and admittance control, to manage the interaction between industrial robots and their surroundings [122, 41, 94]. In these approaches, the robot executes predefined motions while maintaining a desired force or impedance profile. Although these methods have proven successful in various applications, such as industrial assembly [22, 86] and grinding [48], they require manual adjustment of control parameters and are not suitable for tasks involving intricate environmental interactions. Furthermore, force-control-based solutions are vulnerable to environmental uncertainties and lack adaptability to new tasks or manipulated objects.

To overcome these limitations, researchers have delved into the realm of learning-based methods and explored the integration of richer sensory data, such as vision [45] and tactile sensing [65, 119], to facilitate robots in performing contact-rich manipulation tasks. Notable advancements have been achieved through the application of deep learning methods in this domain. For example, the Dex-Net project [77, 75, 76] has successfully employed deep learning in object grasping by leveraging a comprehensive dataset of 3D object models and associated grasps to train a deep neural network to predict the quality of a grasp for a given object. Another area of research worth mentioning where deep learning has been extensively applied is in the realm of robotic industrial insertion. Yu et al. [132] and Triyonoputro et al. [114] have introduced deep learning-based visual servoing (VS), which computes the robot's motion based on visual input processed by a convolutional neural network (CNN). Furthermore, a multimodal approach, as proposed in works by Spector et al. [104, 105], has combined visual and wrench components to compute the desired robot motion, exhibiting greater resilience to environmental uncertainties and adaptability to newly manipulated objects compared to a visual-only approach.

Reinforcement learning has emerged as a promising solution to address the problem of contact-rich manipulation, as evidenced by the recent literature [140]. A prevalent approach involves training a neural network policy using proprioceptive and haptic feedback data to control industrial robots [46, 49]. These policies are often combined with compliance control, allowing robots to interact safely with their environment. While typical RL agents compute the desired robot motion, recent studies [71, 79, 9, 1] have demonstrated that allowing RL agents to adjust the control system parameters online can improve the performance of the learned policy.

However, the reliability of learned policies, based on proprioceptive and haptic feedback data, is frequently compromised by environmental uncertainties, making efficient training and deployment in the real world challenging. Therefore, researchers have investigated the use of visual [64, 118, 95, 68] and multimodal [62, 72] observations for policy training. In these approaches, visual feedback is predominantly acquired from vision sensors attached to the robot's end-effector, whereas multimodal feedback comprises visual, wrench components, and proprioception. These techniques have shown promise in terms of robustness to environmental uncertainties and adaptability to new manipulated objects.

However, the aforementioned approaches typically involve training an RL agent from scratch for each new task or manipulated object, which presents limitations in real-world scenarios due to the time-consuming nature of the training process. To address this issue, common approaches include integrating human demonstrations [100, 120, 70] and performing transfer learning from simulated environments to the real world (sim2real) [4, 133, 136]. In learning from human demonstrations (LfD), expert demonstrations are utilized to pre-train the policy, which is then fine-tuned using reinforcement learning or jointly sampled with online collected data. Another set of techniques that facilitate the transferability of learned policies between tasks involves context-based meta-learning [96, 137, 124] and weakly supervised learning, such as image segmentation of the observed environment [126, 47].

1.5.2 Multiple peg-in-hole Insertion

In the previous section, an overview of contact-rich manipulation was provided within the field of robotic manipulation tasks, along with a discussion of recent advancements in reinforcement learning-based robotic solutions. This dissertation focuses on a case study of flexible manufacturing systems, specifically an electronic component assembly task. The insertion of an electronic part, as a multiple-peg-in-hole task, poses challenges for industrial robots due to the complex geometry of the object [92, 22]. To address this challenge, Hou et al. [44] have proposed an RL-based method utilizing a Deep Deterministic Policy Gradient (DDPG) algorithm [67] supported by a fuzzy logic system and variable timescale prediction. In this approach, the RL agent computes a 6-dimensional action representing translations and rotations along the XYZ-axes based on the object's pose relative to the target and wrench components. Similar approaches have been introduced by Hou et al. [43] and Xu et al. [127], where DDPG is the core algorithm used, and the policy network output is employed to correct the control signal computed by the manually tuned proportional-derived (PD) force controller. However, these works differ in the reward function they use. Notably, Xu et al. [127] propose a fuzzy reward system in place of a complex hand-crafted reward function. These advances aim to expedite training and ensure safe exploration. It is important to note that the manipulated objects in these works were solid metal blocks, which are more resistant to damage than electronic components. Moreover, these objects were rigidly attached to the robot's end-effector, simplifying the problem by reducing uncertainties from the grasping procedure.

In contrast, Ma et al. [73] propose a reinforcement learning-based solution for the assembly of electronic components, such as the pin header. Their approach incorporates high-quality cameras and a precise 6D force/torque (F/T) sensor. However, the agent's observation space consists solely of wrench components, and cameras are used for the pre-policy control step. In contrast to the aforementioned works, the action space in their approach computes only translations along the XYZ-axes.

1.5.3 Representation Learning in Reinforcement Learning

Representation learning, a subfield of machine learning, is concerned with acquiring a compact and informative representation of input data. There are three primary approaches: variational information bottleneck (VIB) [2, 21], contrastive learning [116, 38, 14], and reconstruction-based methods [55, 18, 26]. These

approaches have been effectively utilized in various machine learning tasks, prompting recent investigations into their application in reinforcement learning to enhance sample efficiency and policy generalization.

Early studies involved training autoencoders (AEs) to learn representations for reconstructing observations, which were subsequently used to train RL agents [57, 58]. However, this approach does not ensure that the learned representations encapsulate the information necessary for effective policy computation. Furthermore, training a policy using raw visual input for continuous action spaces often exhibits poor performance and sample inefficiency compared to state-based observations.

Yarats et al. [131] proposed an approach called SAC-AE, which involves joint training of the policy and a deterministic regularized autoencoder (RAE) [25] to learn representations relevant for policy computation. The RAE's encoded representation serves as input for actor and critic networks. Moreover, the authors observed training instability when passing gradients from the actor to the encoder and subsequently proposed stopping this gradient flow. However, a model solely trained for observation reconstruction may not capture relevant representations as effectively as a model learning environmental dynamics.

Consequently, researchers have explored the use of latent dynamics models to learn more informative representations. Ha and Schmidhuber [29] proposed World Models, a method that generates sequential latent representations of the observable environment. This approach involves a two-stage training procedure, first training a VAE with a standard evidence lower bound objective (ELBO) [50], followed by training a dynamic recurrent model based on the VAE's latent representations.

In addition, PlaNet [33] and Dreamer [32] are algorithms that jointly learn a continuous latent dynamics model and observation encoder, with the resulting latent representation used for training the RL agent. These models predict future latent states and rewards while reconstructing observations, enabling long-horizon planning through the use of the Recurrent State-Space Model (RSSM). Subsequent improvements to the Dreamer algorithm [34, 35] significantly improved sample efficiency and policy generalization.

Furthermore, Lee et al. [61] introduced the Stochastic Latent Actor-Critic (SLAC) algorithm, which learns latent beliefs through a latent variable model incorporating model-based rollouts for the computation of belief representation. The aforementioned methods integrate latent dynamic models with off-policy model-free RL algorithms. Meanwhile, an approach suitable for an on-policy model-free RL algorithm has been proposed by Gregor et al. [27]. This method computes a latent belief representation as does SLAC.

Reconstruction-based representation learning has been shown to significantly improve the sample efficiency of the RL agent. However, these approaches are often sensitive to irrelevant changes in the observed environment, particularly when dealing with visual input [135]. As a result, there has been a recent exploration of integrating non-reconstruction-based methods with RL agents. One notable approach is CURL, introduced by Laskin et al. [60], which integrates contrastive learning with an off-policy model-free RL algorithm. The method proposes a contrastive objective tailored for the reinforcement learning domain, computed between randomly cropped images. Another method, Augmented Temporal Contrast (ATC), presented by Stooke et al. [106], also employs contrastive learning. ATC trains a convolutional encoder to associate pairs of observations separated by short time intervals under image augmentation using contrastive loss. In this case, the input images are augmented by random shifting [17].

In addition to the exploration of contrastive learning in the realm of reinforcement learning, there are studies directly focusing on learning task-irrelevant representation. A notable work by Zhang et al. [3] introduces a method called DBC. DBC uses a bisimulation metric to learn task-relevant features. Experimental results have shown that employing the bisimulation metric substantially enhances the generalization capability of the RL agent. Furthermore, a study by Fu et al. [23] presents an improvement that improves the extraction of task information for the Dreamer algorithm. The extraction of task-informed latent representation is achieved through the introduction of two models: the task model and the distractor model. The task model captures task-relevant features associated with reward prediction, while the distractor model captures task-irrelevant features through adversarial objectives. Furthermore, Fan and Lee [19] have developed a robust method called DRIBO, which is based on RSSM and employs a multiview-information bottleneck (MIB). In this work, the MIB objective is adapted to the sequential model. Another noteworthy approach involves prototypical representation [129]. The self-supervised framework, called Proto-RL, links representation learning with exploration through prototypical representations. These prototypes serve as task-agnostic representations that are pre-trained solely with exploratory data, without any task-informative signals.

The above-mentioned methods involve incorporating auxiliary objectives, often accompanied by additional reconstruction models, into RL agents. At the same time, research on image augmentation has shown promising results. Yarats et al. [17, 130] presented the foundations for this area with their method called DrQ, demonstrating that applying simple random shift image augmentation during training significantly improves the sample-efficiency of the RL agent. Similarly, Laskin et al. [59] introduced a method named RAD, which employs random crop image augmentation during training. In addition to DrQ, the SVEA [84] and SODA [37] algorithms have been proposed, showing that involving hard augmentation techniques such as random convolution and random overlay during training significantly improves the generalizability of the RL agent.

1.5.4 Multimodal Generative Learning

In the context of multimodal generative modeling, previous approaches have focused primarily on cross-modal generation. When provided with data from two domains x_1 and x_2, these approaches aim to learn the conditional generative model $p(x_1|x_2)$. It should be noted that the conditioning modality x_2 and the generation modality x_1 are typically not interchangeable. One of the most prevalent methodologies for cross-modal generation involves conditional variational autoencoders (CVAEs) [54, 102, 128, 85]. CVAEs leverage the input modality to condition the latent representation, allowing for the generation of samples from the target modality. However, these techniques are generally limited to handling only two modalities and are not adept at scaling to multiple modalities.

Recent research efforts have delved into approaches that compute the joint latent representation of multiple modalities. Among these, the Joint Multimodal Variational Autoencoder (JMVAE) [112] and the Multimodal Variational Autoencoder (MVAE) [125] have gained prominence. The JMVAE seeks to explicitly learn a joint distribution but does so by training separate inference networks for each potential subset of present modalities, which becomes computationally infeasible as the number of modalities increases. On

the other hand, the MVAE approximates the joint latent representation using the *Product-of-Experts* (PoE), offering greater scalability and the ability to handle missing modalities while processing multiple modalities.

Another notable approach, introduced by Shi et al. [99], is the Mixture of Multimodal Variational Autoencoders (MMVAE), which employs a *Mixture-of-Experts* (MoE) to estimate the joint variational posterior based on individual unimodal posteriors. Sutter et al. have proposed two enhancements to multimodal variational autoencoders, namely mmJSD [107] and MoPoE [108]. The mmJSD is a method for training multimodal VAEs that takes advantage of the Jensen-Shannon (JS) divergence instead of the Kulback-Leibler (KL) divergence to quantify the disparity between the true and approximate posteriors. Meanwhile, MoPoE amalgamates the advantages of MVAE and MMVAE by computing the joint posterior for all subsets of modalities. Nevertheless, it is important to note that the aforementioned methodologies primarily assess their robustness in terms of their capacity to handle missing modalities during inference, while not explicitly addressing their limitations against perturbations in the input data.

1.6 Contributions

According to the information presented in Section 1.3, this dissertation aims to propose an innovative solution for robotic industrial assembly tasks. The primary focus of the experimental results in this work is on the electronic component assembly task, as it continues to present challenges for industrial robots in real-world applications. Additionally, the complexity of this task makes it an excellent benchmark for evaluating the RL agent's generalization capability. The main contributions of this dissertation are outlined as follows:

- Development of an efficient framework for training RL agents on actual robotic systems, with a specific focus on industrial robotic assembly tasks. This framework features a multimodal observation space that contains state-based and visual observations, as well as an admittance control system to ensure safe interaction with the environment. Additionally, a communication software stack has been integrated to facilitate efficient data exchange between the RL agent and the *Robot Operating System 2* (ROS 2) middleware [74].

- Introduction of a novel approach called *Multimodal Variational DeepMDP*, designed to improve the transferability of RL agents to new objects on production lines. This method integrates a multimodal dynamic latent representation learning mechanism with the model-free off-policy RL algorithm, Soft Actor-Critic [31]. In this proposed approach, multimodal information is combined using the *generalized Product-of-Experts* (gPoE) mechanism [12].

- A comprehensive analysis of the impact of the multimodal fusion mechanism on policy performance and generalization capability. The experimental results demonstrate that the proposed method significantly improves the transferability of the RL agent to new objects and can extract task-relevant features from multimodal observations.

- A detailed analysis of the method for preprocessing state-based observations of different physical quantities. It is shown that the independent processing of state-based observations significantly improves the overall performance of the RL agent.

In the upcoming chapters, the dissertation will demonstrate the aforementioned contributions by presenting a series of experiments conducted on an actual robotic system. The experimental findings will provide insights into the effectiveness of the proposed method and its influence on the generalizability of the RL agent. The dissertation will be summarized with a discussion of the results obtained and potential avenues for future research.

Chapter 2

Background

2.1 Reinforcement learning

In this work, the robotic industrial insertion problem is modeled as a standard reinforcement learning [91] framework. The interaction between the agent and the environment is formulated as a Markov Decision Process [8], which is a discrete-time stochastic control decision-making process. The MDP is represented by a tuple $(\mathcal{X}, \mathcal{A}, Pr, R)$, where $\mathcal{X}$ denotes the state space and $\mathcal{A}$ represents the action space. The function Pr represents the probability of state transition, denoted as $Pr\left(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t\right)$, which controls the agent's transition from state $\mathbf{x}_t$ to state $\mathbf{x}_{t+1}$ after taking action $\mathbf{a}_t$. The function R defines the reward function $R\left(\mathbf{x}_t, \mathbf{a}_t\right)$, which evaluates the quality of the agent's decision.

In the RL framework, which is illustrated in Figure 2.1, at each discrete time step t, the agent is in state $\mathbf{x}_t \in \mathcal{X}$, computes and executes the action $\mathbf{a}_t \in \mathcal{A}$. Following the execution of the action $\mathbf{a}_t$, the RL agent transitions to the next state $\mathbf{x}_{t+1}$, which is randomly selected from an unknown state transition distribution $\mathbf{x}_{t+1} \sim Pr(\cdot \mid \mathbf{x}_t, \mathbf{a}_t)$, and receives a scalar reward value r_t that evaluates the quality of the decision made, calculated by the reward function $r_t = R(\mathbf{x}_t, \mathbf{a}_t)$. The agent interacts with the environment until the terminal state $\mathbf{x}_t^d$ is reached, ending the episode, or until a predefined time horizon T is reached. An episode with a predefined time horizon is referred to as an episodic task, while an episode with an infinite time horizon is referred to as a continuing task. The state $\mathbf{x}_t$, the action $\mathbf{a}_t$, and the reward r_t collected during the interaction are collectively referred to as the trajectory τ of length T:

$$\tau = \left\{\left(\mathbf{x}_0, \mathbf{a}_0, r_0\right), \left(\mathbf{x}_1, \mathbf{a}_1, r_1\right), \ldots, \left(\mathbf{x}_T, \mathbf{a}_T, r_T\right)\right\} \tag{2.1}$$

The main objective of the RL agent is to learn a policy $\pi(\mathbf{a}_t \mid \mathbf{x}_t)$ that maps the state $\mathbf{x}_t$ to the action $\mathbf{a}_t$ in order to maximize the expected return G. The return G is calculated as the sum of discounted rewards over the time horizon T, expressed as:

$$G = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^{T-1} r_{T-1} + \gamma^T r_T = \sum_{t=0}^{T} \gamma^t r_t \tag{2.2}$$

Therefore, the objective to be optimized is the following:

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{2.3}$$

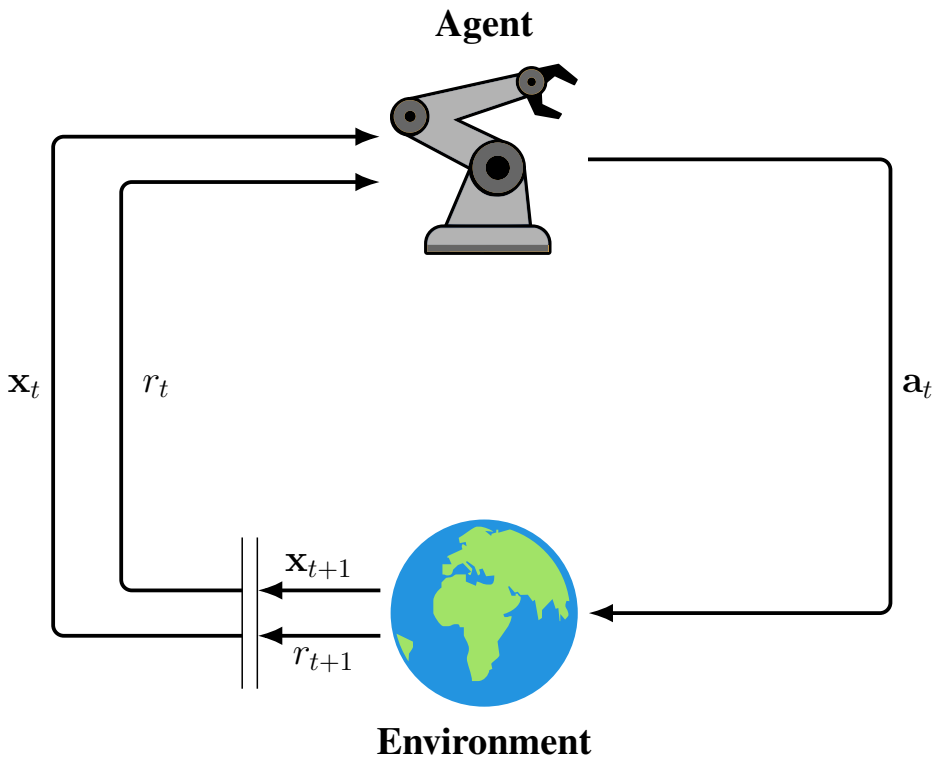


Figure 2.1: Reinforcement learning block diagram. The agent computes action $\mathbf{a}_t$ based on the observed state $\mathbf{x}_t$. Following this interaction, the agent transitions to the new state $\mathbf{x}_{t+1}$ and receives a scalar reward value r_t

where $\gamma \in [0, 1]$ is the discount factor that influences the importance of future rewards and balances the trade-off between short-term and long-term rewards. A γ value close to 0 implies that the agent focuses on immediate rewards, while a γ value close to 1 suggests that the agent considers future rewards. The policy π can be deterministic or stochastic, depending on the action space. In a deterministic policy, the agent selects the action with the highest probability, whereas in a stochastic policy, the agent selects the action based on the probability distribution over the action space.

When addressing real-world problems, it is common for the problem at hand to be complex and high-dimensional, making it difficult to directly learn the optimal policy. Additionally, the state transition dynamics are often not known, and the state is not fully observable. Consequently, such problems are often characterized as Partially Observable Markov Decision Processes (POMDPs) [139], which is an extension of the MDP framework that takes into account the agent's partial observation of the environment. The POMDP is specified by a tuple $(\mathcal{X}, \mathcal{A}, \Omega, Pr, R, O)$, where Ω represents the observation space, and O is a set of conditional observation probabilities. In this framework, the agent's observation $\mathbf{o}_t \in \Omega$ serves as a noisy and partial representation of the state $\mathbf{x}_t$, and is generated by the observation function $O(\mathbf{o}_{t+1} \mid \mathbf{x}_{t+1}, \mathbf{a}_t)$. In this setting, the agent's objective remains the same as in the MDP framework, but the policy $\pi(\mathbf{a}_t \mid \mathbf{o}_t)$ is conditioned on the observation $\mathbf{o}_t$ rather than the state $\mathbf{x}_t$. The graphical models for MDP and POMDP are depicted in Figure 2.2.

The problem defined as an MDP or POMDP is commonly addressed using either value-based or policy-based methods. Value-based methods involve estimating the value function $V^\pi(\mathbf{x})$ or the action-value func-

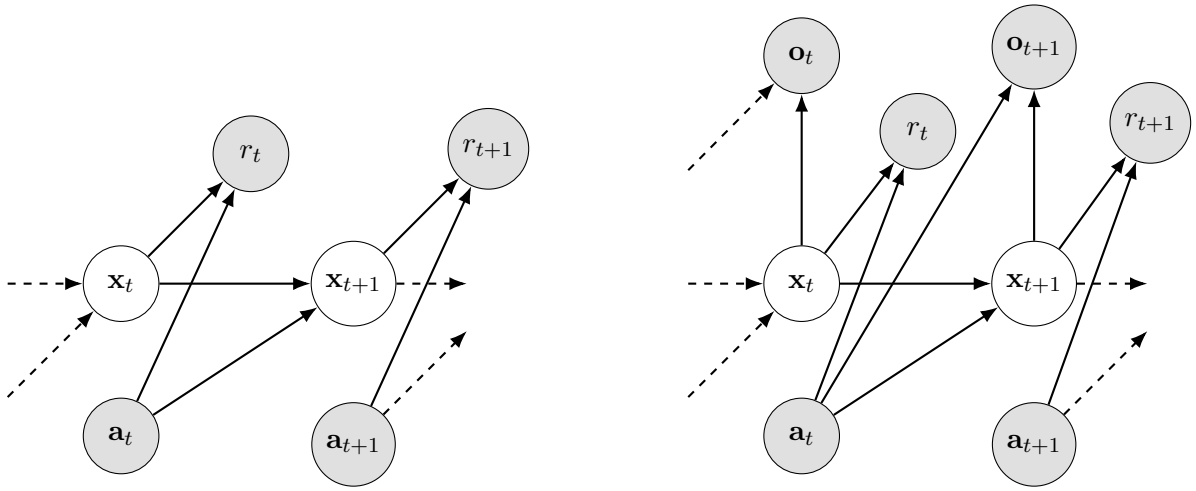


Figure 2.2: Probabilistic graphical models for MDP and POMDP. **Left:** MDP graphical model, where the agent interacts with the environment by selecting actions $\mathbf{a}_t$ based on the observed state $\mathbf{x}_t$. **Right:** POMDP graphical model, where the agent interacts with the environment by selecting actions $\mathbf{a}_t$ based on the partial observation $\mathbf{o}_t$ of the state $\mathbf{x}_t$.

tion $Q^\pi(\mathbf{x}, \mathbf{a})$ to determine the optimal policy. On the other hand, policy-based methods directly learn the policy $\pi(\mathbf{a}_t \mid \mathbf{x}_t)$ to maximize the expected return. Subsequent sections will dive into both approaches in detail.

2.1.1 Value-based Methods

Value-based methods are used to estimate the optimal policy π^* based on the cumulative reward obtained following the policy π from an initial state $\mathbf{x}_0$ to a terminal state $\mathbf{x}_t^d$. These methods are typically solved using dynamic programming (DP) [7], which iteratively improves the value function estimates until convergence. In the context of solving MDP, the Bellman equation plays a fundamental role in decomposing the value function into two components: the immediate reward r_t and the discounted value of the next state $\mathbf{x}_{t+1}$. For the state-value function $V^\pi(\mathbf{x})$, the Bellman equation takes the form:

$$V^\pi(\mathbf{x}) = \mathbb{E}_\pi \left[r_t + \gamma V^\pi(\mathbf{x}_{t+1}) \mid \mathbf{x}_t = \mathbf{x} \right] \tag{2.4}$$

where $\mathbb{E}_\pi$ denotes the expected value of a random variable given that the agent follows policy π, and t is the time step. It is important to note that the value function for the terminal state $\mathbf{x}_t^d$ is always zero, as there are no future rewards to be collected. Following the formulation of the Bellman equation for the value function, the optimal value function V^* has to satisfy the following formula:

$$V^*(\mathbf{x}) = \max_\pi V^\pi(\mathbf{x}) \tag{2.5}$$

for all states $\mathbf{x} \in \mathcal{X}$. Furthermore, for the action-value function $Q^\pi(\mathbf{x}, \mathbf{a})$, the Bellman equation takes the form:

$$Q^\pi(\mathbf{x}, \mathbf{a}) = \mathbb{E}_\pi \left[r_t + \gamma Q^\pi(\mathbf{x}_{t+1}, \mathbf{a}_{t+1}) \mid \mathbf{x}_t = \mathbf{x}, \mathbf{a}_t = \mathbf{a} \right] \tag{2.6}$$

Subsequently, the optimal action-value function Q^* is defined as follows:

$$Q^*(\mathbf{x}, \mathbf{a}) = \max_{\pi} Q^{\pi}(\mathbf{x}, \mathbf{a}) \tag{2.7}$$

for all states $\mathbf{x} \in \mathcal{X}$ and actions $\mathbf{a} \in \mathcal{A}$. By introducing the Bellman equations for the value function $V^{\pi}(\mathbf{x})$ and the action-value function $Q^{\pi}(\mathbf{x}, \mathbf{a})$, the optimal policy π^* can be determined by selecting the action $\mathbf{a}$ that maximizes the action-value function $Q^*(\mathbf{x}, \mathbf{a})$:

$$\pi^*(\mathbf{a} \mid \mathbf{x}) = \underset{\mathbf{a}}{\text{argmax}} \mathbf{a} Q^*(\mathbf{x}, \mathbf{a}) \tag{2.8}$$

Temporal-Difference Learning

One of the most widely used value-based approaches to determine the optimal policy π^* is Temporal-Difference Learning (TD) [109]. This method combines concepts from both Monte Carlo (MC) [101] and dynamic programming (DP) [7]. Comparable to MC techniques, TD learning extracts knowledge directly from raw experience without the need for a model of the dynamics of the environment. Currently, similar to DP methods, TD learning adjusts the value function estimates according to the Bellman equation. Benefiting from aspects of both MC and DP, TD learning achieves faster convergence compared to MC methods and improved sample efficiency relative to DP methods. Unlike MC techniques, TD learning operates without necessitating knowledge of the environment's dynamics, making it particularly applicable to continuous state and action spaces.

The TD(0) algorithm, which constitutes the fundamental form of TD learning, iteratively estimates and updates the value function $V^{\pi}(\mathbf{x})$ by computing the temporal difference error between the current estimate and the target value. The TD error is defined as the difference between the sum of the immediate reward r_t and the discounted value of the subsequent state $\mathbf{x}_{t+1}$, and the current value estimate $V(\mathbf{x}_t)$. Thus, the update rule for the value function is expressed as:

$$V(\mathbf{x}_t) \leftarrow V(\mathbf{x}_t) + \eta \underbrace{\Big[\overbrace{r_t + \gamma V(\mathbf{x}_{t+1})}^{\text{TD-target}} - V(\mathbf{x}_t)\Big]}_{\text{TD-error}} \tag{2.9}$$

where η denotes the learning rate and γ represents the discount factor. A summary of the TD(0) algorithm for estimating the value function $V^{\pi}(\mathbf{x})$ is presented in Algorithm 1.

Q-learning

One of the important advances in the field of reinforcement learning came with the introduction of Q-learning [121]. Q-learning is an off-policy temporal difference learning algorithm. The $Q(\mathbf{x}, \mathbf{a})$ function is updated by solving the following equation:

$$Q(\mathbf{x}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{x}_t, \mathbf{a}_t) + \eta \Big[r_t + \gamma \max_{\mathbf{a}} Q(\mathbf{x}_{t+1}, \mathbf{a}) - Q(\mathbf{x}_t, \mathbf{a}_t)\Big] \tag{2.10}$$

where η denotes the learning rate, $\max_{\mathbf{a}} Q(\mathbf{x}_{t+1}, \mathbf{a})$ represents the maximum Q-value for the next state $\mathbf{x}_{t+1}$, and γ is the discount factor. In particular, Q-learning directly estimates the optimal action-value function $Q^*(\mathbf{x}, \mathbf{a})$ independently of the policy being followed.

Algorithm 1 TD(0) algorithm for estimating the value function $V^{\pi}(\mathbf{x})$

Parameters: learning rate $\eta \in (0, 1]$
Initialize $V(\mathbf{x})$, for all $\mathbf{x} \in \mathcal{X}$, arbitrarily except that $V(\mathbf{x}_t^d) = 0$
for *each episode* **do**
  Initialize state $\mathbf{x}_t$
  for *each step of episode* **do**
    Take action $\mathbf{a}_t$ according to policy $\pi(\mathbf{a}_t \mid \mathbf{x}_t)$
    Observe reward r_t and next state $\mathbf{x}_{t+1}$
    $V(\mathbf{x}_t) \leftarrow V(\mathbf{x}_t) + \eta \left[r_t + \gamma V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t)\right]$
    $\mathbf{x}_t \leftarrow \mathbf{x}_{t+1}$
  end for
end for

The interaction between the agent and the environment is facilitated by the behavior policy π_b, which is instrumental in exploring the environment. When dealing with a discrete action space, the behavior policy typically employs a ϵ-greedy exploration strategy. This strategy selects the action with the highest Q-value with probability $1 - \epsilon$ and picks a random action with probability ϵ, effectively balancing the trade-off between exploration and exploitation. A common practice is to gradually decrease the value of ϵ over time to reduce the exploration rate as the agent learns more about the environment. This approach allows the agent to improve its understanding of the environment while also leveraging its acquired knowledge. For further details, the Q-learning algorithm is outlined in Algorithm 2.

Algorithm 2 Q-learning algorithm

Parameters: learning rate $\eta \in (0, 1]$, exploration rate $\epsilon > 0$
Initialize $Q(\mathbf{x}, \mathbf{a})$, for all $\mathbf{x} \in \mathcal{X}$ and $\mathbf{a} \in \mathcal{A}$, arbitrarily except that $Q(\mathbf{x}_t^d, \mathbf{a}) = 0$
for *each episode* **do**
  Initialize state $\mathbf{x}_t$
  for *each step of episode* **do**
    Choose action $\mathbf{a}_t$ using policy derived from Q (e.g., ϵ-greedy)
    Take action $\mathbf{a}_t$, observe reward r_t and next state $\mathbf{x}_{t+1}$
    $Q(\mathbf{x}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{x}_t, \mathbf{a}_t) + \eta \left[r_t + \gamma \max_{\mathbf{a}} Q(\mathbf{x}_{t+1}, \mathbf{a}) - Q(\mathbf{x}_t, \mathbf{a}_t)\right]$
    $\mathbf{x}_t \leftarrow \mathbf{x}_{t+1}$
  end for
end for

2.1.2 Policy Gradient Methods

In the preceding section, the concept of value-based methods has been introduced, which involve estimating the value function to determine the optimal policy. To achieve this, most of these methods rely on selecting actions based on the estimated action-values. On the contrary, the policy gradient methods directly learn the

policy through optimization of the policy parameters, denoted as $\boldsymbol{\theta} \in \mathbb{R}^d$, with the goal of maximizing the expected return. Consequently, the policy is represented as a probability distribution

$$\pi_\theta(\mathbf{a}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) = p(\mathbf{a}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) \tag{2.11}$$

that maps the state $\mathbf{x}_t$ and the parameter vector $\boldsymbol{\theta}$ to the action $\mathbf{a}_t$. One advantage of directly learning the policy lies in its independence from the need to estimate the value function, which makes it particularly suitable for problems with high-dimensional action spaces. The optimization of the policy parameters is accomplished based on the gradient of an objective function $\mathcal{J}(\boldsymbol{\theta})$ with respect to the policy parameters $\boldsymbol{\theta}$. Since the aim of the RL agent is to maximize the expected return, the gradient ascent method is employed to update the parameters as follows:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \eta \nabla \mathcal{J}(\boldsymbol{\theta}_t) \tag{2.12}$$

where $\nabla \mathcal{J}(\boldsymbol{\theta}_t)$ signifies the gradient of the objective function $\mathcal{J}(\boldsymbol{\theta}_t)$ with respect to the policy parameters $\boldsymbol{\theta}_t$, and η denotes the learning rate. Approaches that adhere to this method are commonly known as *policy gradient methods*, regardless of whether they learn the approximate value function. For methods that learn the value function alongside the policy, the term *actor-critic methods* is used, with the *actor* denoting the policy and *critic* denoting the value function, typically a state-value function. Actor-critic methods prove to be particularly valuable in mitigating the variance of the gradient estimates and enhancing learning stability.

One of the key benefits of policy gradient methods is their ability to represent the policy as a distribution. This allows the approximated policy to gradually approach the deterministic policy. On the contrary, value-based methods that use an ϵ-greedy exploration strategy may struggle to converge to the optimal policy, as there is always a chance of selecting a suboptimal action. Additionally, parameterizing the policy as a distribution allows it to be stochastic, which is advantageous in environments with unpredictable dynamics and significantly enhances the agent's exploration capabilities.

REINFORCE Algorithm

One of the most fundamental policy gradient methods is the REINFORCE algorithm [123], which is based on the likelihood ratio policy gradient theorem [111, 78]. In this approach, the policy gathers a trajectory τ of length T by interacting with the environment and calculates the *reward-to-go*, denoted as $\hat{G}_t$, at each time step t. The reward-to-go is computed as the sum of the rewards obtained from time step t to the end of the trajectory, discounted by the factor γ, as shown in the equation below:

$$\hat{G}_t = \sum_{k=t}^{T} \gamma^{k-t} r_k \tag{2.13}$$

The length of the trajectory T can be limited to a fixed time horizon that spans multiple episodes or can be set to the length of a single episode. Once the trajectory is collected, the policy parameters $\boldsymbol{\theta}$ are updated based on the gradient of the action probability $\mathbf{a}_t$ with respect to the policy parameters, scaled by the reward-to-go $\hat{G}_t$. The policy gradient theorem assumes the differentiability of the action probability $\mathbf{a}_t$ with respect to the policy parameters $\boldsymbol{\theta}$. However, in practical scenarios, the environment model is unknown and the policy is stochastic, making it infeasible to estimate the gradient based on the trajectory probabilities.

Therefore, the REINFORCE algorithm approximates the gradient of the policy parameters using the log-likelihood of the action $\mathbf{a}_t$, as expressed by:

$$\nabla \mathcal{J}(\boldsymbol{\theta}) \propto \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{x}_t) \hat{G}_t \right] \tag{2.14}$$

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \eta \nabla \mathcal{J}(\boldsymbol{\theta}) \tag{2.15}$$

where $\nabla \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{x}_t)$ denotes the gradient of the log-likelihood of the action $\mathbf{a}_t$ with respect to the policy parameters $\boldsymbol{\theta}$. The complete REINFORCE algorithm is outlined in Algorithm 3.

Despite its simplicity and proven convergence properties, the REINFORCE algorithm is plagued by high variance in gradient estimates, which can impede learning progress and destabilize training. Moreover, this issue frequently results in becoming stuck in local minima, hindering the agent's ability to learn the optimal policy. The primary cause of this high variance is the reward-to-go $\hat{G}_t$, which represents a noisy estimate of the return due to the stochastic nature of trajectory collection.

Algorithm 3 REINFORCE algorithm

Parameters: learning rate $\eta \in (0, 1]$
Initialize policy parameters $\boldsymbol{\theta}$ (e.g., randomly)
for *each episode* **do**
  Collect trajectory $\tau = \{(\mathbf{x}_0, \mathbf{a}_0, r_0), (\mathbf{x}_1, \mathbf{a}_1, r_1), \ldots, (\mathbf{x}_T, \mathbf{a}_T, r_T)\}$ of length T using $\pi(\mathbf{a}_t \mid \mathbf{x}_t, \boldsymbol{\theta})$
  for *each step of trajectory* $t = 0, 1, \ldots, T$ **do**
    $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ ▷ Compute the return of the trajectory
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \nabla \mathcal{J}(\boldsymbol{\theta})$ ▷ Update policy parameters using (2.14)
  end for
end for

Actor-Critic Methods

The REINFORCE algorithm, although effective in learning the policy, is hindered by high variance in the gradient estimates, which can impede the learning process. However, Williams [123] demonstrated that the policy gradient theorem can be generalized to include a comparison of the reward-to-go to an arbitrary *baseline* $b(\mathbf{x}_t)$:

$$\nabla \mathcal{J}(\boldsymbol{\theta}) \propto \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{x}_t) (\hat{G}_t - b(\mathbf{x}_t)) \right] \tag{2.16}$$

The baseline $b(\mathbf{x}_t)$ can be any function depending on the state $\mathbf{x}_t$. A natural choice for the baseline is the value function $V^{\pi}(\mathbf{x}_t)$ parameterized by $\boldsymbol{\omega}$, which estimates the expected return from the state $\mathbf{x}_t$. By using the value function as a baseline, the variance of the gradient estimates can be reduced, resulting in more stable learning. This is known as the *actor-critic* method. It is important to note that the actor-critic method is a general framework that can be implemented using various policy gradient algorithms and value function estimation methods.

One of the simplest actor-critic methods is the One-step Actor-Critic algorithm, which combines the REINFORCE algorithm with TD(0) learning. In this approach, the policy parameters $\boldsymbol{\theta}$ are optimized using

formula (2.16), while the parameters of the value function $\boldsymbol{\omega}$ are updated using the TD(0) update rule (2.9). This means that optimization occurs at every step of the episode, rather than waiting until the trajectory is complete. The pseudocode for the one-step actor-critic algorithm is presented in Algorithm 4.

Algorithm 4 One-step Actor-Critic algorithm

Parameters: learning rates $\eta_\pi, \eta_V \in (0, 1]$
Initialize policy parameters $\boldsymbol{\theta}$ and value function parameters $\boldsymbol{\omega}$ (e.g., randomly)
for *each episode* **do**
  Initialize state $\mathbf{x}_t$
  Initialize importance sampling ratio $I = 1$
  for *each step of episode* **do**
    $\mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{x}_t)$ ▷ Sample action from policy
    $\mathbf{x}_{t+1} \sim Pr(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t), r_t = R(\mathbf{x}_t, \mathbf{a}_t)$ ▷ Observe reward and next state
    $\delta_t = r_t + \gamma V_{\boldsymbol{\omega}}(\mathbf{x}_t^{'}) - V_{\boldsymbol{\omega}}(\mathbf{x}_t)$ ▷ Compute TD error
    $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \eta_V \delta_t \nabla_{\boldsymbol{\omega}} V_{\boldsymbol{\omega}}(\mathbf{x}_t)$ ▷ Update value function parameters
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_\pi I \delta_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{x}_t)$ ▷ Update policy parameters
    $I \leftarrow \gamma I$ ▷ Update importance sampling ratio
    $\mathbf{x}_t \leftarrow \mathbf{x}_{t+1}$
  end for
end for

2.1.3 On-policy and Off-policy

There are two primary categories of reinforcement learning methods: on-policy and off-policy. On-policy methods, such as REINFORCE [123], learn the policy $\pi(\mathbf{a}_t \mid \mathbf{x}_t, \boldsymbol{\theta})$ that is the same as the behavior policy π_b. This approach introduces sensitivity to the exploration strategy, as the policy is updated based on the actions selected by the behavior policy. Therefore, the optimization process can produce suboptimal policies with more conservative behavior. Moreover, due to the way the policy is updated, on-policy methods require a large number of samples to learn the optimal policy. However, the training process is more stable and less sensitive to the choice of hyper-parameters.

In contrast, off-policy methods, such as Q-learning [121], learn the policy $\pi(\mathbf{a}_t \mid \mathbf{x}_t, \boldsymbol{\theta})$ that is different from the behavior policy π_b. This distinction allows off-policy methods to learn from historical data, improving sample efficiency, and incorporating more aggressive exploration strategies. However, despite being able to converge faster than on-policy methods, off-policy methods are more sensitive to the choice of hyper-parameters and require careful tuning to ensure convergence. This drawback is significant when the method is parameterized by a function approximator, such as a neural network, as the training process can become unstable.

2.1.4 Model-free and Model-based Methods

In the reinforcement learning framework, two main categories of methods can be distinguished: model-free and model-based methods. Model-free methods, exemplified by Q-learning [121] and REINFORCE [123], directly infer the policy or value function from the gathered data without explicitly modeling the environment's dynamics. In contrast, model-based methods, such as Dyna-Q [110] and PILCO [16], learn a model of the environment's dynamics and utilize it for planning the agent's actions. Thus, it can be stated that model-based methods rely on *planning* as the primary mechanism for policy improvement, while model-free methods rely primarily on *learning*.

In practice, model-based methods are more sample-efficient than model-free methods, as they can exploit the learned model to plan the agent's actions. This planning process is facilitated by the model's ability to estimate transition dynamics $\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t)$ and predict reward $r_t \sim p(r_t \mid \mathbf{x}_t, \mathbf{a}_t)$, enabling the simulation of the environment and prediction of hypothetical trajectories. Formerly, the model was represented as a tabular lookup table, updated using the collected data. Currently, the model is often depicted as a neural network (e.g., MuZero [97], and Dreamer [32]) trained through supervised learning to predict the next state and reward.

However, model-based methods are more sensitive to model errors, potentially resulting in suboptimal performance in practical applications. In contrast, model-free methods are not influenced by model errors, but they necessitate more samples to learn the policy. Nevertheless, both approaches are based on the same fundamental principles for computing the value function $V^\pi(\mathbf{x})$ or the action-value function $Q^\pi(\mathbf{x}, \mathbf{a})$. Furthermore, all of the methods stem from the practice of anticipating future events, computing a backed-up value, and utilizing it as an update target for an approximate value function.

2.1.5 Deep Reinforcement Learning

The early research in reinforcement learning algorithms, exemplified by Q-learning, were initially designed for tabular representations of the state-action space, presenting efficiency in solving problems with small state-action spaces through dynamic programming methods. However, this approach faces scalability issues when dealing with continuous state and action spaces, as the number of bins required to discretize the state-action space expands exponentially with the number of dimensions. Furthermore, the quantization process may result in the loss of crucial information, significantly impacting the learning process.

To address these limitations, an alternative approach involves employing linear function approximators to generalize the value function across the state-action space; however, these approximators might not adequately represent complex functions, leading to suboptimal performance in high-dimensional state-action spaces. In addition, both tabular and linear function approximators are affected by the curse of dimensionality, making them inadequate for real-world problems such as robotic manipulation.

In response to these challenges, deep reinforcement learning methods have emerged, using deep neural networks as function approximators to learn the value function or policy. Nonetheless, training deep neural networks using reinforcement learning poses significant challenges due to the instability of the learning process, particularly in the case of off-policy algorithms. The pioneering Deep Q-Network (DQN) algorithm, introduced by Mnih et al. [82], successfully integrated neural networks with off-policy reinforcement

learning, achieving human-level performance in playing Atari games. This achievement was facilitated by techniques such as experience replay and target networks, which were proposed to stabilize the training process.

Target networks are used to stabilize the learning process by fixing the target Q-values for a specific number of iterations or by updating the target network parameters using an exponential moving average. Meanwhile, experience replay involves storing the agent's experiences in a replay buffer and sampling mini-batches of experiences to train the RL agent.

The success of the DQN algorithm has spurred the development of various DRL algorithms, including the DDPG [67], the Twin Delayed Deep Deterministic Policy Gradient (TD3) [24], and the SAC [31], which have demonstrated efficacy across a broad spectrum of control tasks, including robotic manipulation. These advances highlight the evolving landscape of reinforcement learning and its increasing applicability to complex real-world problems.

2.1.6 Soft Actor-Critic

The Soft Actor-Critic is an off-policy actor-critic algorithm that integrates the actor-critic optimization approach with the maximum entropy reinforcement learning framework, as introduced by Haarnoja et al. [31]. This framework encourages the agent to explore the environment by maximizing the entropy of the policy. Additionally, SAC learns a stochastic policy that enables the generation of a diverse set of actions, which is advantageous for exploration. This algorithm has shown notable effectiveness in leveraging samples efficiently and maintaining stability in continuous control tasks. It has been successfully applied to a wide range of robotic manipulation tasks, including dexterous manipulation and robotic assembly. Consequently, SAC is currently one of the most widely used algorithms for contact-rich robotic manipulation tasks.

As mentioned above, SAC is an off-policy actor-critic algorithm; thus its architecture uses neural networks to approximate the policy $\pi_{\boldsymbol{\phi}}(\mathbf{a}_t \mid \mathbf{x}_t)$ and the soft Q-function $Q_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{a}_t)$, which are parameterized by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. The soft Q-function parameters are updated by minimizing the *soft Bellman residual*, which is defined as follows:

$$\mathcal{J}_Q(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}_t, \mathbf{a}_t \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_{\boldsymbol{\theta}} \left(\mathbf{x}_t, \mathbf{a}_t \right) - \left(r_t + \gamma V_{\bar{\boldsymbol{\theta}}} \left(\mathbf{x}_{t+1} \right) \right) \right)^2 \right] \tag{2.17}$$

where $\mathcal{D}$ is an experience replay buffer that stores transitions $(\mathbf{x}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{x}_{t+1})$, and y_t denotes the *soft value target* that can be approximated by the minimum value of the soft Q-functions. The soft value target is computed as:

$$V_{\bar{\boldsymbol{\theta}}} \left(\mathbf{x}_t \right) = \min_{i \in \{1,2\}} Q_{\bar{\boldsymbol{\theta}}_i} \left(\mathbf{x}_t, \mathbf{a}_t \right) - \alpha \log \pi_{\boldsymbol{\phi}} \left(\mathbf{a}_t \mid \mathbf{x}_t \right) \tag{2.18}$$

where $\bar{\boldsymbol{\theta}}$ is the target soft Q-function, and α is an entropy temperature coefficient. The parameters $\bar{\boldsymbol{\theta}}$ of the target soft Q-function are obtained as an exponential moving average of the weights of the soft Q-function. SAC uses two soft Q-functions with independent parameters to mitigate positive bias in the policy improvement steps. The target Q-value is computed by taking the minimum value from the Q-function approximations. Both networks are independently optimized by solving the objectives $\mathcal{J}_{Q_i}(\boldsymbol{\theta}_i)$. The Gaussian

policy parameters $\boldsymbol{\phi}$ are trained by minimizing

$$\mathcal{J}_\pi(\boldsymbol{\phi}) = \mathbb{E}_{\mathbf{x}_t \sim \mathcal{D}} \left[\alpha \log \pi_{\boldsymbol{\phi}} \left(\mathbf{a}_t | \mathbf{x}_t \right) - \min_{i \in \{1,2\}} Q_{\boldsymbol{\theta}_i} \left(\mathbf{x}_t, \mathbf{a}_t \right) \right] \tag{2.19}$$

using the reparameterization trick [55]. Finally, the entropy temperature coefficient α can be fixed during training or dynamically adjusted, as proposed by [30, 115]. This coefficient can be optimized by solving the following objective:

$$\mathcal{J}(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \mathcal{D}} \left[\alpha \log \pi_{\boldsymbol{\phi}}(\mathbf{a}_t \mid \mathbf{x}_t) - \alpha \bar{\mathcal{H}} \right] \tag{2.20}$$

where $\bar{\mathcal{H}}$ is a target entropy that is usually set empirically to $\bar{\mathcal{H}} = \dim(\mathcal{A})$. The SAC algorithm with automatic entropy tuning is summarized in Algorithm 5.

Algorithm 5 Soft Actor-Critic Algorithm

Parameters: learning rates $\eta_Q, \eta_\pi, \eta_\alpha$, target update coefficient υ
Initialize function approximators parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\phi}$, temperature coefficient α
Replay buffer $\mathcal{D}$
for *each iteration* **do**
  Initialize state $\mathbf{x}_t$
  for *each environment step* **do**
    $\mathbf{a}_t \sim \pi_{\boldsymbol{\theta}} \left(\mathbf{a}_t | \mathbf{x}_t \right)$ ▷ Sample action from policy
    $\mathbf{x}_{t+1} \sim Pr \left(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t \right), r_t = R \left(\mathbf{x}_t, \mathbf{a}_t \right)$ ▷ Sample next state and reward
    $\mathcal{D} \leftarrow \mathcal{D} \cup \left(\mathbf{x}_t, \mathbf{a}_t, r \left(\mathbf{x}_t, \mathbf{a}_t \right), \mathbf{x}_{t+1} \right)$ ▷ Store transition in replay buffer
    $\mathbf{x}_t \leftarrow \mathbf{x}_{t+1}$
  end for
  for *each update step* **do**
    $\{(\mathbf{x}_t, \mathbf{a}_t, r_t, \mathbf{x}_{t+1})\} \sim \mathcal{D}$ ▷ Sample batch of transitions from replay buffer
    $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_Q + \eta_Q \nabla_{Q_{\boldsymbol{\theta}}} \mathcal{J}_Q(\boldsymbol{\theta}_i)$ for $i \in \{1, 2\}$ ▷ Update Q-function parameters with (2.17)
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \eta_\pi \nabla_{\pi_{\boldsymbol{\phi}}} \mathcal{J}_\pi(\boldsymbol{\phi})$ ▷ Update policy parameters with (2.19)
    $\alpha \leftarrow \alpha + \eta_\alpha \nabla_\alpha \mathcal{J}_\alpha(\alpha)$ ▷ Update temperature coefficient with (2.20)
  end for
  $\bar{\boldsymbol{\theta}}_i \leftarrow (1 - \upsilon) \bar{\boldsymbol{\theta}}_i + \upsilon \boldsymbol{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update $\bar{\boldsymbol{\theta}}_i$ using EMA of $\boldsymbol{\theta}_i$, where υ defines Polyak coefficient
end for

Squashed Gaussian Policy

The SAC algorithm utilizes a Gaussian policy to generate stochastic actions, which, by default, can result in unbounded action values. In scenarios where the policy interacts with real-world systems, such as robotic manipulators, the unbounded action space can pose safety risks and potential harm to the system. To address this, Haarnoja et al. [31] introduced a modification to the Gaussian policy by incorporating a squashing function to confine the action space to a bounded interval. The sampled action $\mathbf{a}_t$ is transformed using the $\tanh$ function, which maps the action to the interval $[-1, 1]$. Given $\mathbf{z} \in \mathbb{R}^d$ as a sample from a standard

Gaussian distribution and $\pi(\mathbf{z} \mid \mathbf{x})$ as the corresponding density with infinite support, the random variable $\mathbf{a} = \tanh(\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation of the Gaussian policy, has support in $[-1, 1]$ with density given by:

$$\pi(\mathbf{a} \mid \mathbf{x}) = \pi(\mathbf{z} \mid \mathbf{x}) \left| \det \left(\frac{d\mathbf{a}}{d\mathbf{z}} \right) \right| \tag{2.21}$$

The objective of policy optimization requires the log-likelihood of the policy distribution. Given that the Jacobian of the $\tanh$ function is $\frac{d\mathbf{a}}{d\mathbf{z}} = \operatorname{diag}\left(1 - \tanh^2(\mathbf{z})\right)$ and is diagonal, the log-likelihood of the squashed Gaussian policy can be expressed as:

$$\log \pi(\mathbf{a} \mid \mathbf{x}) = \log \pi(\mathbf{z} \mid \mathbf{x}) - \sum_{i=1}^{d} \log(1 - \tanh^2(\mathbf{z}_i)) \tag{2.22}$$

where z_i is the i-th element of the vector $\mathbf{z}$.

2.2 Compliance Control in Industrial Robotics

In the field of manufacturing, industrial robots predominantly operate in position or velocity control modes. While these control strategies are suitable for general applications, they are not well-suited for tasks that require the robot end-effector to manipulate objects or perform operations on surfaces. Examples of such tasks include deburring, polishing, and assembly, where the robot needs to be compliant with external forces to prevent damage to the workpiece or the robot itself. Compliance control systems address these tasks by allowing the robot to adapt to external forces while maintaining high precision.

When considering compliance or interaction control, two primary approaches can be distinguished: passive and active interaction control. Passive interaction control relies on the mechanical compliance of the robot structure or the use of passive compliance devices such as flexible grippers. On the contrary, active interaction control involves the use of force sensors and control algorithms to regulate the interaction forces between the robot and the environment.

The passive approach to compliance control is characterized by its simplicity and cost-effectiveness. However, it is limited in terms of flexibility and adaptability. Specifically, for specific tasks, a new compliant end-effector must be designed and installed on the robot. In contrast, active compliance control utilizes force sensors and control algorithms to regulate interaction forces, enabling a wide range of tasks without requiring modification of the robot's end-effector, as control parameters can be adjusted on-the-fly. However, active compliance control has a slower response time compared to passive compliance control, as it requires additional computational resources to process force sensor data and adjust control signals.

This section will provide an in-depth discussion of the most common active compliance control strategies used in industrial robotics, given their relevance to the scope of this dissertation.

2.2.1 Notation Introduction

In order to delve into the discussion of compliance control strategies, it is essential to first provide a brief overview of the symbols used in this section. The position of the robot end-effector is expressed as $\boldsymbol{x} =$

$[x, y, z, q_w, q_x, q_y, q_z]^T \in \mathbb{R}^7$, where x, y, z denote the Cartesian coordinates of the position of end-effector and q_w, q_x, q_y, q_z represent its orientation in quaternion form. Following this notation, the twist of the end-effector is denoted as $\dot{\boldsymbol{x}} = [\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T \in \mathbb{R}^6$, where $\dot{x}, \dot{y}, \dot{z}$ represent the linear velocity of the end-effector, and $\omega_x, \omega_y, \omega_z$ denote the angular velocity. Finally, the wrench of forces and torques acting on the end-effector is represented as $\mathcal{W} = [F_x, F_y, F_z, \tau_x, \tau_y, \tau_z]^T \in \mathbb{R}^6$, where F_x, F_y, F_z represent the forces and τ_x, τ_y, τ_z denote the torques.

2.2.2 Hybrid Force/Motion Control

In robotic applications, the hybrid force/motion control strategy is commonly used to achieve compliance control. This method involves the use of two mutually orthogonal control loops: one for position control and the other for force control. Initially introduced for torque-actuated robots by Craig and Raiber [15, 89], this control strategy has also been examined in Manson's research [81]. In this section, an in-depth overview of the position-based version of the hybrid force/motion control strategy will be presented.

The hybrid force/motion control system computes two control signals, $\boldsymbol{x}_r^p$ and $\mathcal{W}_r$, for position and force control, respectively. The position control signal, $\boldsymbol{x}_r^p$, is derived based on the error, $\Delta\boldsymbol{x}$, between the desired position of the end-effector, $\boldsymbol{x}_d$, and the current position, $\boldsymbol{x}$. Simultaneously, the force control signal, $\mathcal{W}_r$, is calculated using the error, $\Delta\mathcal{W}$, between the desired contact wrench, $\mathcal{W}_d$, and the measured contact wrench, $\mathcal{W}$. In this approach, both proportional (P) and proportional-integral (PI) gains are commonly used to compute control signals and can be represented as follows:

$$\begin{aligned} \boldsymbol{x}_r^p &= \boldsymbol{x}_d + \mathbf{K}_{Pp}\Delta\boldsymbol{x} + \mathbf{K}_{Ip}\int_0^t \Delta\boldsymbol{x} d\tau \\ \mathcal{W}_r &= \mathcal{W}_d + \mathbf{K}_{Pf}\Delta\boldsymbol{F} + \mathbf{K}_{If}\int_0^t \Delta\mathcal{W} d\tau, \end{aligned} \tag{2.23}$$

with semi positive-definite gain matrices $\mathbf{K}_{Pp}$, $\mathbf{K}_{Ip}$ for position control, and $\mathbf{K}_{Pf}$, $\mathbf{K}_{If}$ for force control. The integration terms in Equation (2.23) serve to eliminate steady-state errors in control signals. Depending on the specific application, different controllers, such as PD or proportional controllers, can be employed.

The control signals $\boldsymbol{x}_r^p$ and $\mathcal{W}_r$ are combined into a common reference signal $\boldsymbol{x}_r$ using the following equation that incorporates two selection matrices $\mathbf{S}_p$ and $\mathbf{S}_f$:

$$\boldsymbol{x}_r = \mathbf{S}_p\boldsymbol{x}_r^p + \mathbf{K}^{-1}\mathbf{S}_f\mathcal{W}_r \tag{2.24}$$

where $\mathbf{K}^{-1}$ represents the inverse stiffness matrix used to map the force control signal to the position control signal. The obtained control signal $\boldsymbol{x}_r$ is represented in the Cartesian space and then mapped to the joint space using the inverse Jacobian matrix $\mathbf{J}^{-1}$ as follows:

$$\boldsymbol{q}_r = \mathbf{J}^{-1}(\boldsymbol{q})\boldsymbol{x}_r \tag{2.25}$$

In practice, the joint control signal $\boldsymbol{q}_r$ is computed using the inverse kinematics (IK) solver, which converts the Cartesian control signal $\boldsymbol{x}_r$ to the joint control signal $\boldsymbol{q}_r$. The hybrid force/motion control system is graphically depicted in Figure 2.3. A key feature of the hybrid approach is the decoupling of the position and force control signals, allowing the robot to adapt to external forces while maintaining high precision in position control. Selection matrices are required to satisfy the relationship $\mathbf{S}_p^T\mathbf{S}_f = \mathbf{0}$ in each configuration. In addition, users need to specify mutually exclusive components of $\boldsymbol{x}_d$ and $\mathcal{W}_d$ for given tasks.

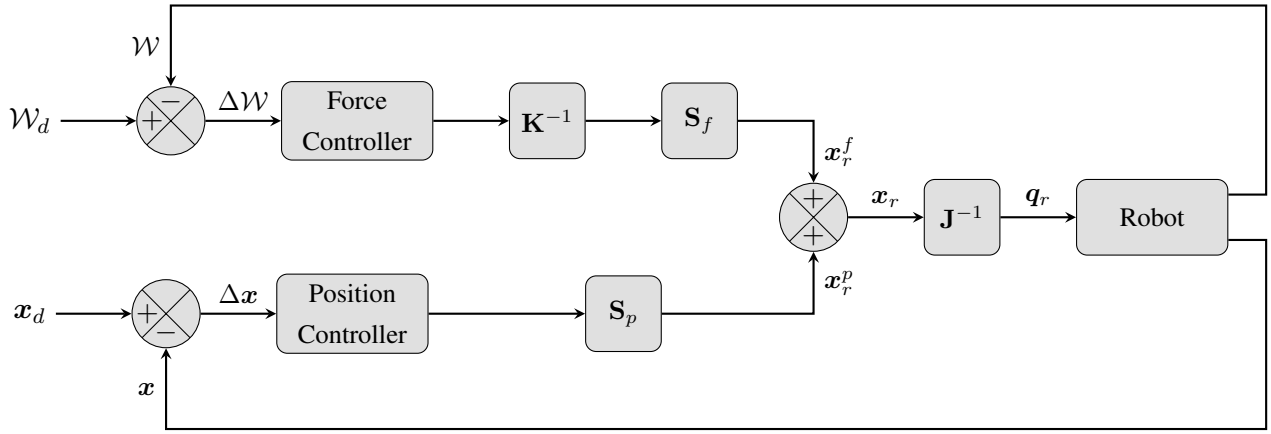


Figure 2.3: Hybrid force/position control system.

2.2.3 Admittance Control

The hybrid force/motion control system, while effective, comes with several limitations. One notable drawback is that the position-controlled degrees of freedom (DOFs) lack compliance, meaning that they do not adapt to external forces acting on the robot. Additionally, this approach fails to offer precise control over the position of the robot's end-effector when it comes to force-controlled DOFs. Alternatively, the admittance control [98] together with the impedance control [41] provides a more adaptable and robust solution to maintain a compliant motion in robotic systems. This is achieved by modeling the end-effector using a dynamical system.

As Hogan [41] pointed out, impedance control can be viewed as a shift from *flow to effort*, while admittance control involves the conversion from *effort to flow*. The dynamics of the end-effector are often represented through a general mass-spring-damper system, as expressed by the equation:

$$\mathbf{M}\Delta\ddot{\boldsymbol{x}}_r + \mathbf{D}\Delta\dot{\boldsymbol{x}} + \mathbf{K}\Delta\boldsymbol{x} = \mathcal{W}, \quad \Delta\boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{x} \tag{2.26}$$

where $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ represent the mass, damping, and stiffness positive-definite matrices, respectively. Impedance control involves substituting the time derivatives of the desired trajectory $\boldsymbol{x}_d(t)$ and the actual trajectory $\boldsymbol{x}(t)$ into Equation (2.26) to derive the control wrench $\mathcal{W}$. The control wrench $\mathcal{W}$ is represented in Cartesian space and then mapped to the joint torques $\boldsymbol{\tau}$ using the Jacobian transpose matrix $\mathbf{J}^T$ according to the equation:

$$\boldsymbol{\tau} = \mathbf{J}^T(\boldsymbol{q})\mathcal{W} \tag{2.27}$$

This relationship is a fundamental aspect of Khatib's operational space formulation [51, 52], which is widely employed in robotics. Conversely, admittance control treats Equation (2.26) as a differential equation of the variable $\boldsymbol{x}$. In this case, the desired trajectory $\boldsymbol{x}_d$, $\dot{\boldsymbol{x}}_d$, and $\ddot{\boldsymbol{x}}_d$ are considered as the reset position of the system. The solution $\boldsymbol{x}(t)$ is obtained by numerically integrating the Equation (2.26). When employing the forward Euler method as a numerical integrator, Equation (2.26) needs to be converted to a set of 1st-order Ordinary Differential Equations (ODEs) in the state-space representation. From the state-space representation, the simulated $\boldsymbol{x}$ is taken as the Cartesian reference position $\boldsymbol{x}_r$ and is mapped to the joint space using the Equation (2.25). The general admittance control system block diagram is depicted in Figure 2.4.

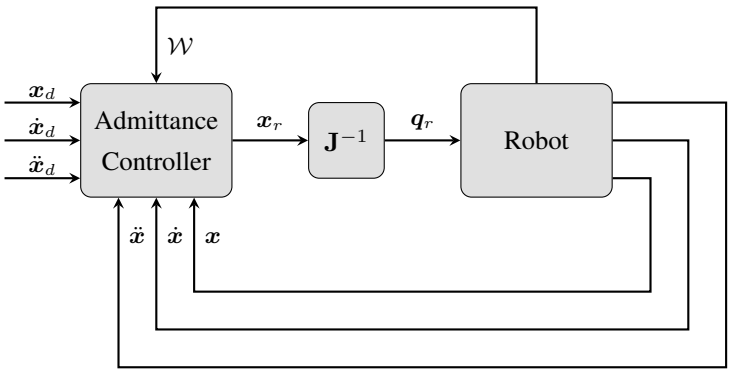


Figure 2.4: Admittance control system.

2.3 Variational Autoencoders

Variational autoencoders [55] are a type of generative models that utilize neural networks to learn the underlying representation of the data distribution. The overview of the VAE architecture is depicted in Figure 2.5. This framework can be seen as a directed graphical model with latent variables, such as Gaussian latent variables, used to model the data distribution. The VAE generative process includes sampling a latent variable $\mathbf{z}$ from a prior distribution $\mathbf{z} \sim p(\mathbf{z})$ and generating the data $\mathbf{x}$ from the conditional distribution $p(\mathbf{x} \mid \mathbf{z})$. The VAE inference process aims to estimate the posterior distribution $q(\mathbf{z} \mid \mathbf{x})$ of the latent variable $\mathbf{z}$ given the data $\mathbf{x}$. The distributions $q(\mathbf{z} \mid \mathbf{x})$ and $p(\mathbf{x} \mid \mathbf{z})$ are modeled by neural networks with parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. However, directly optimizing the parameters is challenging due to the intractability of the posterior distribution. To tackle this challenge, the VAE framework estimates the parameters in the stochastic variational gradient Bayes (SVGB) framework by maximizing the evidence lower bound objective (ELBO [50]), which is defined as follows:

$$\mathcal{J}(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathbf{x}, \mathbf{z}) = \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right]}_{\text{Reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))}_{\text{Regularization term}} \tag{2.28}$$

where $D_{\mathrm{KL}}(\parallel)$ denotes the Kullback-Leibler divergence between the posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ and the prior distribution $p(\mathbf{z})$. The reconstruction term in Equation (2.28) is defined as the negative log-likelihood (NLL) of the data $\mathbf{x}$ given the latent variable $\mathbf{z}$. However, in practice, the generative distribution $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ is often approximated as a Gaussian distribution with a diagonal covariance matrix $\hat{\mathbf{x}} \sim \mathcal{N}(g(\mathbf{z}), \mathbf{I})$, where $g(\mathbf{z})$ denotes the output of the decoder neural network and is treated as the mean of the distribution. Therefore, the reconstruction term is approximated by the mean squared error (MSE) between the input data $\mathbf{x}$ and the reconstructed data $\hat{\mathbf{x}}$:

$$\begin{aligned} \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] &\approx -\frac{1}{2} \sum_{i=1}^{D} \left(1 + \log(\boldsymbol{\sigma}_i^2) - \boldsymbol{\mu}_i^2 - \boldsymbol{\sigma}_i^2\right) \\ &\approx \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \end{aligned} \tag{2.29}$$

In order to make the optimization of Equation (2.28) feasible in practical applications, it is common to make certain assumptions. The prior distribution $p(\mathbf{z})$ and the posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ are often assumed to be Gaussian distributions with diagonal covariance matrices. The prior distribution is defined

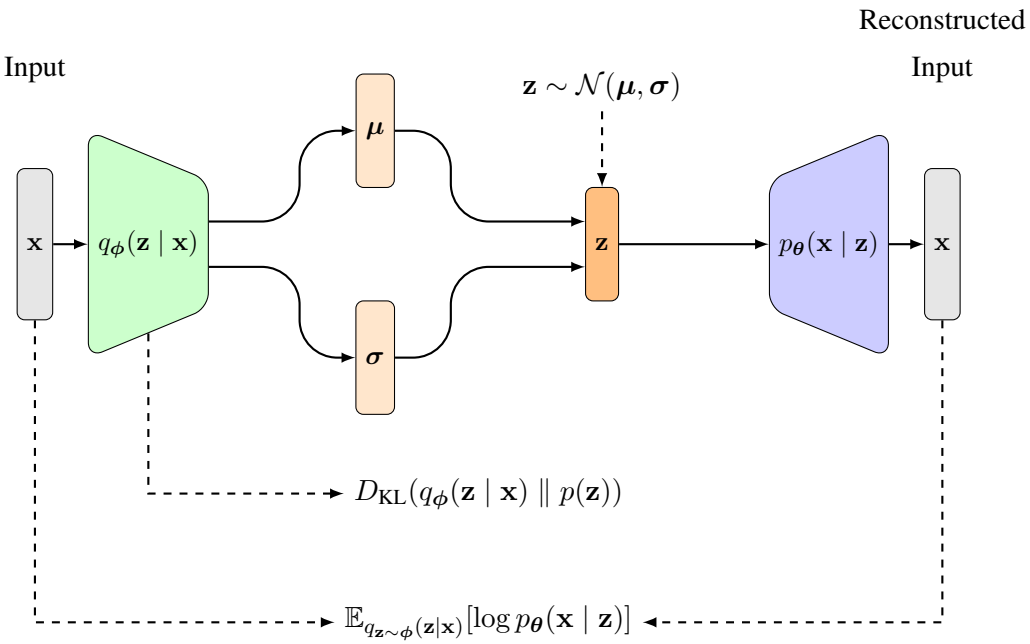


Figure 2.5: Model architecture of the VAE with a Gaussian latent space. The encoder network $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ maps the input data $\mathbf{x}$ to the latent space $\mathbf{z}$. The decoder network $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ reconstructs the input data from the given latent space. The VAE parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are optimized by minimizing the ELBO objective.

as $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{0}$ is the zero vector and $\mathbf{I}$ is the identity matrix. The posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ is parameterized by a neural network, which outputs the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of the Gaussian distribution. Therefore, in order to enable the gradient to flow through the sampling operation, the reparameterization trick [55] is employed as follows:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.30}$$

2.3.1 β-VAE

The vanilla VAE framework is often plagued by the issue of posterior collapse, in which the latent variables are marginalized by the model, resulting in subpar reconstruction quality. In response to this challenge, Higgins et al. [39] introduced the β-VAE framework, which introduces a hyperparameter β to regulate the disentanglement of latent variables. This hyperparameter is integrated into the ELBO objective function as shown below:

$$\mathcal{J}(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathbf{x}, \mathbf{z}) = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] - \beta D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})) \tag{2.31}$$

Choosing an appropriate value for the hyperparameter β is essential in order to achieve the desired disentanglement of the latent variables. When $\beta = 1$, the model follows the standard VAE framework, while setting $\beta > 1$ encourages the model to learn more disentangled representations. On the contrary, when $\beta < 1$, the model focuses more on the quality of the reconstruction. The β-VAE framework has found widespread use in various applications, such as image generation, representation learning, and anomaly detection.

Chapter 3

Framework for Robotic Industrial Assembly Tasks

3.1 Industrial Assembly Tasks as a Reinforcement Learning Problem

In this dissertation, a robotic assembly task, such as insertion of electronic parts in a PCB, is framed as a reinforcement learning problem. The environment is based on the framework for assembling electronic parts tasks proposed by Bartyzel et al. [28]. Within this setup, a robotic manipulator represents the agent and the workspace serves as the environment in which the robot interacts with the task objects. To equip the agent with the information necessary for the execution of the task, a multimodal observation space is constructed. This space comprises two images captured from cameras attached to the end-effector, the tool's current pose relative to the target insertion location, a wrench measurement, and the tool's current twist. The tool's orientation perceived by the RL agent is represented by Euler angles, resulting in a 6-dimensional vector for the pose modality. Figure 3.1 illustrates the process of capturing images using the vision tool. The angle of view of the camera is empirically set at 30° to provide a comprehensive view of the assembly location and its surroundings. The raw images, with an arbitrarily large resolution $H \times W$, are typically too large for real-time processing by the neural network. Therefore, the images are downsampled to $\frac{H}{8} \times \frac{W}{8}$ pixels and then resized to achieve a 1:1 aspect ratio, resulting in $\frac{W}{8} \times \frac{W}{8}$ pixels to facilitate the use of the neural network. In this dissertation, the raw images are 1024×512 pixels in the RGB color space. As a result, the processed images are 128×128 pixels.

Let's denote the robot base frame as $\{\mathbf{r}\}$, the workspace frame as $\{\mathbf{w}\}$, and the tool frame as $\{\mathbf{t}\}$. The workspace frame $\{\mathbf{w}\}$ represents the insertion pose of the currently manipulated object, while the tool frame $\{\mathbf{t}\}$ is restricted to the tip of the attached tool. These reference frames are depicted in Figure 3.2. The robot operates within a cylindrical workspace with a radius of R_{limit} and an infinite height. The robot tool is restricted to a maximum rotation of ψ_{limit} about each axis to prevent damage to the attached cameras and cables. At the beginning of each episode, the robot moves to an initial pose $\boldsymbol{x}_{init}$. Depending on the task requirements, the initial Z-axis position z_{init} can be positioned just above the surface of the workspace or in the free-space above the workspace. Furthermore, the initial Z-axis position can be adjusted during the training process to enhance the agent's learning. For the electronic parts insertion task, the initial position

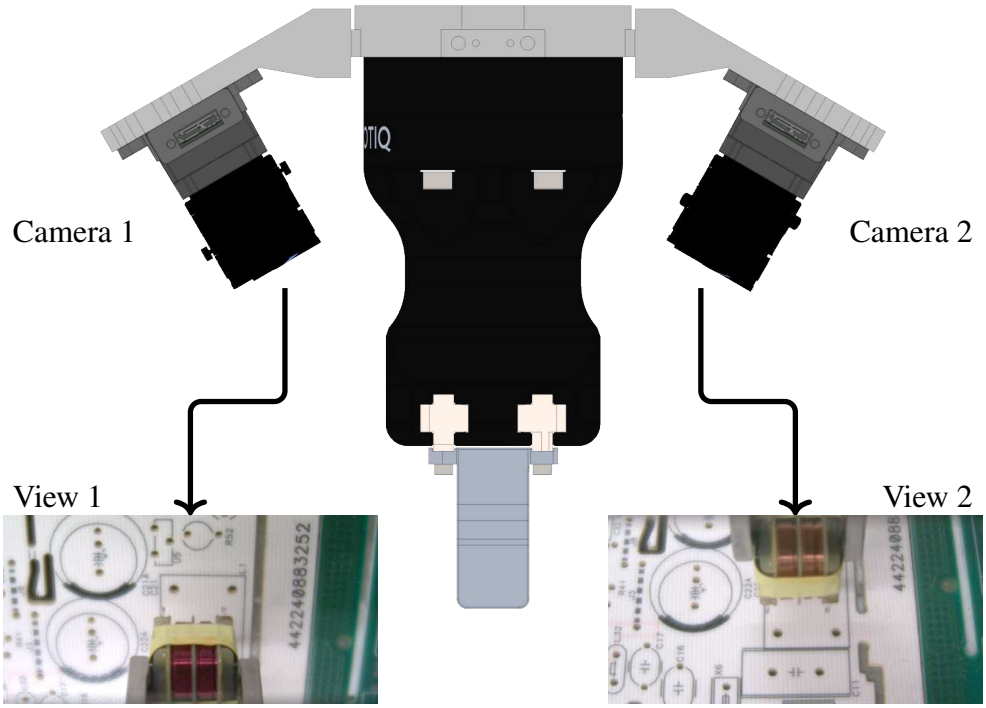


Figure 3.1: The conceptual visualization of the image acquisition process. The selection of the camera mounting angle at 30° was based on empirical trials. This setup allows for a detailed view of the manipulated object and the surrounding tools.

on the Z axis is set to $z_{init} = 2\,\mathrm{mm}$ above the surface of the workspace, the radius of the workspace is $R_{limit} = 15\,\mathrm{mm}$, and the rotation limits are $\psi_{limit} = 45°$.

In each episode, the agent's interactions with the environment are limited to 100 steps. In a single step, the agent computes a translation along the XYZ-axes and a rotation around the Z-axis in the tool reference frame $\{\mathbf{t}\}$, resulting in a 4-dimensional action space $\mathbf{a}_t = [\Delta x_t, \Delta y_t, \Delta z_t, \Delta\psi_t^z]$. It is important to note that the action space can be extended to include additional DOFs, such as rotations about the X- and Y-axes, to handle more complex tasks. Each variable in the action space is limited within a specific range $\{a^i_{min} \leq a^i_t \leq a^i_{max} : i \in \{1,2,3,4\}\}$, where a^i_{min} and a^i_{max} represent the minimum and maximum values for the i-th action variable. The limits for the action space variables are determined empirically based on the task requirements. For the electronic part insertion task, the translations range from $-2.0\,\mathrm{mm}$ to $2.0\,\mathrm{mm}$, and the rotations range from $-1.0°$ to $1.0°$. The policy output $\mathbf{a}_t$ is given in the reference frame of the tool $\{\mathbf{t}\}$, while the robot control system requires the target pose $\boldsymbol{x}_d$ in the reference frame of robot base $\{\mathbf{r}\}$. Furthermore, the RL agent operates in the reference frame of the workspace $\{\mathbf{w}\}$. The homogeneous transformation between the tool frame and the workspace frame is given by $\mathcal{T}_{\mathbf{wt}} = \mathcal{T}_{\mathbf{w}}^{-1}\mathcal{T}_{\mathbf{t}}$, and the transformation between the robot base frame and the workspace frame is given by $\mathcal{T}_{\mathbf{rw}} = \mathcal{T}_{\mathbf{r}}^{-1}\mathcal{T}_{\mathbf{w}}$. Therefore, to obtain the target pose $\boldsymbol{x}_d$ in the reference frame of the robot base, the output of the policy $\mathbf{a}_t$ is transformed by $\mathcal{T}_{\mathbf{rt}} = \mathcal{T}_{\mathbf{r}}^{-1}\mathcal{T}_{\mathbf{w}}^{-1}\mathcal{T}_{\mathbf{t}}$.

For each non-terminal step, the agent receives a reward given by the equation

$$r = -\tanh\left(\alpha \cdot d\right), \tag{3.1}$$

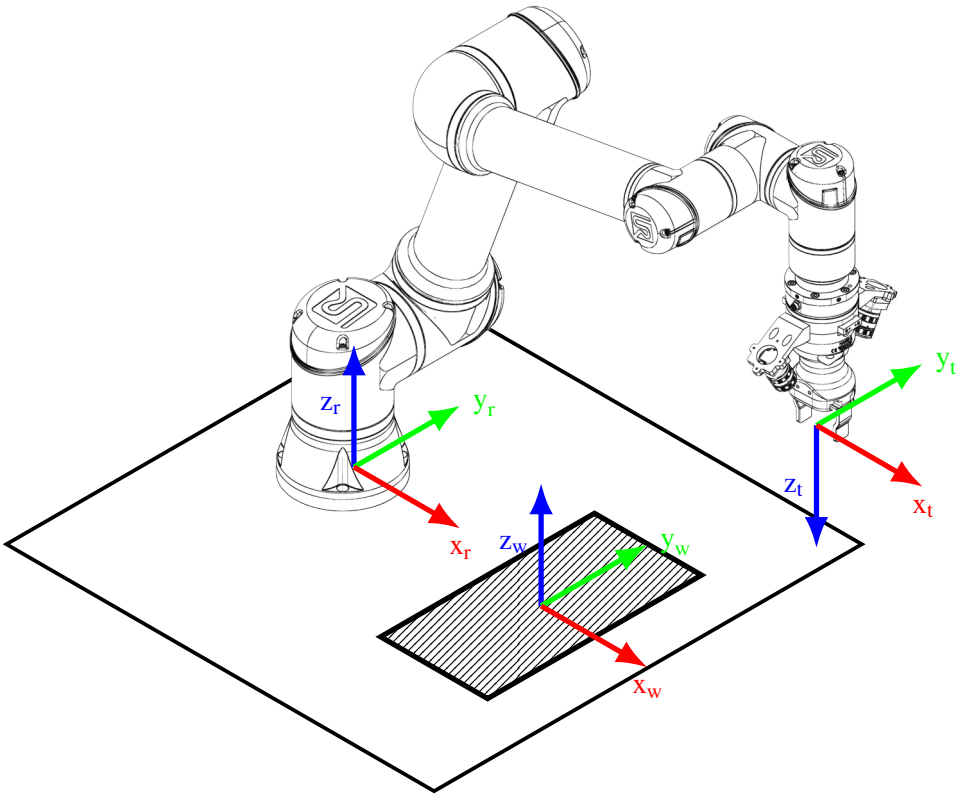


Figure 3.2: In the framework for robotic assembly tasks, the reference frames are designated as follows: The robot base frame is denoted as $\{\mathbf{r}\}$, the workspace frame is denoted as $\{\mathbf{w}\}$, and the tool frame is denoted as $\{\mathbf{t}\}$. The workspace frame is specifically defined as the target insertion pose of the currently manipulated object.

where d represents the ℓ_2-distance between the tool center point (TCP) and the insertion pose, and α is the reward sensitivity coefficient. The insertion pose used in the reward calculation is the target pose of the tool in the reference frame of the workspace. The parameter α influences the shape of the reward function, thereby affecting the agent's learning process. The task is considered complete when the Z-axis position of the tool in the workspace reference frame is less than or equal to 0.0 mm, indicating the successful insertion of the electronic part into the target location, and the agent receives a reward of $r = 10.0$ in this case.

An episode is terminated if the agent exceeds the time limit, leaves the workspace, or exceeds the rotation limits. If the episode is interrupted due to the agent leaving the workspace or surpassing the time limit, the agent receives the same reward as during non-terminal steps. However, if the episode is terminated due to exceeding the rotation limits, the agent receives a reward of $r = -2.0$ to ensure the safety of the attached cameras and cables. The design of the reward function aims to encourage the agent to complete the task efficiently and safely.

The discerning reader may observe that the reward function does not impose a penalty on the agent for damaging the inserted part. This deliberate design choice stems from the fact that if damage occurs during the insertion attempt, the agent will continuously receive negative rewards resulting from Equation (3.1). To illustrate, consider the task of inserting electronic components. A common form of damage is the bend of the leads, as depicted in Figure 3.3. If the RL agent bends the leads during the trial, the part becomes

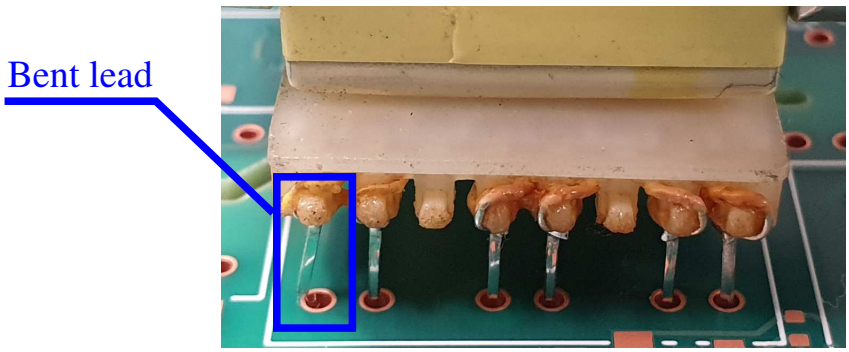


Figure 3.3: Close up of the damaged electronic part.

infeasible to insert, resulting in negative rewards for each attempt. This adverse feedback encourages the agent to learn more delicate insertion techniques.

3.2 Control Architecture

To facilitate effective interaction between the agent and the real-world robotic system, Bartyzel et al. [28] developed a control architecture that allows communication between the agent and the robot. The block diagram of the control architecture can be found in Figure 3.4. This architecture comprises two control loops: a high-level loop utilizing an off-policy RL algorithm as a control unit and a low-level loop employing the Cartesian admittance control system to directly regulate the robot. In this study, the off-policy RL algorithm adopted is SAC [31]. The RL agent transmits commands to the admittance controller at a frequency of $10\,\mathrm{Hz}$ and receives feedback information at the same frequency. The control signal is computed on the basis of multimodal observation. Feedback from the admittance controller includes state information such as the pose $\boldsymbol{x}$, twist $\dot{\boldsymbol{x}}$, and contact wrench $\mathcal{W}$ of the tool. Images are obtained from the cameras' drivers, which operate independently of the controller. The low-level control loop runs at a frequency of $500\,\mathrm{Hz}$ to ensure smooth motion of the industrial robot. The admittance controller receives the control signal representing the desired tool pose $\boldsymbol{x}_d$ and calculates the reference joint positions $\boldsymbol{q}_r$ using the IK solver. The resulting joint positions $\boldsymbol{q}_r$ are then sent to the robot controller to execute the desired motion.

The conventional training process of the reinforcement learning agent is a sequential operation involving interaction with the environment, receiving feedback, and updating the policy based on the collected data. This approach can be computationally expensive and time-consuming, particularly when training in real-world robotic systems. With a sequential training pipeline, it is challenging to achieve real-time control with a stable frequency of commands sent to the robot. To address this challenge, the Ape-X architecture [42] has been integrated with the SAC algorithm. This architecture enables distributed training of the RL agent, where multiple actors are spawned, each with its instance of the environment. The actors generate experiences and store them in the shared replay buffer. The learner samples mini-batches from this shared replay buffer and updates the network parameters. The actors' parameters are periodically synchronized with the latest learner's parameters. In this study, the Ape-X architecture is implemented with one spawned actor, since only one robot is used in the experiments. However, this solution can be easily scaled to a large number of robots, for example, in the case of a cooperative multi-robot assembly process.

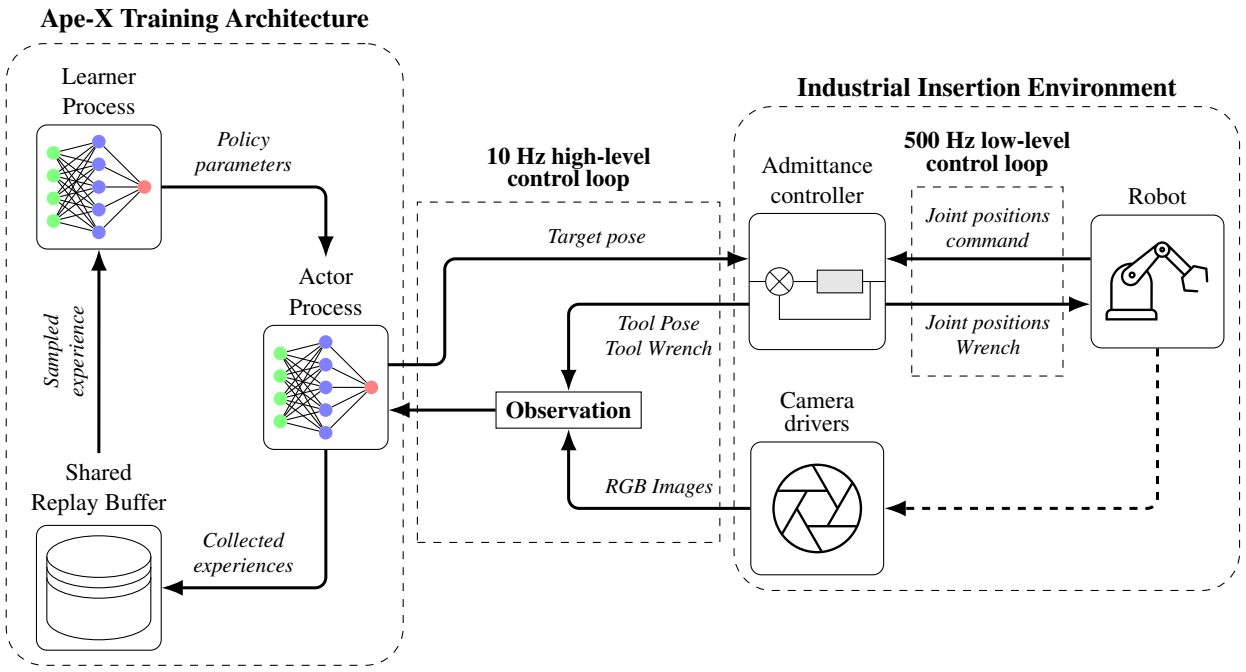


Figure 3.4: The block diagram of the control architecture for the industrial insertion task. **Left:** the Ape-X framework for asynchronous RL training. **Right:** the industrial insertion environment with the admittance controller and peripheral devices drivers. The high-level control loop involves the RL agent sending a target pose $\boldsymbol{x}_d$ to the admittance controller at a frequency of $10\,\mathrm{Hz}$. The controller then commands the joint positions $\boldsymbol{q}_r$ of the robot at a frequency of $500\,\mathrm{Hz}$. The RL agent receives feedback that includes the pose $\boldsymbol{x}$, twist $\dot{\boldsymbol{x}}$, and contact wrench components of the tool $\mathcal{W}$, and RGB images acquired from the cameras.

3.2.1 Robot-Agent Communication Software Stack

The control architecture is implemented using the ROS 2 middleware [74]. ROS 2 is an open source middleware that provides a comprehensive set of tools and libraries to develop robotic applications. Among its key features are real-time capabilities and support for distributed systems. In ROS 2, the software stack components are represented as nodes in a graph structure, with each node being responsible for a specific task. The nodes communicate with each other using the publish-subscribe mechanism or service calls. Data are transferred between nodes using the Data Distribution Service (DDS)[1] protocol, ensuring reliable and decentralized communication.

In the presented control architecture, each component of the system is implemented as a separate ROS 2 node. The nodes responsible for communication with the robot controller, the cameras, and the F/T sensor are implemented in C++, to ensure real-time and reliable communication. The RL agent with Ape-X architecture is implemented in Python with the RLlib [66] library. RLlib provides a ready-to-use engine for distributed training as well as the framework for implementing various reinforcement learning algorithms. RL agent models are implemented using the PyTorch library [87].

The RL agent interacts with the environment through the well-established interface for reinforcement learning tasks, known as Gymnasium [113] (formerly OpenAI Gym [10]). Gymnasium provides a stan-

[1] `https://www.omg.org/spec/DDS/`

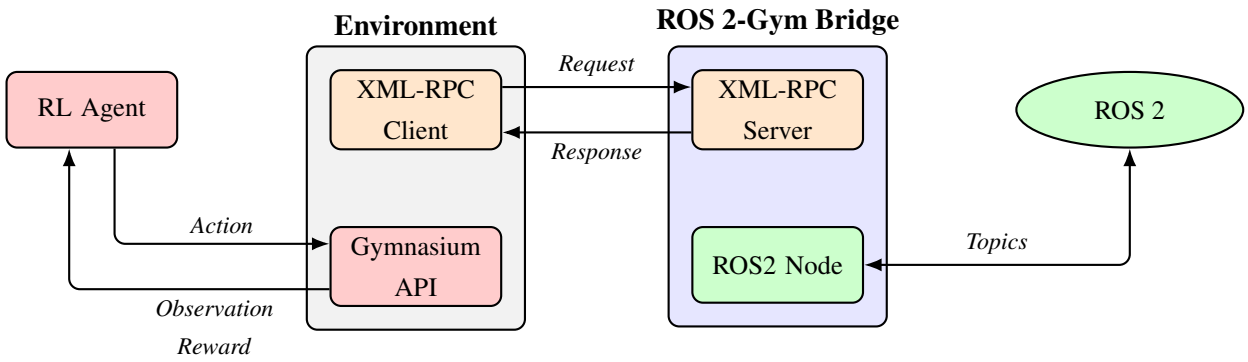


Figure 3.5: The diagram illustrating the communication flow between the RL agent and the robotic environment. The interaction between the *Gymnasium* environment and *ROS 2* nodes is performed via the *ROS 2-Gym Bridge* node, which facilitates the data exchange by exposing ROS 2 functionalities through the XML-RPC protocol.

dard API for defining reinforcement learning tasks, which includes the definition of the observation space, action space, and reward function. Implemented in Python, this library is convenient for use with frameworks like RLlib. The Gymnasium API allows one to control the environment via `env.reset()` and `env.step(action)` methods. The former resets the environment to the initial state and returns the initial observation, while the latter executes the action and transitions to the next state, returning the next observation, reward, and information about episode termination. Additionally, Gymnasium provides an additional functionality for applying wrappers over the environment, which can be used to preprocess observations, rewards, or actions before they are passed to the agent. Wrappers enable easy modification of the environment for broader experimentation without changing the core implementation, such as converting RGB images to grayscale or resizing images to a specific resolution.

In the software stack for training the RL agent, the Gymnasium environment is directly integrated into the training pipeline. Therefore, the transition to the next state is synchronized with the inference time of the RL policy. However, as mentioned at the beginning of this section, ROS 2 middleware is used to control the industrial robot and handle peripheral devices. This middleware relies heavily on the message-passing mechanism, which is asynchronous by design. Therefore, the node is synchronized with the communication ecosystem through the internal trigger mechanism. The messages are handled in the callback functions, which are triggered when the new message is received. The most common and efficient way to synchronize the nodes in ROS 2 is to use a call trigger mechanism within an endless loop. This ensures that the node is always ready to handle the most recent message. The asynchronous nature of the ROS 2 middleware contrasts with the synchronous nature of the Gymnasium environment. If the triggering mechanism in the Gymnasium environment were directly implemented, the RL agent would end up waiting for the next observation while the robot is still executing the previous action.

To address this issue, a bridge has been developed between the Gymnasium environment and the ROS 2 middleware. This bridge, called *ROS 2-Gym Bridge*, is implemented as a separate ROS 2 C++ node and exposes ROS 2 messages via the XML-RPC[2] protocol. The ROS 2 Gym Bridge combines ROS 2 node

[2] `https://xmlrpc.com/`

with the XML-RPC server, which is responsible for handling communication with the Gymnasium environment. To ensure compatibility with the reinforcement learning pipeline and facilitate implementation of the environment logic, this server offers the following methods:

- `execute_trajectory` - executes the given trajectory on the robot, such as moving the TCP to the element grasping or insertion pose. The trajectory is performed via the joint trajectory controller.
- `execute_action` - executes the action provided by the RL agent. The action is transformed to the robot base reference frame and then executed by the Cartesian admittance controller.
- `get_observation` - gathers time-synchronized sensory data from robot's sensors and returns it to the RL agent. The observation is composed of RGB images, the pose, the twist, and the wrench of the tool.
- `command_gripper` - commands the robot's gripper to open or close.

These methods are accessed within the Gymnasium environment through the XML-RPC client. The use of the XML-RPC protocol allows for seamless integration of ROS 2 middleware with the Gymnasium environment, ensuring that the RL agent is always in sync with the recent state of the robot and the sensors. The communication diagram for robot-agent communication is presented in Figure 3.5.

3.3 Laboratory Setup

In accordance with the control architecture described in Section 3.2, we established a laboratory setup to facilitate industrial assembly experiments. The illustration of the laboratory setup is depicted in Figure 3.6. The primary objective of this setup is to emulate the operational conditions of a real-world production line. The laboratory setup encompasses various devices and tools, including:

- **Universal Robots UR5e**[3] - a 6-DOF collaborative robot arm with a reachability of $850\,\mathrm{mm}$.
- **Robotiq Hand-E**[4] - a servoelectric gripper with $50\,\mathrm{mm}$ stroke and the maximum gripping force of $185\,\mathrm{N}$.
- **ATI Axia 80**[5] - a 6D F/T sensor with following specifications: $F_x = 200\,\mathrm{N}$, $F_y = 200\,\mathrm{N}$, $F_z = 360\,\mathrm{N}$, $\tau_x = 8\,\mathrm{N\,m}$, $\tau_y = 8\,\mathrm{N\,m}$, $\tau_z = 8\,\mathrm{N\,m}$.
- **Custom vision tool** - a tool equipped with two Basler Dart cameras[6] with a resolution of 1280×960 pixels. The visualization of this tool is presented on Figure 3.1
- **Workstation** - a computer with Ubuntu 22.04 LTS operating system, ROS 2 Humble Hawksbill middleware[7], and custom software for controlling the robot and handling the peripheral devices. The

[3] `https://www.universal-robots.com/products/ur5-robot/`
[4] `https://robotiq.com/products/hand-e-adaptive-robot-gripper`
[5] `https://www.ati-ia.com/products/ft/ft_models.aspx?id=Axia80-M20&campaign=axia80`
[6] `https://www.baslerweb.com/en/cameras/dart/`
[7] `https://docs.ros.org/en/humble/index.html`

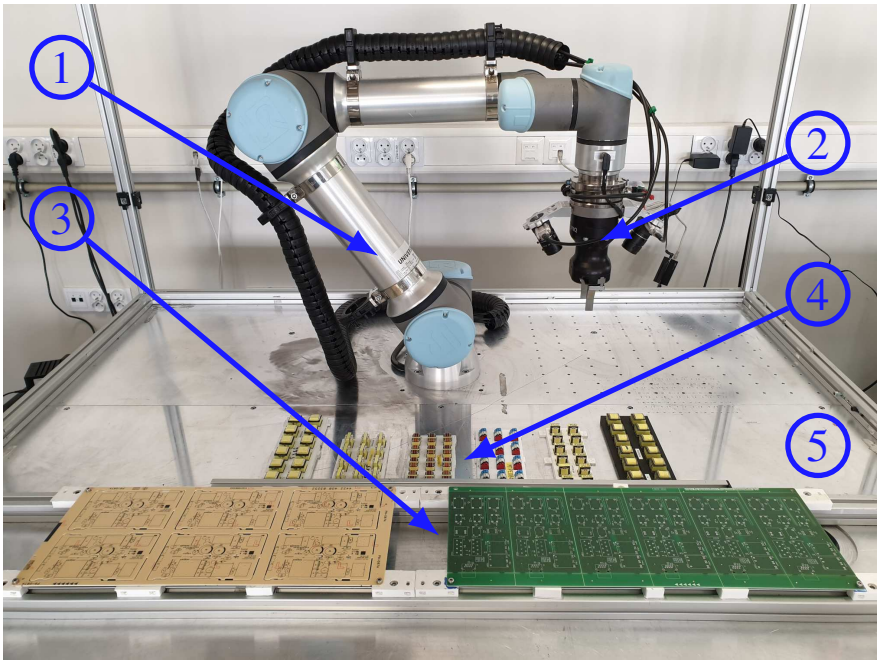


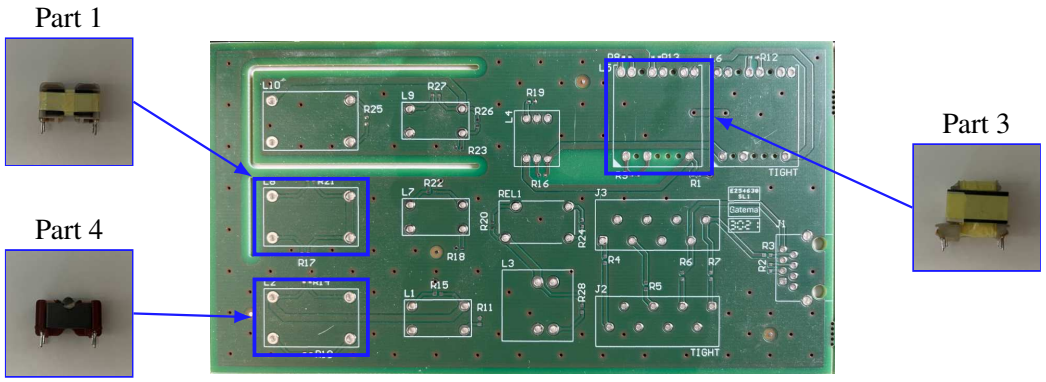
Figure 3.6: The laboratory setup used for the experiments that involve the control architecture. This setup consists of: 1 - industrial robot Universal Robots UR5e-series; 2 - tool with servoelectric gripper Robotiq Hand-e, F/T sensor ATI Axia80 and custom vision system with two Basler Dart cameras; 3 - workspace with various PCBs panels; 4 - trays with electronic componenets; 5 - workstation with Ubuntu 22.04 LTS, ROS 2 Humble Hawksbill and custom software.

computer is equipped with an AMD Threadripper 1950X CPU, 64 GB of RAM, and an NVIDIA GeForce RTX 4070 GPU[8].

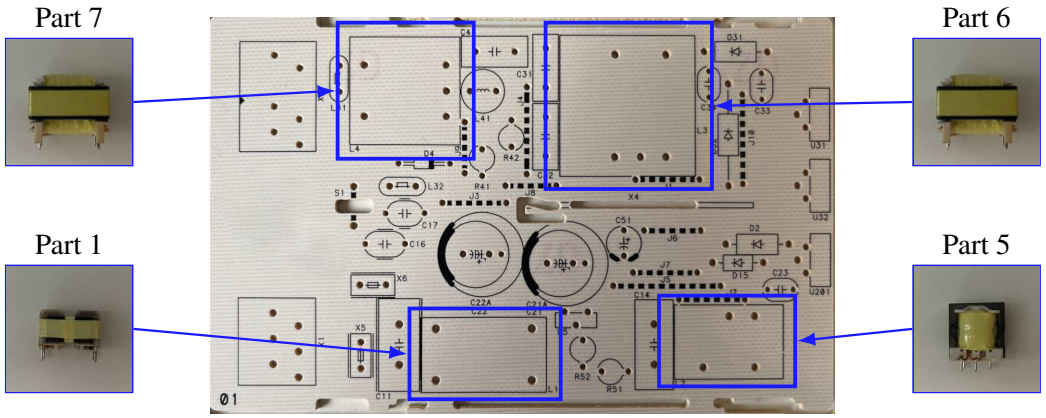
In the experimental setup, the robot's workspace is equipped with various PCB panels and electronic components to closely mimic real-world assembly line conditions. In production lines, PCBs are delivered to robotic manufacturing cells in panel form, each panel containing multiple PCBs. The use of panels aims to streamline the handling of PCBs and optimize the production process, such as defining the constant belt conveyor width and standardizing the clamping mechanism. Within the test bed, electronic parts are placed on 3D-printed trays specifically designed for the experiments. Each electronic part has a dedicated tray to ensure consistent and reliable experimentation. These trays are positioned on the workstation's surface near the robot's workspace, with the electronic parts arranged in a predetermined orientation known to the environment software stack.

In the laboratory setup, to simulate real-world conditions in a flexible assembly line, the robot's workspace contains various PCB panels and electronic components. The specific PCBs and electronic parts used in the experiments are illustrated in Figure 3.7. The electronic parts selected for the experiments are transformers chosen for their non-standardized physical appearance (such as geometrical shape), requiring manual insertion by human workers. These electronic parts vary in shape, appearance, number of leads, and lead arrangements, providing a wide range of components for evaluating the proposed framework. Through-

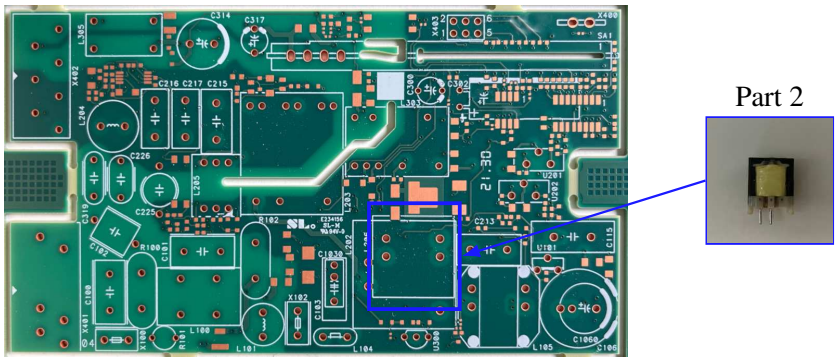
[8] https://www.nvidia.com/pl-pl/geforce/graphics-cards/40-series/rtx-4070-family/



(a) PCB Layout 1



(b) PCB Layout 2



(c) PCB Layout 3

Figure 3.7: The presentation of the PCB variants and the electronic parts that were used in the experiments. For each electronic part, the corresponding insertion place is highlighted. Electronic parts differ in shape, color, and number of leads.

out this work, electronic parts are referred to as *Electronic Part 1*, through *Electronic Part 7*, or shorter *Part 1* to *Part 7*. Additionally, the PCBs in the workspace differ in design layout, laminate color, and hole clearance, which can have a significant impact on the insertion process.

3.3.1 Environment reset procedure

In the context of training an RL agent within a real-world environment, the reset procedure is of significant importance to ensure the reproducibility of experiments and the safe operation of the robotic system during the training process. Specifically, in the context of industrial assembly tasks presented in this dissertation, it is assumed that electronic parts are picked anew from the trays for each training episode, and further, the subsequent PCB from the panel is chosen, thus affecting the target pose of the insertion denoted $\boldsymbol{x}_{target} \in \mathcal{P}_{target}$.

To achieve this desired behavior, the environment reset procedure is initiated by selecting a grasping pose $\boldsymbol{x}_{grasp} \in \mathcal{P}_{grasp}$, which is drawn from a predefined set $\mathcal{P}_{grasp}$ corresponding to the various poses in the trays. The size of $\mathcal{P}_{grasp}$ depends on the design of the tray, which can accommodate, for example, up to 12 electronic parts in 3D-printed trays and potentially hundreds of parts in trays provided by the manufacturer. Following the selection of the grasping pose, the robot executes a grasp action to pick up the object located in the specified pose $\boldsymbol{x}_{grasp}$. Subsequently, the target pose $\boldsymbol{x}_{target}$ is chosen from a set of target poses denoted as $\mathcal{P}_{target}$. The size of $\mathcal{P}_{target}$ aligns with the number of PCBs present in a single panel.

To introduce randomness during the training process, the initial pose $\boldsymbol{x}_{init}$ is perturbed by sampling the disturbance values from a uniform distribution, denoted as $\mathcal{U}(-l^{xy}, l^{xy})$ for the XY-plane and $\mathcal{U}(-l^{\psi}, l^{\psi})$ for rotation around the Z-axis, where l^{xy} and l^{ψ} represent the range values for disturbances in the XY-plane and rotation around the Z-axis respectively. Consequently, the initial pose $\boldsymbol{x}_{init}$ is obtained from the transformation $\mathcal{T}_{init} = \mathcal{T}_{target}^{-1}\mathcal{T}_{\Delta}$. The disturbance transformation matrix $\mathcal{T}_{\Delta}$ is computed based on the sampled disturbance values and has the form:

$$\mathcal{T}_{\Delta} = \begin{bmatrix} \cos(\Delta\psi) & -\sin(\Delta\psi) & 0 & \Delta x \\ \sin(\Delta\psi) & \cos(\Delta\psi) & 0 & \Delta y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Having established the initial pose $\boldsymbol{x}_{init}$, the robot proceeds to move to this pose and perform the insertion test. Afterward, upon completion of the trial, the manipulated object is returned to its grasp pose $\boldsymbol{x}_{grasp}$, thus preparing the environment for the next episode. The environmental reset procedure is outlined in Algorithm 6.

3.3.2 Admittance control system

In the context of the control architecture, the admittance control system plays a critical role by ensuring the robot's compliant interaction with the environment, particularly in the manipulation of fragile objects such as electronic components. Section 2.2.3 provided an overview of the general admittance control law. However, in the framework designed for robotic industrial assembly tasks, the admittance control system is specifically implemented as a position-based Cartesian admittance controller. Consequently, the control law can be expressed as follows:

Algorithm 6 Environment reset procedure

Require:

Set of electronic parts poses $\mathcal{P}_{grasp}$ in the trays

Set of target insertion poses $\mathcal{P}_{target}$ placed on the PCBs

Initial pose disturbance range values l^{xy} and l^{ψ}

1: Get the grasp pose $\boldsymbol{x}_{grasp}$ from $\mathcal{P}_{parts}$

2: Get the target pose $\boldsymbol{x}_{target}$ from $\mathcal{P}_{target}$

3: Perform the grasp action to pick up the object $\mathbf{p}_{grasp}$

4: Sample the disturbance from $[\Delta x, \Delta y] \sim \mathcal{U}(-l^{xy}, l^{xy})$ and $\Delta\psi \sim \mathcal{U}(-l^{\psi}, l^{\psi})$

5: Compute the disturbance transformation $\mathcal{T}_{\Delta}$ from sampled disturbance values

6: Compute initial pose $\boldsymbol{x}_{init}$ from $\mathcal{T}_{init} = \mathcal{T}_{target}^{-1}\mathcal{T}_{\Delta}$

7: Move the robot to the initial pose $\boldsymbol{x}_{init}$

8: Perform insertion trial ▷ RL agent interacts with the environment

9: Place back manipulated object to previous pose $\boldsymbol{x}_{grasp}$

$$\ddot{\boldsymbol{x}}_r = \mathbf{M}^{-1}\left(\mathcal{W} - \mathbf{D}\dot{\boldsymbol{x}} - \mathbf{K}\left(\boldsymbol{x} - \boldsymbol{x}_d\right)\right), \tag{3.3}$$

The control signal $\ddot{\boldsymbol{x}}_r$ denotes the intended acceleration of the end-effector of the robot in the reference frame of the robot base. However, since this signal cannot be sent directly to the robot, it is necessary to integrate the desired acceleration to acquire the desired pose $\boldsymbol{x}_r$. Subsequently, this target pose is transformed into joint space utilizing the IK solver. Within the implemented control system, the computation of the coefficients d_{ii} of the damping matrix $\mathbf{D}$ is governed by the following expression:

$$d_{ii} = 2\zeta_{ii}\sqrt{k_{ii}m_{ii}}, \tag{3.4}$$

where ζ_{ii} signifies the damping ratio, k_{ii} denotes the coefficient of the stiffness matrix $\mathbf{K}$, and m_{ii} represents the coefficient of the inertia matrix $\mathbf{M}$. Index i corresponds to the degrees of freedom of the robot. The parameters of the admittance control system chosen for the developed framework are outlined in Table 3.1. In order to ensure smooth motion of the robot's end-effector and to mitigate excessive oscillations, the wrench measurement is subjected to filtering through a low-pass filter with a cutoff frequency of $25\,\mathrm{Hz}$.

To verify the performance of the admittance control system, the robot was commanded to perform a series of Z-axis end-effector movements. The outcomes of the robot's response to the admittance control law are detailed in Figure 3.8. Initially, the robot executed a free-space movement, resulting in the end-

Table 3.1: Admittance control system parameters

Parameter	Value	Unit
$\mathbf{K}$	$\mathrm{diag}([1000, 1000, 1000, 20, 20, 20])$	N/m, Nm/rad
ζ	$\mathrm{diag}([2.8, 2.8, 2.8, 2.8, 2.8, 2.8])$	-
$\mathbf{M}$	$\mathrm{diag}([3.0, 3.0, 3.0, 0.04, 0.04, 0.04])$	kg, kgm^2

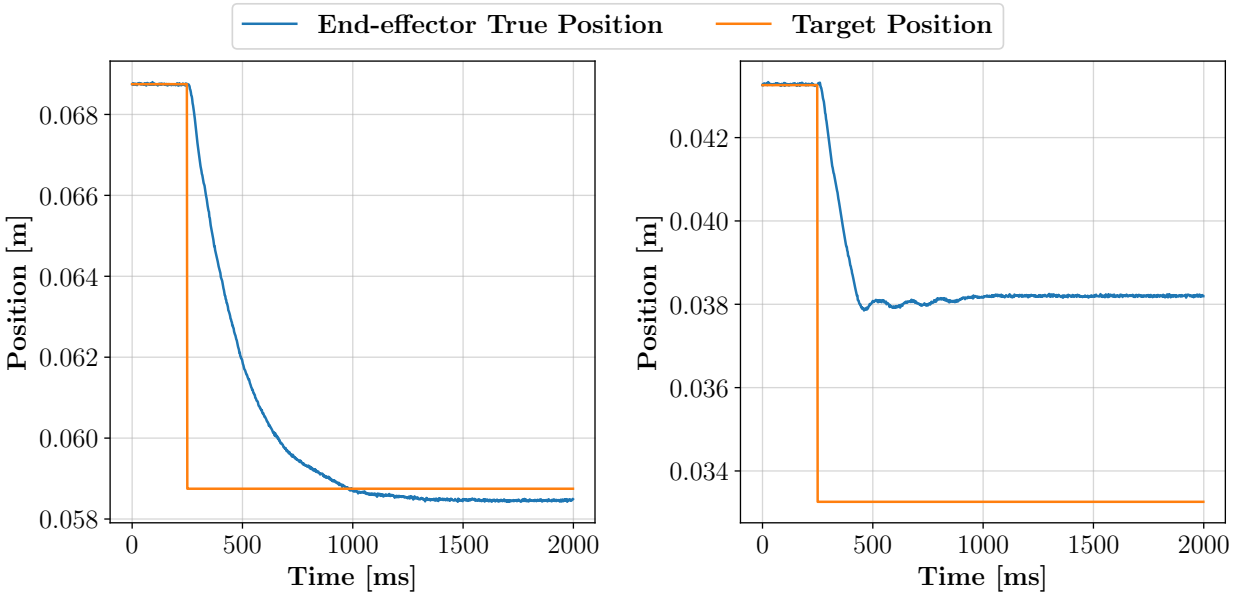
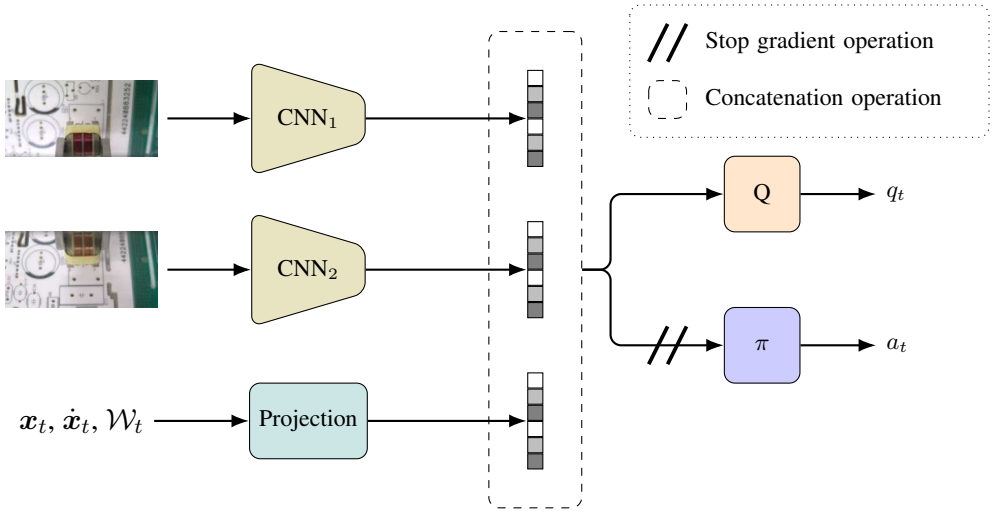


Figure 3.8: Visualization of admittance controller logs obtained from the robotic system during various z-axis end-effector movements. The orange line illustrates the desired position, and the blue line represents the actual position. **Left:** The command corresponded to a free-space movement, resulting in close adherence to the target position. **Right:** The robot wad commanded to make contact with a rigid surface, causing the actual position to be limited by the admittance control law.

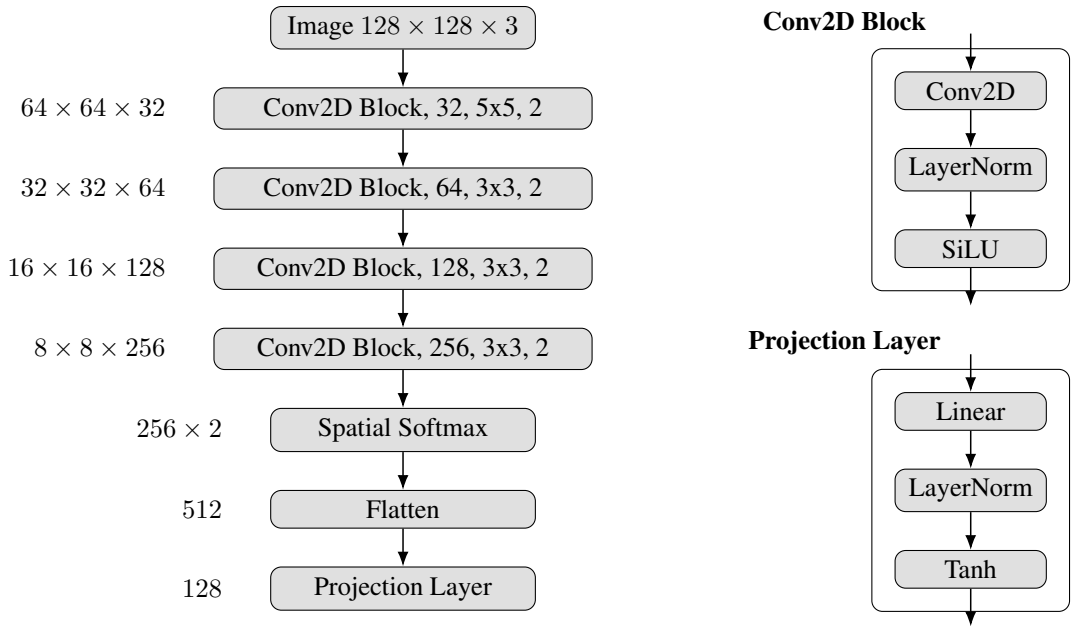
effector closely approaching the target position. Subsequently, the robot was commanded to make contact with a rigid surface, causing the controller to limit the actual position. The acquired results substantiate the effectiveness of the formulated admittance controller in facilitating a compliant interaction between the robot and its surroundings. This attribute is crucial to the successful completion of robotic assembly operations, particularly when handling fragile items, such as electronic parts.

3.4 Model Architecture

The SAC algorithm was chosen as the RL agent in the framework for robotic industrial assembly tasks. As outlined in Section 3.1, the agent is provided with multimodal observations, including two camera images and state information such as the pose, twist, and contact wrench of the tool. The original SAC implementation was designed only for state-based observations. Therefore, the model architecture was modified to accommodate the multimodal observation space. Visual modalities are processed by independent CNNs, while state information is handled by the projection layer. The projection layer, which consists of a dense layer followed by LayerNorm [5] and a tanh activation function, is responsible for normalizing the input data and projecting them into a specific dimension. The state information is projected into a 64-dimensional embedding. The CNN used to process the visual modality comprises four convolutional layers with filter sizes of 32, 64, 128, and 256, followed by a spatial softmax layer [64]. The CNN's features are then normalized via the projection layer, resulting in a 128-dimensional embedding. Each convolutional layer is followed by LayerNorm with the SiLU activation function [90]. The pixel values of the input images are divided by 255



(a) Block diagram of the model architecture for the SAC agent.



(b) CNN architecture for the visual modalities and overview of the projection layer.

Figure 3.9: Visualization of the model architecture used in the experiments.

and scaled to the range of $[-0.5, 0.5]$. The obtained embeddings are concatenated and then directly passed to the actor π_ϕ and critic Q_θ networks. Both function approximators are implemented as feedforward neural networks with two hidden layers of 512 units each. ReLU is used as an activation function for every layer. Since the SAC's policy is stochastic, the actor-network outputs the mean and log standard deviation of a Gaussian distribution. To ensure the stability of the training process, the logarithm of the standard deviation of the policy distribution is limited to the range of $\log \sigma = [-11.5, 1.6]$. The action sampled from the Gaussian distribution is then clipped via the tanh function to ensure that the action is within the valid range. The complete model architecture is illustrated in Figure 3.9.

The CNNs utilized for processing visual inputs are shared by both the actor and the critic networks. This approach significantly reduces the model's parameters and training time, making it suitable for real-

world robotic applications. However, as demonstrated by Yarats et al. [131], sharing convolutional layers between actor and critic networks may lead to training instability, as the gradient computed from policy loss introduces noise during the optimization phase. To address this, the authors proposed restricting the gradient flow from the policy network to the convolutional layers. This optimization approach was implemented in the presented study.

3.5 Experiments and Results

The proposed framework for robotic industrial assembly tasks was evaluated through a series of experiments conducted in a laboratory setup (Figure 3.6). The primary objective of these experiments was to evaluate the performance of the RL agent in executing the insertion of electronic parts into PCBs. The evaluation process involved a comparison of the agent's performance with conventional methods, such as random search, and an investigation into the impact of visual observations on the agent's performance. In this section, the results presented are solely for Electronic Part 1 and PCB Layout 1 (see Figure 3.7). Additional results for the remaining electronic parts are provided in Section B.1.

3.5.1 Training Procedure

Throughout the training phase, the RL agent aims to learn the optimal policy for placing electronic components on PCBs. The training process consists of a total of 50000 steps, with each step representing a single interaction between the agent and the environment. The framework combines the Ape-X architecture with the SAC algorithm, allowing asynchronous data collection during the training process. In this setup, the actor sends a rollout of 10 transitions to the shared replay buffer, while the learner process synchronizes the latest parameters of the policy network with the actor every 10 steps. Similarly to the original Ape-X work [42], the transitions are stored in the prioritized experience replay (PER) buffer [93]. The prioritized replay buffer is a variation of the replay buffer that assigns a priority to each transition based on the TD-error. Transitions with higher priority are sampled more frequently, improving the learning process. The learner process starts by updating the policy network after the replay buffer contains at least 400 samples. This configuration ensures that the RL agent can be trained from scratch in approximately 2.5 hours for multimodal observations and 1.5 hours for state-based observations.

To improve the agent's exploration capabilities and its robustness to disturbances, the disturbance values for the initial pose were set to $l^{xy} = 2\,\mathrm{mm}$ and $l^{\psi} = 3°$. Furthermore, the initial Z-axis position of the tool z_{init} was gradually increased from $2\,\mathrm{mm}$ to $30\,\mathrm{mm}$ after every 100 consecutive successful insertions to optimize the training process and collect diverse data. This approach, known as curriculum learning, is designed to facilitate the agent's learning process by gradually increasing the task's complexity.

The more detailed hyperparameters of the SAC algorithm used for the experiments can be found in Chapter A. The policy and critic networks, as well as the temperature coefficient, were optimized using the Adam optimizer [53] with a learning rate of 0.0003. The target entropy was set to $-\frac{1}{2}\dim(\mathcal{A})$, instead of $-\dim(\mathcal{A})$ as suggested in the original SAC paper.

3.5.2 Evaluation Metrics

In the manufacturing industry, the efficiency of the assembly process assumes paramount importance as it directly influences the production line throughput and overall manufacturing costs. Key metrics traditionally used to assess the performance of assembly applications include the success rate and the mean time to complete a given task. These fundamental yet robust metrics offer valuable insight into the performance of the method and the efficacy of the assembly process. The success rate is defined as the proportion of successful insertions relative to the total number of attempts, while the mean time to complete the task denotes the average duration taken by the method to successfully insert the manipulated object into the target location. In a real-world production setting, these metrics are computed over specific time frames, such as a single shift or a full day of operation. Nonetheless, in the context of the laboratory experiments presented in this dissertation, the metrics are computed over 100 trials, as this quantity of trials yields a reliable estimation of the method's performance.

3.5.3 Evaluation Procedure

The solutions implemented in industrial robotic assembly cells must not only achieve high-performance metrics but also demonstrate robustness to disturbances and uncertainties. Within these cells, three primary sources of disturbance can be identifiable:

- **Objects feeding system** - This particular disturbance is associated with the accuracy of the application, which defines the grasping position of the manipulated object. Typically, object detection systems are used to establish the grasping position. The precision of the object detection system is highly dependent on the quality of the cameras, the lighting conditions, and the algorithms utilized.

- **End-effector design** - The design of the end-effector plays a crucial role in ensuring precise insertion. For example, using grippers designed to specific object's shapes helps ensure high grasp precision. However, this approach limits the robot's rapid adaptation to different tasks and increases the costs of the robotic setup. In contrast, the use of universal grippers can reduce costs and improve the flexibility of the robotic setup, but it may also introduce uncertainty in the grasping position.

- **Assembled object positioning mechanism** - In manufacturing lines, assembled items are commonly placed on conveyor belts or on trays. The accuracy of the object positioning mechanism can greatly affect the precision of the insertion. For example, in the electronics industry, PCB panels are transported to robotic workstations via conveyor belts. The precision of the clamping mechanism can cause misalignment between the programmed target insertion position and the actual object position.

In the experiments presented, the first two sources of disturbances are directly modeled by the laboratory setup design and the environment reset procedure. The gripper utilized in the experiments comprises two universal fingers designed to grasp a wide range of objects. Although the laboratory setup presented in Figure 3.6 does not incorporate an object detection system, the design of the trays, in combination with the environment reset procedure, introduces uncertainty in the grasping pose. This intentional feature in the

Table 3.2: Test cases for evaluating the agent's performance under initial pose disturbances.

	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
l^{xy}	0 mm	5 mm	10 mm	0 mm	0 mm	5 mm	10 mm
l^{ψ}	0°	0°	0°	7°	15°	7°	15°

experimental setup aims to replicate real-world conditions. The third source of disturbances is modeled by applying the disturbance values to the initial pose during the environment-reset procedure.

The evaluation process was developed to assess the performance of the agent in the presence of the third source of disturbance. The test scenario comprises 7 test cases, each case corresponding to a specific disturbance value. These cases are detailed in Table 3.2 and cover a broad spectrum of disturbances that can occur in a real-world robotic assembly environment. Notably, test cases 3 and 7 are the most demanding, introducing significant disturbances in the XY plane and rotation around the Z-axis. Although disturbances of this magnitude are unlikely to be encountered in a real-world robotic assembly environment, they serve as good indicators of the agent's robustness to disturbances. The performance of the agent during the evaluation is evaluated based on the metrics outlined in Section 3.5.2.

3.5.4 Impact of Visual Observations

When considering the use of RL agents for continuous control tasks, it is common to have a state-based observation space, which may include the robot's joint angles, end-effector pose, wrench measurements, and so on. In order to effectively train an RL agent in such an environment, the task needs to be formulated as an MDP, where the current state encompasses all the information necessary for decision-making. However, in real-world robotic applications, the state-based observation space can suffer from disturbances and uncertainties, making it difficult for the agent to learn the optimal policy. Real-world robotic tasks, such as robotic assembly, can be viewed as POMDP, where the agent lacks access to full state information. This limitation can be mitigated by providing the agent with additional observations, such as visual data.

In the following section, an investigation is conducted on the impact of visual observations on the performance of the RL agent. The SAC algorithm was used to train the agent with and without visual observations. The state-based observation space was characterized by the concatenation of the tool pose, twist, and wrench data. Due to the substantial influence of the seed on the training process, the agent was trained with five different seeds for each observation space. The outcomes of the training are depicted in Figure 3.10.

The duration of training for the agent using visual observations was approximately 2.5 hours while training with state-based observations alone took approximately 2 hours. The SAC model with combined observations converged in approximately 10000 steps, equivalent to 25 minutes of training. On the contrary, most iterations with state-based observations commenced converging after 40000 steps. However, the training curves indicate that achieving full convergence would require more than 50000 steps. This observation suggests that an agent relying on state-based observations requires more data to mitigate the disturbances and uncertainties introduced in real-world robotic applications.

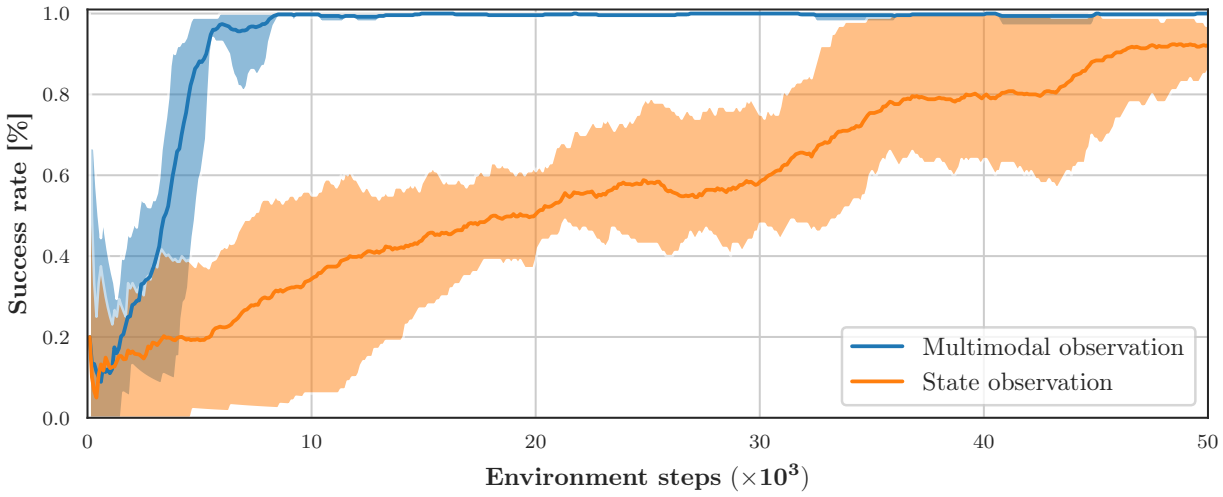


Figure 3.10: Training results for the SAC algorithm with and without visual observations. For each method, five runs of different random seeds were performed. The shaded area represents the minimum and maximum values, while the bold line represents the mean. The results show that the use of visual observations leads to a significant improvement in the learning performance.

After the agents completed their training, they were assessed using the test scenario outlined in Section 3.5.3. The evaluation results are detailed in Figure 3.11. Both agents demonstrated comparable success rates, although the agent with visual feedback in the observation space achieved a slightly higher success rate. However, the most notable disparity between the agents was observed in the average time taken to complete the task. The agent with combined visual feedback with the state-based information notably outperformed the state-based agent in terms of time efficiency. Specifically, as disturbance values increased, the state-based agent required more time to complete the task, while the visually trained agent consistently maintained a mean completion time of 1 s to 3 s. This suggests that visual feedback significantly enhances the agent's performance in the presence of disturbances and uncertainties, as the CNN is able to learn and extract pertinent features from the images relevant to the task.

3.5.5 Performance Against Baselines

In the manufacturing industry, most robotic cells designed for assembly tasks rely on traditional methods, which are typically rule-based and use predefined trajectories or scripts. They often employ a hybrid force/position control system (Section 2.2.2) to execute the specified trajectory. Although these methods are easy to implement and require minimal computational resources, they lack the flexibility and adaptability necessary to handle disturbances and uncertainties in real-world robotic assembly tasks. In this section, the performance of the method introduced in this dissertation is compared with the following baselines:

Straight down - The robot moves vertically along the Z-axis until it detects the signal to stop, like successful insertion. Displacements on the XY plane are set to zero. The force controller maintains a constant contact force of 2 N to control the vertical displacement.

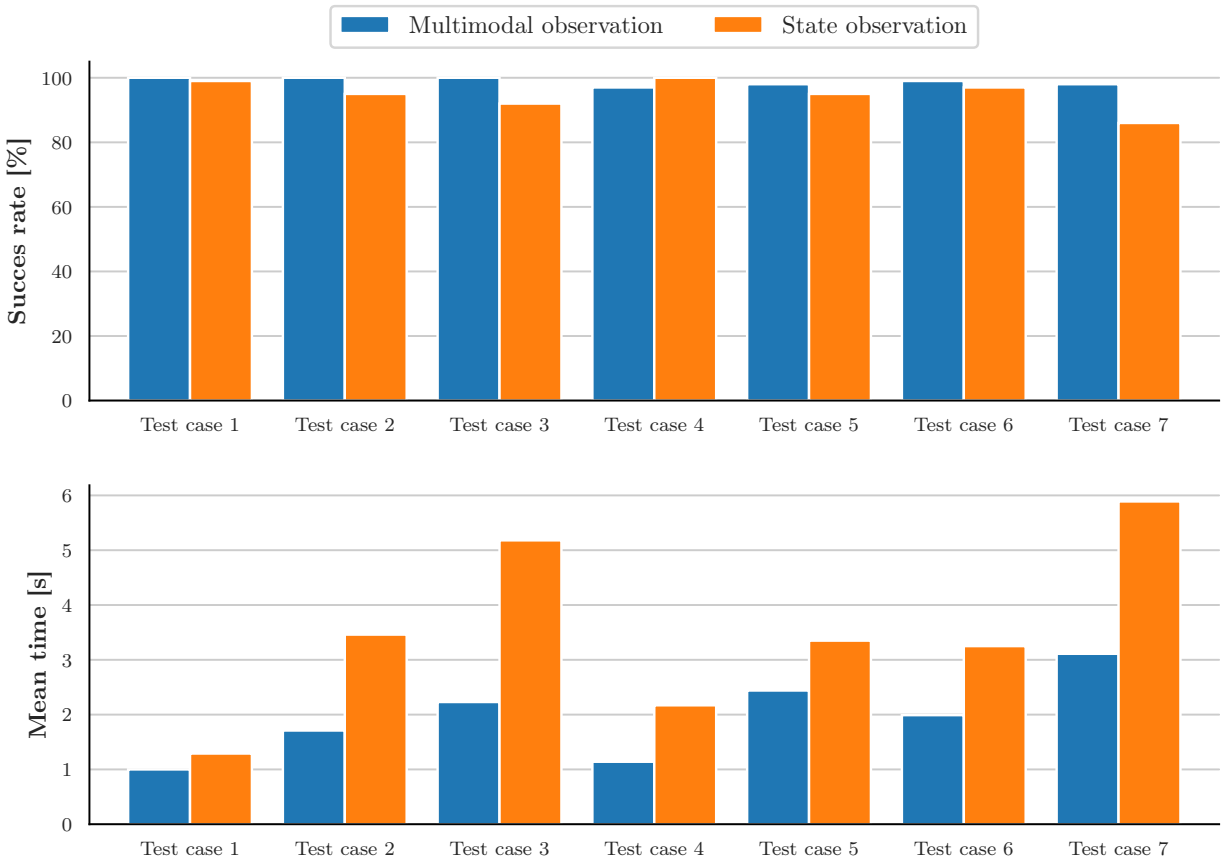


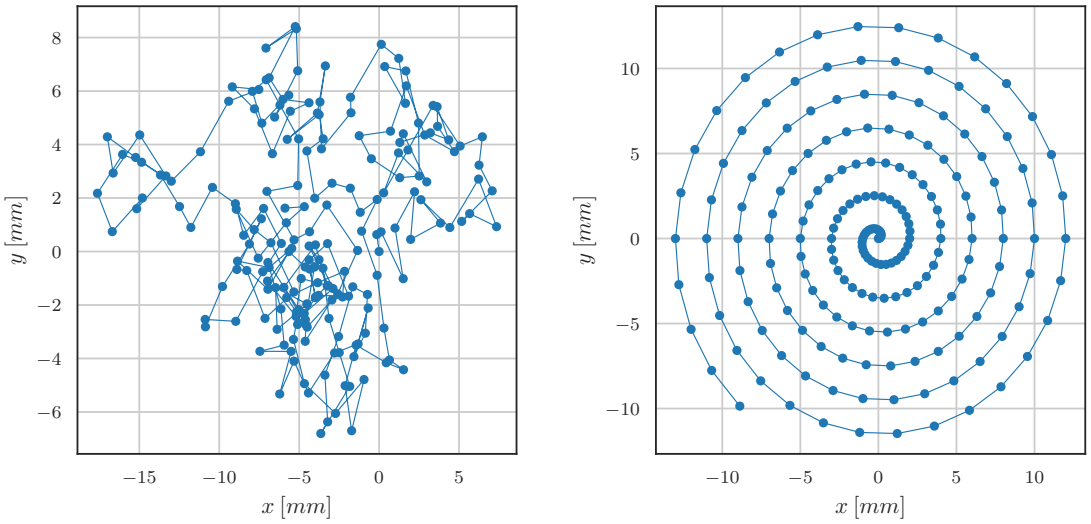
Figure 3.11: Evaluation results for the SAC algorithm with and without visual observations. The results are shown for the electronic part type 1. The advantage of using visual observations is clearly visible in context of the mean time to complete the task. The success rate is also slightly higher when using visual observations, but the difference is not as significant as in the case of the mean time. **Top:** Success rate over 100 trials, higher is better. **Bottom:** Mean time to complete the task over 100 trials, lower is better.

Random search - Stochastic search strategy, described in [80], where the robot moves randomly in the XY plane until it detects a successful insertion or receives a signal to terminate the trial. Displacements along these axes are sampled from a uniform or normal distribution at each time step. This strategy can be defined as follows:

$$\begin{aligned} x_{t+1} &= x_t + k_x \Delta x, \quad \Delta x \sim \mathcal{U}(-1.0, 1.0) \\ y_{t+1} &= y_t + k_y \Delta y, \quad \Delta y \sim \mathcal{U}(-1.0, 1.0) \end{aligned} \tag{3.5}$$

where x_t and y_t represent the current position of the robot in the XY plane, and k_x and k_y denote the step sizes along the X and Y axes, respectively. The displacement in the Z-axis is controlled by the force controller, maintaining a constant contact force of 2 N. The sample trajectory of the random search strategy is illustrated in Figure 3.12a.

Spiral search - The robot executes a predefined spiral trajectory [80] on the XY plane until it detects a successful insertion or receives a signal to terminate the trial. The trajectory can be mathematically



(a) Random search

(b) Spiral search

Figure 3.12: Conventional strategies for object insertion task. Trajectories were generated by 200 time steps.

expressed as:

$$\begin{aligned} x_{t+1} &= x_t + r_t \cos(\theta_t) \\ y_{t+1} &= y_t + r_t \sin(\theta_t) \end{aligned} \tag{3.6}$$

where $\theta_t = n\Delta\theta$, $r_t = \Delta r\theta_t$, n represents the current step, $\Delta\theta$ is the angle step, and Δr is the radius step. The force controller maintains a constant contact force of 2 N on the Z-axis. The spiral search strategy's sample trajectory is depicted in Figure 3.12b.

In this section, the results for the Electronic Part 1 and PCB Layout 1 (see Figure 3.7) are presented. Additional results for the remaining electronic parts are provided in Section B.1. The methods were evaluated based on the test scenario outlined in Section 3.5.3. The results, depicted in Figure 3.13, highlight the significant outperformance of the RL agent compared to the baselines in terms of success rate and mean time to complete the task. The RL agent achieved a success rate of 100% or close to 100% in all test cases, demonstrating its robustness to disturbances and uncertainties.

In contrast, baselines exhibited a notable decrease in performance as disturbance values increased. In particular, the straight-down method, which executes only vertical movements, failed to solve the task when disturbances were applied in the XY plane. Furthermore, even the random search strategy and the spiral search strategy, while more flexible than the straight-down method, were unable to achieve success rates comparable to those of the RL agent. The poor performance of these strategies can be explained by analyzing the generated trajectories, as illustrated in Figure 3.12.

The random search strategy samples consecutive displacements in the XY plane at each time step from a uniform distribution, theoretically allowing it to find the target position with more allowable steps. Similarly, the spiral search strategy, with a significantly low radius step Δr and angle step $\Delta\theta$, should be able to cover the entire XY plane around the target position. However, in both cases, these configurations lead to an increased mean time to complete the task, as the robot executes many unnecessary movements before

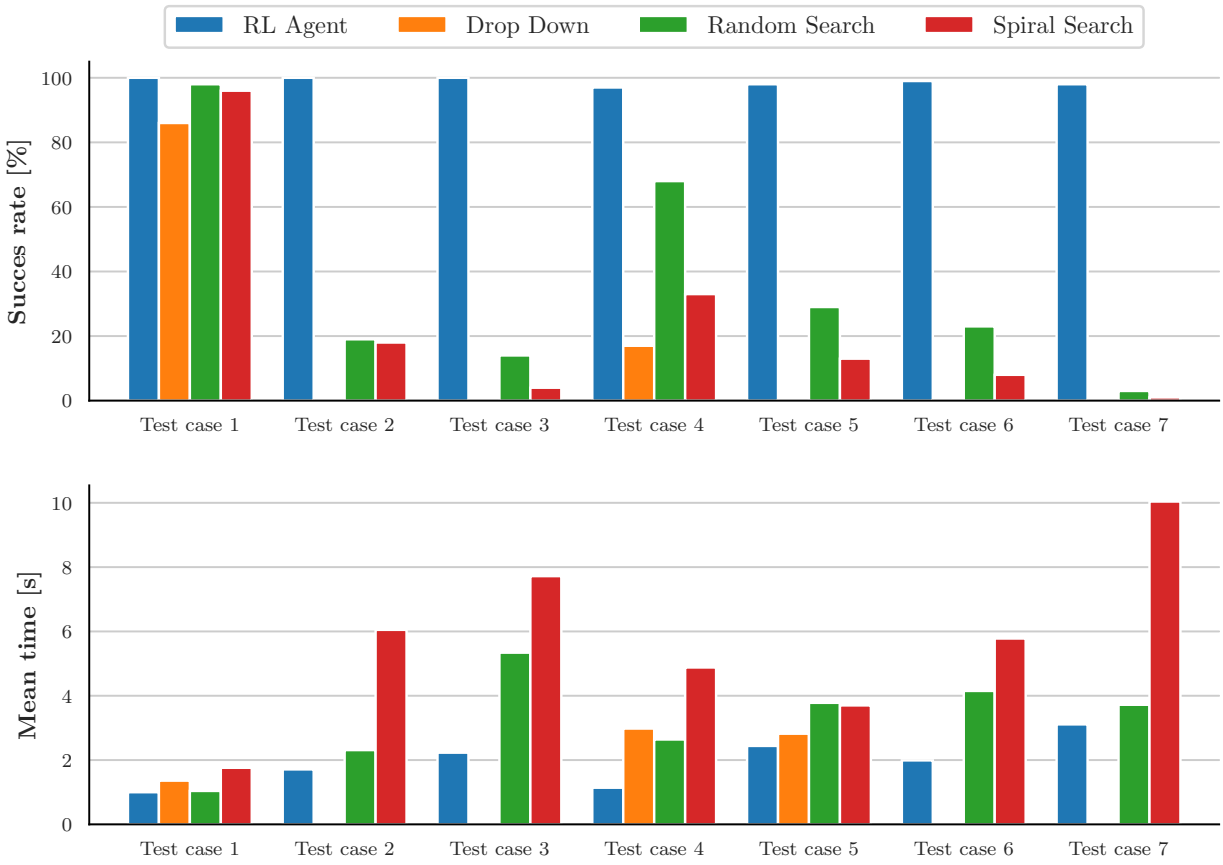


Figure 3.13: Comparison of the RL agent and the conventional methods against the initial pose disturbances. The results are shown for the Electronic Part 1. It is clear that the RL agent outperforms the conventional methods in terms of success rate and mean time to complete the task. **Top:** Success rate over 100 trials, higher is better. **Bottom:** Mean time to complete the task over 100 trials, lower is better.

reaching the goal. In contrast, the RL agent learned the optimal policy for the task, resulting in a high success rate and a low mean time to complete the task.

In conclusion, the findings suggest that the RL agent is capable of replacing conventional methods in robotic assembly tasks, offering a more flexible and adaptable solution.

3.5.6 Qualitative Analysis

The preceding sections have mainly focused on quantitatively evaluating the performance of the RL agent in robotic assembly tasks. Metrics such as success rate and mean time to task completion have provided valuable insights into the agent's efficiency. However, to gain a deeper understanding of the agent's decision-making process and the strategies used, a qualitative analysis of its behavior is necessary. In this section, a qualitative analysis of the RL agent's behavior will be conducted, with a specific focus on its trajectories and wrench measurements collected during the insertion process.

In order to provide a comprehensive overview of the agent's behavior, the agent completed 100 insertion trials for Electronic Part 1 and PCB Layout 1. In this experimental setup, the initial Z-axis position (z_{init}) was set at $40\,\mathrm{mm}$ above the target insertion position, while the initial XY position was perturbed by the

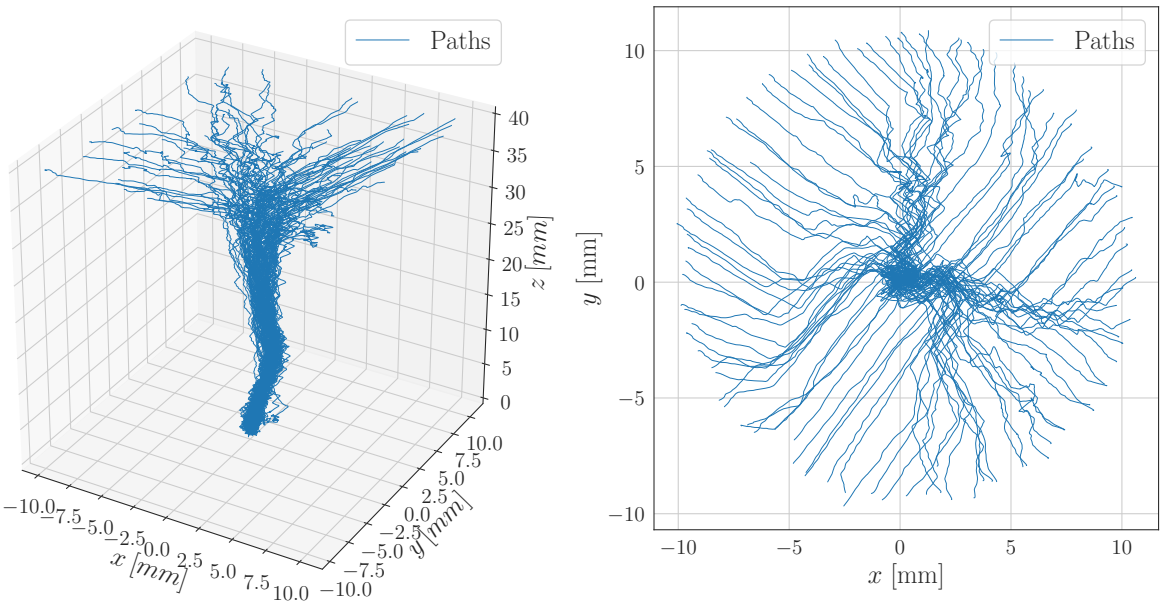


Figure 3.14: Visualization of the trajectories produced by the RL agent over 100 insertion attempts. The insertion task started 40 mm above the target position to showcase the efficacy of the proposed method. The paths illustrate successful insertion attempts. **Left:** 3D view of the trajectories. **Right:** 2D view of the trajectories.

disturbance values specified in test case 7. During the insertion process, the agent's trajectories and wrench measurements were recorded at a sampling rate of 250 Hz.

The trajectories of the agent during the insertion process are depicted in Figure 3.14. The subplot on the left displays the trajectories in a 3D space, whereas the subplot on the right shows a top-down view of the trajectories in the XY plane. It is evident from the visualization that the agent initially moves toward the target insertion position in the XY plane before descending vertically along the Z-axis. This approach demonstrates the efficiency of the RL agent's decision-making process, as it moves directly toward the target position without unnecessary movements, in contrast to conventional methods.

The raw wrench measurements obtained during the insertion process were not directly interpretable due to variations in trajectory lengths. To facilitate analysis, the data were truncated when the robot was within 10 mm of the target insertion position, as the robot's movement in further distance registers minimal contact force. Subsequently, the wrench measurements were binned using a statistical method. The outcomes of this post-processing are shown in Figure 3.15. The wrench measurements provide valuable insights into the interaction between the robot's end-effector and the environment during the insertion process. In particular, as the robot approaches the target insertion position, the contact force in the Z-axis increases, signifying contact between the tool and the surface. Furthermore, the lateral forces in the XY plane deviate from 0 N, indicating the robot's positional adjustments to align with the target insertion position. Ultimately, the wrench measurements stabilize as the robot completes the insertion process. The analysis of the wrench measurements leads to the conclusion that the RL agent combined with the admittance control system effectively manages the interaction between the robot's end-effector and the environment during insertion.

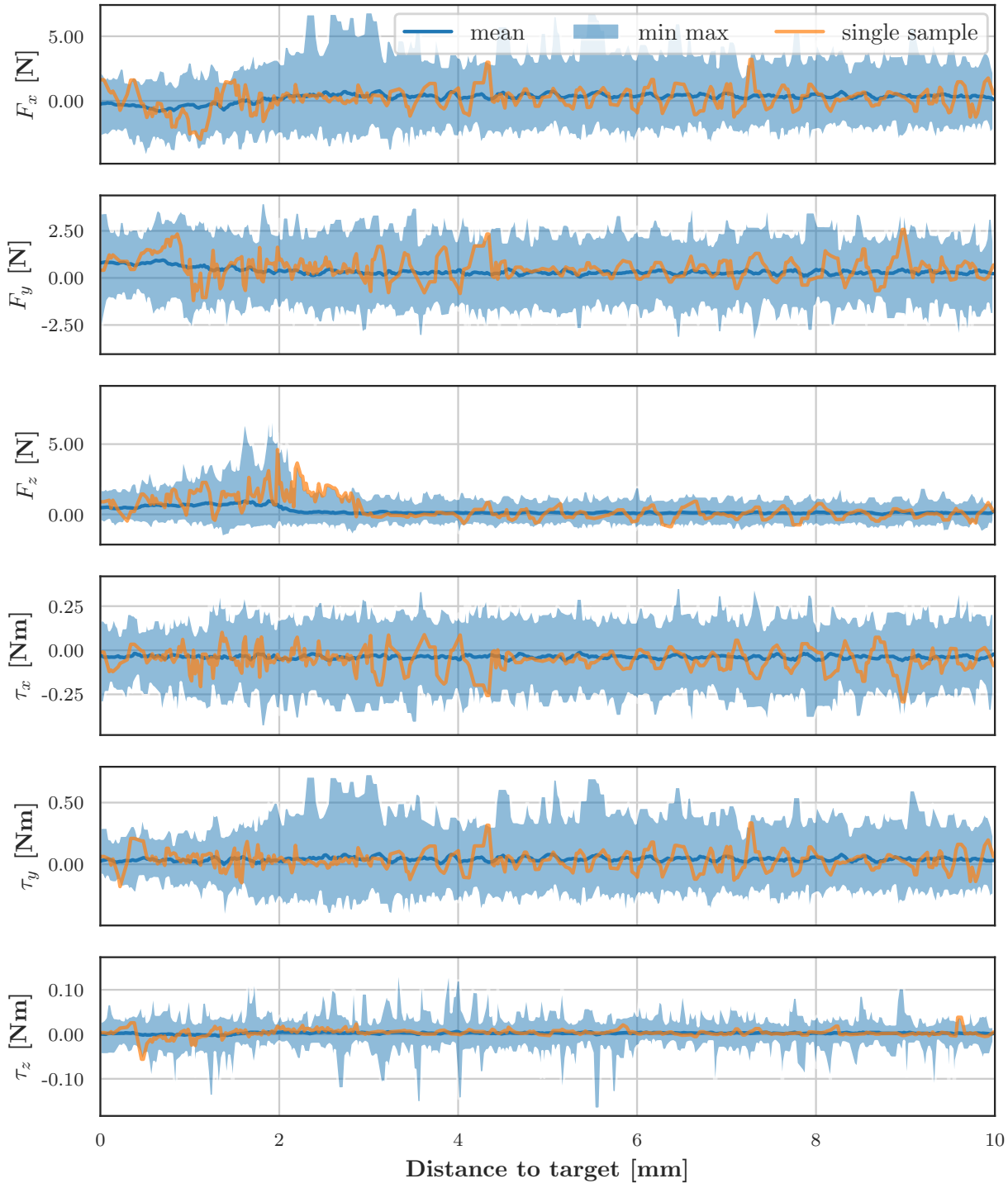


Figure 3.15: Band plot of the wrench measurements obtained from 100 insertion attempts. The insertion trial started 40 mm above the target position to ensure consistent data for subsequent analysis. The collected data was truncated when the robot reached a 10 mm distance from the target position, as beyond this point, the robot typically executed free-motion without environmental contact. The plots were generated by grouping the data through statistical data binning, with subsequent calculation of mean, minimum, and maximum values for each bin. Additionally, individual insertion attempt measurements were included to enhance data visualization.

Chapter 4

Multimodal Variational DeepMDP

The previous section of this thesis has focused primarily on introducing a framework for industrial insertion tasks, a crucial component of flexible manufacturing systems. The experimental results presented thus far have demonstrated that the RL-based method outperforms conventional methods like random search in terms of performance and robustness to position perturbations. As a result, the proposed framework shows promise in automating the insertion task in flexible production lines. However, due to the nature of flexible manufacturing systems, the RL-based method still faces limitations in generalizing to new tasks or environments, as generalization remains a challenging problem in reinforcement learning [20, 134, 25, 103]. Training an RL agent from scratch for each new task or environment is impractical in real-world applications due to the significant time and resources it requires. Therefore, this chapter introduces a novel approach to improve the transferability of RL agents within the environment of flexible production lines. The concept and some preliminary results of this chapter have been published in a journal article [6]. The chapter begins with an extensive introduction to the proposed approach, providing a detailed explanation of the model architecture and its integration with the off-policy algorithm. Subsequently, a series of experiments are conducted to assess the performance of the proposed method in terms of transferability to new tasks or environments. The results of the experiments are presented and thoroughly discussed. Finally, the chapter concludes with a discussion of the potential impact of the proposed method in the field of flexible manufacturing systems and outlines future research directions.

4.1 Methodology

The successful deployment of RL agents in real-world applications is highly dependent on transferability and robustness. The previous chapter demonstrated the ability of the proposed framework for industrial insertion to learn a policy that effectively inserts objects into a target position despite various disturbances. However, the generalizability of the RL agent to new tasks or environments remains a significant challenge. This dissertation tries to address this limitation by introducing a novel approach called *Multimodal Variational DeepMDP*. This approach combines the benefits of variational inference of multimodal data and the DeepMDP framework [13]. MVDeepMDP is designed to learn a joint multimodal latent representation of observed modalities and predict transitions and rewards in the latent space, similar to the DeepMDP algo-

rithm. The joint latent representation is computed using the generalized Product-of-Experts [12] mechanism, which balances the confidence of each expert based on the input modalities. This approach can seamlessly integrate with any off-policy algorithms, allowing the RL agent to learn a policy that can be quickly transferred to the assembly of new product types or operated in a new environment. Furthermore, the proposed method is built on the framework detailed in Chapter 3; thus, the SAC algorithm is used as the core component.

4.1.1 Mixing Multimodal Observation

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ represent a collection of M modalities observed by the agent. Each modality $\mathbf{x}_i$ may belong to different types, such as images, wrench measurements, or point clouds. In scenarios like this, it is common to preprocess each modality separately using a corresponding encoder f_{enc}^i and then concatenate the extracted features into a single representation vector, which is subsequently directly fed into the policy and value networks. However, this method does not capture the correlation between the modalities in the representation. Discovering the interaction between the modalities is crucial for learning a task-relevant feature representation. This challenge is often addressed in the modern deep learning literature by employing an attention mechanism [117], which has proven effective in various multimodal tasks [88, 83, 11, 138]. Nonetheless, the attention mechanism is computationally expensive and requires a large amount of data to learn the meaningful interaction between the modalities. On the other hand, an interesting alternative can be found in the domain of variational inference. Existing approaches [125, 63] utilize the Product-of-Experts mechanism to compute a joint multimodal latent representation. In this approach, the modalities are equally weighted, and the joint latent representation is computed as the product of the experts' distributions. The formula for computing the PoE is as follows:

$$P(\mathbf{z}) = \frac{1}{Z} \prod_{i=0}^{M} p\left(\mathbf{z}\right) \tag{4.1}$$

In a study by Hinton et al. [40], it was pointed out that training a model using the Product PoE approach to maximize likelihood is challenging due to the renormalization term Z. However, assuming a special case of Gaussian experts, where $p_i\left(\mathbf{z}|\mathbf{x}_i\right) = \mathcal{N}\left(\mu_i(\mathbf{x}_i)|\Sigma_i(\mathbf{x}_i)\right)$, the output distribution remains Gaussian. The mean and variance of this distribution are given by the following equations:

$$\mu_{PoE}(\mathbf{z}) = \left(\sum_i \mu_i(\mathbf{z}) T_i(\mathbf{z})\right) \left(\sum_i T_i(\mathbf{z})\right)^{-1} \tag{4.2}$$

$$\Sigma_{PoE}(\mathbf{z}) = \left(\sum_i T_i(\mathbf{z})\right)^{-1} \tag{4.3}$$

where $T_i(\mathbf{z}) = \Sigma_i(\mathbf{z})^{-1}$ is the precision of the i-th Gaussian expert in $\mathbf{z}$. However, the PoE approach tends to result in the output distribution being influenced more by highly confident experts than less confident ones, potentially affecting the model's ability to generalize and discover interactions between modalities. This limitation can be overcome using the gPoE, which introduces a weighted mechanism to balance the

confidence of each expert. The formula for computing the gPoE is as follows:

$$P(\mathbf{z}) = \frac{1}{Z} \prod_i p^{\lambda_i(\mathbf{z})}(\mathbf{z}) \tag{4.4}$$

Assuming Gaussian experts once again, the mean and variance of the gPoE distribution can be defined as follows:

$$\mu_{gPoE}(\mathbf{z}) = \left(\sum_i \mu_i(\mathbf{z}) \lambda_i(\mathbf{z}) T_i(\mathbf{z}) \right) \left(\sum_i \lambda_i(\mathbf{z}) T_i(\mathbf{z}) \right)^{-1} \tag{4.5}$$

$$\Sigma_{gPoE}(\mathbf{z}) = \left(\sum_i \lambda_i(\mathbf{z}) T_i(\mathbf{z}) \right)^{-1} \tag{4.6}$$

The weights λ_i can be computed in various ways. The most common approach is to consider them as a change in entropy from prior to posterior at point $\mathbf{x}$, denoted as $\Delta H(\mathbf{x}) = H(p(\mathbf{z}|\mathbf{x})) - H(p(\mathbf{z}))$. This approach requires almost no additional computational cost. Using the entropy change to weigh experts is suitable for supervised learning tasks, where the entropy change can be directly derived from the data. However, in the reinforcement learning domain, computing the change in entropy requires using the previous latent representation $\mathbf{z}_{t-1}$ during the inference of the current latent representation $\mathbf{z}_t$. Additionally, training such a model requires storing the previous observations and recomputing the latent representation at each training step, which is computationally and memory intensive.

In the proposed method, the weights λ_i are parameterized by the function $f_{gPoE}(\mathbf{E})$ approximated by a neural network, where $\mathbf{E}$ is a set of output embeddings $\{\mathbf{e}_1, \ldots, \mathbf{e}_M\}$. A single output embedding $\mathbf{e}_i$ is obtained from the feature extractor g_{ϕ_i} of the i-th modality. The part of the encoders that extracts features is shared with the parameterization function $f_{gPoE}(\mathbf{E})$. It is important to note that e_i is the output of the encoder g_{ϕ_i}, not the stochastic latent representation $\mathbf{z}_i$. To avoid the "veto" problem, where the expert with the highest confidence $\lambda_i \to \infty$ dominates the output, the obtained weights are normalized using the softmax function to distribute importance across available modalities, ensuring that for each latent dimension $\sum_i \lambda_i = 1$.

Having introduced the gPoE, we can formulate a joint latent representation as:

$$\begin{aligned} \mathbf{z}_{joint} &\sim \mathcal{N}\left(\mu_{gPoE}(\mathbf{z}_1, \ldots, \mathbf{z}_M), \Sigma_{gPoE}(\mathbf{z}_1, \ldots, \mathbf{z}_M)\right) \\ &\sim \mathcal{N}\left(\mu_{gPoE}(\mathbf{Z}), \Sigma_{gPoE}(\mathbf{Z})\right) \end{aligned} \tag{4.7}$$

or, more generally:

$$\begin{aligned} \mathbf{z}_{joint} &\sim q(\mathbf{z}_{joint}|\mathbf{x}_1, \ldots, \mathbf{x}_M) \\ &\sim q(\mathbf{z}_{joint}|\mathbf{X}) \end{aligned} \tag{4.8}$$

where $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$ is a set of latent representations of the observed modalities.

4.1.2 Integration with Reinforcement Learning Agent

In the MVDeepMDP framework, the joint latent representation $\mathbf{z}_t^{joint}$ serves as an input observation for both the actor and the critic networks. As a result, the policy $\pi_\theta\left(\mathbf{a}_t \middle| \mathbf{z}_t^{joint}\right)$ and the Q-value functions $Q_\theta\left(\mathbf{z}_t^{joint}, \mathbf{a}_t\right)$ are conditioned on the multimodal latent representation $\mathbf{z}_t^{joint}$. The actor and critic networks

are parameterized by the functions π_θ and Q_θ, respectively. Given that the joint latent representation is considered an input observation, the SAC algorithm can be directly applied to the MVDeepMDP framework. Consequently, the objectives from Section 2.1.6 take on the following form:

Q-value objective:

$$\begin{aligned} \mathcal{J}_Q(\theta) &= \mathop{\mathbb{E}}_{\substack{\mathbf{z}_t^{joint}\sim q_\phi(\cdot|\mathbf{X_t}),\\ \mathbf{z}_{t+1}^{joint}\sim q_\phi(\cdot|\mathbf{X_{t+1}}),\\ \mathbf{a}_t\sim\mathcal{D}}} \left[\frac{1}{2}\left(Q_\theta\left(\mathbf{z}_t^{joint},\mathbf{a}_t\right)-\left(r_t-\gamma V_{\bar{\theta}}\left(\mathbf{z}_{t+1}^{joint}\right)\right)\right)^2\right] \\ V_{\bar{\theta}}(\mathbf{z}_{t+1}^{joint}) &= \mathop{\mathbb{E}}_{\substack{\mathbf{z}_{t+1}^{joint}\sim q_\phi(\cdot|\mathbf{X_{t+1}}),\\ \mathbf{a}_{t+1}\sim\pi_\theta\left(\cdot\middle|\mathbf{z}_{t+1}^{joint}\right)}} \left[\min_{i=1,2} Q_{\bar{\theta}_i}\left(\mathbf{z}_{t+1}^{joint},\mathbf{a}_{t+1}\right)-\alpha\log\pi_\theta\left(\mathbf{a}_{t+1}\middle|\mathbf{z}_{t+1}^{joint}\right)\right] \end{aligned} \tag{4.9}$$

Policy objective:

$$\mathcal{J}_\pi(\theta) = \mathop{\mathbb{E}}_{\mathbf{z}_t^{joint}\sim q_\phi(\cdot|\mathbf{X_t}),}\left[\alpha\log\pi_\theta\left(\mathbf{a}_t\middle|\mathbf{z}_t^{joint}\right)-Q_\theta\left(\mathbf{z}_t^{joint},\mathbf{a}_t\right)\right] \tag{4.10}$$

Entropy objective:

$$\mathcal{J}_\alpha(\alpha) = \mathop{\mathbb{E}}_{\mathbf{z}_t^{joint}\sim q_\phi(\cdot|\mathbf{X_t}),}\left[\alpha\log\pi_\theta\left(\mathbf{a}_t\middle|\mathbf{z}_t^{joint}\right)-\alpha\bar{\mathcal{H}}\right] \tag{4.11}$$

Similarly to the training procedure defined in Section 3.4, the MVDeepMDP encoder components are shared between the actor and critic networks.

4.1.3 Model Architecture

The MVDeepMDP method combines the learning of multimodal dynamic latent representations with the off-policy actor-critic algorithm. An overview of the MVDeepMDP architecture is illustrated in Figure 4.1. In this framework, each modality is preprocessed by an independent stochastic encoder [55] that computes the mean and standard deviation of a Gaussian distribution, from which a 128-dimensional latent representation is sampled. Depending on the type of observation modality, the encoder is either a fully connected network with two dense layers of size 512 or a CNN as described in Section 3.4. The gPoE weight estimator is parameterized by a neural network with two dense layers of 1024. In this architecture, the reward prediction neural network is shared across all modalities, while the transition models are independent for each modality. The transition models and the reward decoder follow the design of the encoder models for state-based modalities. To obtain the joint dynamic latent representation, the gPoE mechanism is also applied to the dynamic latent representations of the individual modalities. However, the gPoE estimator's parameters are not shared with the encoder networks to ensure the independence of the latent representations. LayerNorm [5] followed by the SiLU activation function [90] is applied to all neural networks in MVDeepMDP.

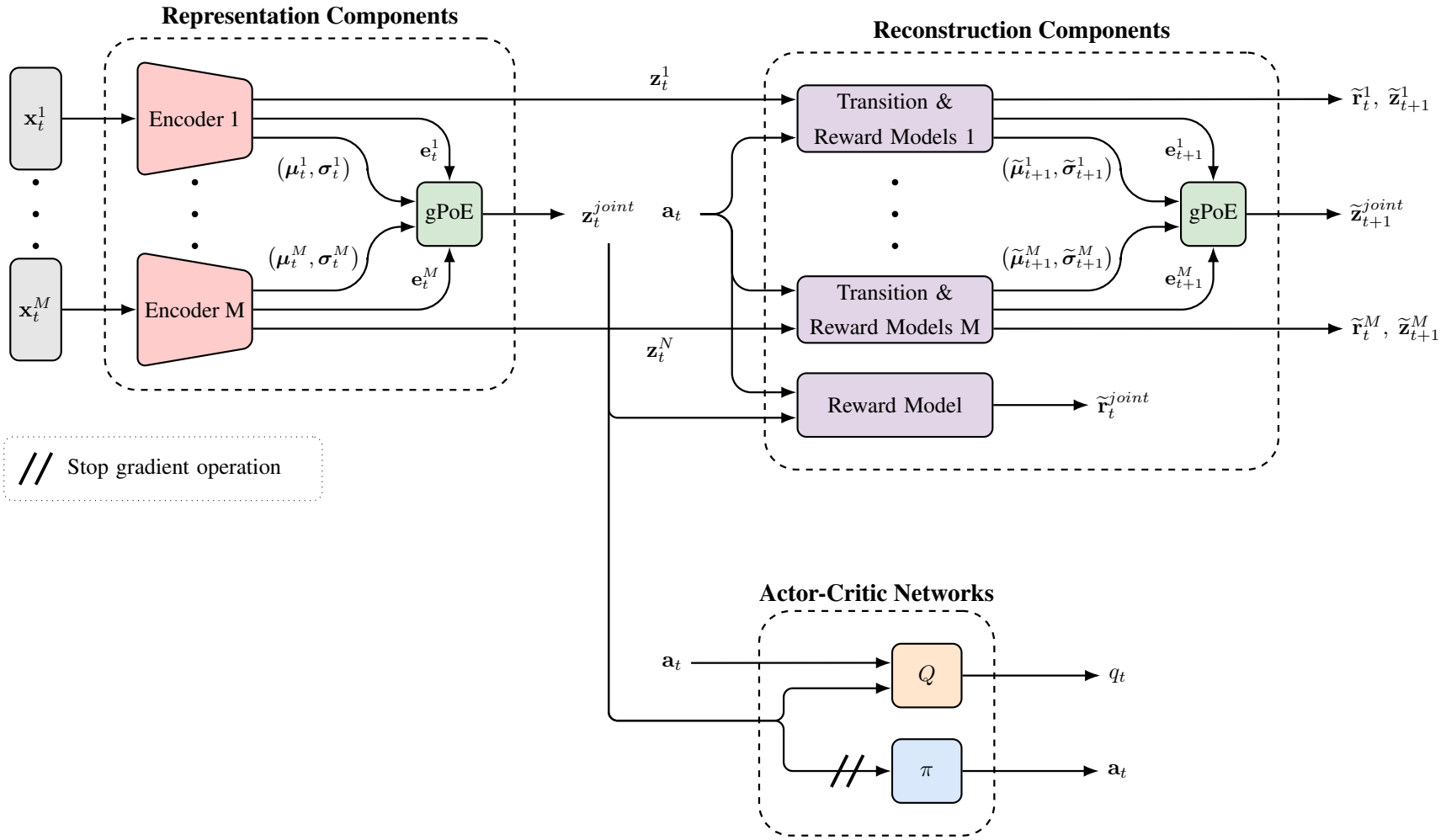


Figure 4.1: Overview of the proposed method architecture. MVDeepMDP consists of encoders and transition models for each observation, together with a shared reward decoder. The latent representation for each modality is computed and merged using gPoE. The resulting joint multimodal latent representation is used as input for actor and critic networks.

4.1.4 Learning Objectives

The proposed approach involves learning a joint latent representation of the multimodal observation space. For each modality, MVDeepMDP independently computes a latent representation $\mathbf{z}_t^i \sim q_{\phi_i}\left(\mathbf{z}_t^i \middle| \mathbf{x}_t^i\right)$ and predicts the transition in the latent space $\widetilde{\mathbf{z}}_{t+1}^i \sim p_{\phi_i}\left(\mathbf{z}_{t+1}^i \middle| \mathbf{z}_t^i, \mathbf{a}_t\right)$ and the reward $r_t \sim p_\phi\left(r_t \middle| \mathbf{z}_t^i, \mathbf{a}_t\right)$. Furthermore, the joint latent representation is sampled from $\mathbf{z}_t^{joint} \sim q_\Phi\left(\mathbf{z}_t^{joint} \middle| \mathbf{X}_t\right)$ and the joint transition $\widetilde{\mathbf{z}}_{t+1}^{joint} \sim p_\Phi\left(\mathbf{z}_{t+1}^{joint} \middle| \mathbf{Z}_t, \mathbf{a}_t\right)$. The components of the MVDeepMDP model, as detailed in Section 4.1.3, are optimized together to maximize the variational lower bound (ELBO [50]). In this proposed approach, the bound includes reward reconstruction terms and a KL regularizer for the transition:

$$\begin{aligned}\mathcal{J}_{\text{DYN}}(\mathbf{\Phi}) = \frac{1}{M}\sum_{i=0}^{M} & D_{\text{KL}}\left(q_{\phi_i}\left(\mathbf{z}_{t+1}^i \middle| \mathbf{x}_t^i\right) \;\middle\|\; p_{\phi_i}\left(\widetilde{\mathbf{z}}_{t+1}^i \middle| \mathbf{z}_t^i, \mathbf{a}_t\right)\right) \\ & + D_{\text{KL}}\left(q_{\mathbf{\Phi}}\left(\mathbf{z}_{t+1}^{joint} \middle| \mathbf{X}_t\right) \;\middle\|\; p_{\mathbf{\Phi}}\left(\widetilde{\mathbf{z}}_{t+1}^{joint} \middle| \mathbf{Z}_t, \mathbf{a}_t\right)\right)\end{aligned} \tag{4.12}$$

$$\begin{aligned}\mathcal{J}_{REW}(\mathbf{\Phi}) = \frac{1}{M}\sum_{i=0}^{M} & \mathbb{E}_{\mathbf{z}_t^i \sim q_{\boldsymbol{\phi}_i}(\cdot|\mathbf{x}_t^i)}\left[\log p_{\boldsymbol{\phi}_i}\left(r_t \middle| \mathbf{z}_t^i, \mathbf{a}_t\right)\right] \\ & + \mathbb{E}_{\mathbf{z}_t^{joint} \sim q_{\mathbf{\Phi}}(\cdot|\mathbf{X}_t)}\left[\log p_{\mathbf{\Phi}}\left(r_t \middle| \mathbf{z}_t^{joint}, \mathbf{a}_t\right)\right]\end{aligned} \tag{4.13}$$

$$\mathcal{J}_M(\mathbf{\Phi}) = \mathcal{J}_{REW}(\mathbf{\Phi}) - \beta\mathcal{J}_{DYN}(\mathbf{\Phi}) \tag{4.14}$$

The parameters $\mathbf{\Phi}$ of the representation and reconstruction models are updated using stochastic back-propagation [55]. The impact of the KL regularizer is controlled by β [39], and in this study, it is set to $\beta = 1$. As discussed in Section 2.3, the reconstruction term in the ELBO is commonly approximated by the MSE loss. However, in the context of MVDeepMDP, the reward prediction term is approximated by the negative NLL loss, with the standard deviation of the Gaussian distribution set to $\sigma = 0.2$. Using the NLL loss instead of the MSE loss ensures the model's robustness to reward prediction, as the NLL loss is less sensitive to outliers. The optimization procedure for MVDeepMDP is outlined in Algorithm 7.

4.2 Experimental Setup

In the FMS environment, automated machines must be able to quickly adapt to new products or sometimes even new tasks. Conventional applications involve mainly manual reprogramming of robotic manipulators or the use of tool changers. However, manual reprogramming is time-consuming and requires a skilled operator, while tool changers are expensive and require additional space. Therefore, a more advanced solution, such as the RL-based method, is needed to address this problem. Nonetheless, the RL-based method still encounters difficulties in generalizing to new tasks and often requires retraining from scratch.

In the following sections, the MVDeepMDP is evaluated in terms of its generalization capability through a series of experiments. Performance analysis begins with a quantitative evaluation, in which the trained policy is tested in various scenarios reflecting real-world challenges. Then a qualitative evaluation is conducted to analyze the impact of the design choices included in the proposed method. The experiments are conducted in the same environment as in Chapter 3, with the same setup.

Algorithm 7 Multimodal Variational DeepMDP

Parameters: learning rates η_Q, η_π, η_M, target update coefficient υ

Initialize function approximators parameters $\theta_Q, \theta_\pi, \mathbf{\Phi}$, temperature coefficient α

Replay buffer $\mathcal{D}$

for each iteration **do**
  Initialize state $\mathbf{X}_t$
  for each environment step **do**
    $\mathbf{z}_t^{joint} \sim p_{\mathbf{\Phi}}^{gPoE}(\mathbf{z}_t|\mathbf{X}_t)$ ▷ Sample latent state at time t
    $\mathbf{a}_t \sim \pi_\theta\left(\mathbf{a}_t \middle| \mathbf{z}_t^{joint}\right)$ ▷ Sample action from policy
    $\mathbf{X}_{t+1} \sim Pr(\mathbf{X}_{t+1}|\mathbf{X}_t, \mathbf{a}_t), r_t = R(\mathbf{X}_t, \mathbf{a}_t)$ ▷ Sample next state and reward
    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{X}_t, \mathbf{a}_t, r_t, \mathbf{X}_{t+1})$ ▷ Store transition in replay buffer
    $\mathbf{X}_t \leftarrow \mathbf{X}_{t+1}$
  end for
  for each update step **do**
    $\{(\mathbf{X}_t, \mathbf{a}_t, r_t, \mathbf{X}_{t+1})\} \sim \mathcal{D}$ ▷ Sample batch of transitions from replay buffer
    $\theta_Q \leftarrow \theta_Q + \eta_Q \nabla_{Q_\theta} \mathcal{J}_Q(\theta)$ ▷ Update Q-function parameters with (4.9)
    $\mathbf{\Phi} \leftarrow \mathbf{\Phi} + \eta_M \nabla_M \mathcal{J}_M(\mathbf{\Phi})$ ▷ Update model parameters with (4.14)
    $\theta_\pi \leftarrow \theta_\pi + \eta_\pi \nabla_{\pi_\theta} \mathcal{J}_\pi(\theta)$ ▷ Update policy parameters with (4.10)
    $\alpha \leftarrow \alpha + \eta_\alpha \nabla_\alpha \mathcal{J}_\alpha(\alpha)$ ▷ Update temperature coefficient with (4.11)
    $\bar{\theta} \leftarrow (1-\upsilon)\bar{\theta} + \upsilon\theta$ ▷ Update $\bar{\theta}$ using EMA of θ, where υ defines Polyak coefficient
  end for
end for

4.2.1 Training Procedure

In the conducted experiments, the training procedure for MVDeepMDP closely resembled the one outlined in Section 3.5.1. However, in this case, the disturbance values for the initial pose were set to $l^{xy} = 5\,\mathrm{mm}$ and $l^{\psi} = 15°$, as in these experiments, the transferability of the trained policy to new tasks was the main focus. Under these conditions, MVDeepMDP was trained for approximately 3 hours, with a noticeable convergence in success rate observed after 18 minutes.

MVDeepMDP representation and reconstruction models were trained using the AdamW optimizer [69], with a learning rate of 0.0003 and a weight decay of 0.01. On the other hand, the RL component of MVDeepMDP was trained using the SAC algorithm with the Adam optimizer [53] and the same learning rate. Optimizing the parameters of the MVDeepMDP model with the AdamW optimizer was found to be more stable and efficient than using the Adam optimizer. The hyper-parameters used for MVDeepMDP can be found in Chapter A.

4.3 Towards Generalization in Flexible Manufacturing Systems

To assess the potential of MVDeepMDP in terms of generalization, two test scenarios were designed to simulate real-world challenges in the FMS environment. The first scenario evaluates the resilience of the trained policy when the background view changes. The second scenario assesses the transferability of the trained policy to new product types within the same domain. Finally, to obtain complete results, the proposed method is evaluated in terms of its cross-domain transferability to objects from a different domain. To provide a comparative evaluation, the MVDeepMDP method was benchmarked against the following state-of-the-art methods:

SAC [31]: This algorithm serves as a core component of the framework for industrial insertion tasks presented in Chapter 3. Moreover, it is considered a benchmark for state-of-the-art model-free learning methods. The same policy checkpoint was used in those experiments as in Section 3.5.

DrQ [17]: This is an extension of SAC that applies a random shift augmentation to visual modalities. It is designed to improve the sample efficiency of the off-policy algorithms, as well as to slightly improve the robustness of the learned policy.

SVEA [84]: This is an extension of DrQ that involves applying extra random convolution augmentation to visual modalities. The hard augmentation in SVEA is specifically applied to the critic's objective. It has been observed that this type of augmentation enhances the robustness of the learned policy against background perturbations.

PAD [36]: This method integrates self-supervised learning with the actor-critic algorithms to adapt to the test environment. In this experiment, the self-supervised task is inverse dynamics prediction, since the original implementation of PAD was designed for continuous control tasks.

DeepMDP [13]: This algorithm forms the foundation of the MVDeepMDP. It combines a semi-supervised objective with an RL. This objective learns a transition model to predict future latent representations and a decoder to predict a reward. In this implementation, the encoders' embeddings are concatenated and fed into the projection layer (see Section 3.4). The output of the projection layer forms the deterministic latent representation.

DBC [3]: The algorithm utilizes bisimulation metrics to learn an invariant representation directly in the latent space. The DBC method improves the generalization capability of the learned policy by extracting task-relevant features from the latent space. As in the DeepMDP, the latent representation of the DBC is obtained from the projection layer.

To ensure a fair comparison, the feature extractor design described in Section 3.4 was used for all of the evaluated methods. The benchmark RL agents were trained for approximately 2.5 hours of training, with noticeable successful repetitions observed between 12 and 20 minutes, depending on the method. When evaluating the trained policies, the metrics introduced in Section 3.5.2 were gathered and analyzed. The results of the experiments are detailed in the following sections.

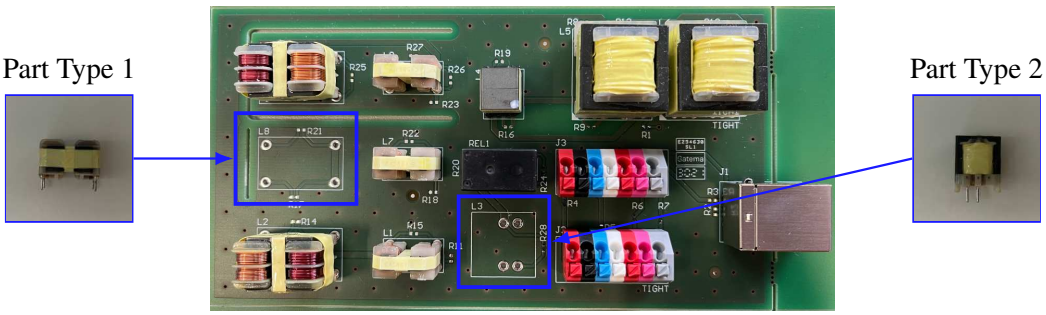


Figure 4.2: Partially assembled PCB variant 1. The target insertion position of part type 1 is surrounded by additional components, and thus it can be seen as an adversarial case.

4.3.1 First Test Scenario: Robustness

The first test scenario evaluates whether the MVDeepMDP can insert elements that were used during training under perturbations of the background view. In the area of electronic assembly, this scenario can be seen as a differing PCB layout or a partially assembled PCB. This holds significant importance from a production perspective, particularly in HMLV production setups, where product configurations are frequently modified. Additionally, this capability has the potential to facilitate the training of an RL agent off the production line and the deployment of it in a real-world robotic cell.

The policies were trained using Electronic Part 1 and PCB Layout 1 (see Figure 3.7a). Subsequently, the trained policies were evaluated in two different test cases. The first test case reflects the scenario in which the PCB is partially assembled; thus, the background view is different from that used during training. The partial assembly is depicted in Figure 4.2. It is important to note that the surrounding components can be viewed as adversarial perturbations since they may obstruct the target location. The second test case reflects the scenario in which the PCB layout is different from the one used during training. The PCB Layout 2 was used in this test case, as illustrated in Figure 3.7b. This PCB varies from the one used during training in terms of laminate color and hole clearance, making it a more challenging task. For the purposes of this experiment, this partially assembled PCB (PCBA) will be referred to as *PCBA Layout 1*. The additional results for the electronic part type 2 are presented in Section B.2.

In each test case, 100 insertion trials were conducted for each policy and performance metrics were collected. The results can be found in Table 4.1. It can be observed that most of the methods effectively inserted electronic parts into the target place on a partially assembled PCB. This was an expected outcome, as the color of the laminate and the layout design were utilized during training, albeit with additional surrounding components. Furthermore, the hole clearances matched those in the training environment, which significantly influenced the performance of the learned policy. However, when it came to inserting components into a PCB that differed in all aspects from the one used during training, the best performing methods were MVDeepMDP, DBC, and SVEA. These methods achieved a success rate close to 100%, demonstrating their resilience to background disturbances. MVDeepMDP and DBC achieved such results by learning a task-relevant feature representation, while SVEA improved the robustness of the learned policy to background perturbations by applying hard augmentation in the critic's objective.

Table 4.1: Comparison of benchmark methods and MVDeepMDP to handle background perturbations. For each method and layout, 100 insertion trials were performed. The results show that MVDeepMDP along with SVEA and DBC were the most successful methods in this test scenario.

	Success rate [%]		**Mean time [s]**	
	PCBA Layout 1	PCB Layout 2	PCBA Layout 1	PCB Layout 2
SAC	4	11	8.94	5.25
DrQ	66	5	4.98	8.93
SVEA	**100**	**100**	**1.77**	1.86
PAD	99	60	2.38	7.89
DeepMDP	**100**	73	2.07	4.50
DBC	**100**	99	1.81	2.61
MVDeepMDP	**100**	**100**	1.81	**1.77**

4.3.2 Second Test Scenario: Transferability

In the next phase, trained policies were evaluated in a second test scenario to determine their transferability to new product types within the same domain. The evaluation involved training the policies using Electronic Part 1 and PCB Layout 1, and then testing them on the remaining Electronic Parts (see Figure 3.7). This experiment is more challenging than the previous one, as the test components differed in the number and arrangement of leads. Moreover, each component had a unique PCB layout designed to potentially disrupt policy inference. From an FMS point of view, the ability to quickly adapt to new products is crucial, as it can minimize downtime and increase overall efficiency of the production line.

As in previous experiments, 100 insertion trials were conducted for each selected test component and policy and performance metrics were recorded. The results are detailed in Table 4.2. It can be immediately seen that the MVDeepMDP outperformed all other algorithms. In four out of the six test cases, the success rate was nearly 100% or even 100%, while for the remaining two electronic parts, the success rate was close to 90%. Among the compared algorithms, only SVEA and DBC achieved relatively high results. However, as emphasized in Section 3.5.2, the mean assembly time is also an important factor in the FMS environment. Further examination of the recorded assembly times revealed that MVDeepMDP completed the task more quickly than the benchmark algorithms. For Electronic Parts 3, 4, and 5, MVDeepMDP completed the task in approximately 2 seconds, while for the remaining electronic parts, the average assembly time ranged between 2.5 and 3.5 seconds. In contrast, the SVEA and DBC methods required significantly more time to complete the task, with the average assembly time ranging from 3 s to 9 s. The least effective methods were SAC and DrQ, which achieved a success rate below 30% in most test cases.

The results of this experiment suggest that acquiring a dynamic representation or integrating more challenging augmentations during training significantly improves transferability to other objects within the same domain. While soft augmentations such as random shift enhance the algorithm's sample efficiency, they do not contribute to its generalization capability. Of particular interest is the performance of the PAD algorithm,

Table 4.2: Comparison of the transferability of benchmark methods and MVDeepMDP to unseen objects within the same domain. The objects used in this test scenario are electronic parts shown in Figure 3.7. For each object and method, the 100 insertion trials were performed. The results show that MVDeepMDP outperforms the benchmark methods in terms of success rate and mean time to complete the task.

	Success rate [%]					
	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
SAC	6	7	1	15	10	5
DrQ	2	0	28	18	13	27
SVEA	43	95	84	12	68	64
PAD	1	32	81	18	6	52
DeepMDP	4	93	91	81	29	31
DBC	56	97	80	79	90	21
MVDeepMDP	**87**	**100**	**100**	**100**	**99**	**85**
	Mean time [s]					
SAC	8.07	8.71	9.10	8.23	6.34	8.93
DrQ	8.57	9.20	7.38	7.96	8.42	7.56
SVEA	6.43	2.76	3.41	8.41	4.99	5.07
PAD	13.85	10.94	5.09	11.91	13.27	8.17
DeepMDP	9.02	2.54	2.92	4.02	7.43	7.37
DBC	6.04	2.91	3.82	4.31	3.72	8.11
MVDeepMDP	**3.59**	**2.15**	**1.83**	**2.08**	**2.13**	**3.55**

which adapts during deployment. Although adaptation during deployment theoretically should improve the generalization capability of the learned policy, the metrics reveal that only learning the inverse dynamics is insufficient for effective adaptation in the insertion task.

4.3.3 Test-time Adaptation

The results presented above show that MVDeepMDP can perform a zero-shot transfer to unseen objects within the same domain in most cases. However, it did not achieve a 100% success rate for all test cases. Specifically, for Electronic Parts 2 and 6, the success rate was below 90%. This raises the question: Can MVDeepMDP be fine-tuned to achieve a success rate 100% for the test cases where it failed in zero-shot transfer? To address this question, a test-time adaptation experiment was conducted. MVDeepMDP was fine-tuned in the test cases where it failed to achieve a 100% success rate. The adaptation involved training the pre-trained MVDeepMDP for 100 episodes using the same hyperparameters as in the initial training phase. The results of the test-time adaptation experiment are presented in Figure 4.3. It can be seen that the proposed method achieved a success rate of 100% in 30 episodes for Part 2 and 40 episodes for Part 6, which corresponds to approximately 5 minutes of training. These results indicate that MVDeepMDP can

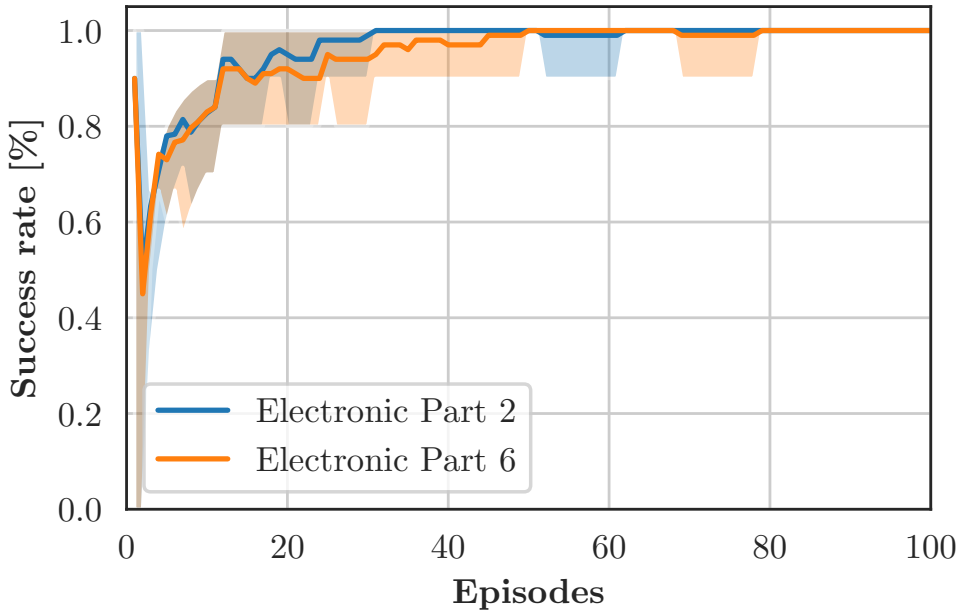


Figure 4.3: Test-time adaptation of the MVDeepMDP method. The pretrained MVDeepMDP was fine-tuned on the test cases from the transferability experiment (refer Section 4.3.2) where it failed to achieve a 100% success rate. The fine-tuning was performed for 100 episodes with the same hyper-parameters as in the training phase.

be effectively fine-tuned if it cannot perform zero-shot transfer. This is particularly important in the FMS environment, as test-time adaptation is significantly faster than training the RL agent from scratch.

4.3.4 Cross-domain Transferability

The previous test scenarios evaluated the transferability of the learned policy to unseen objects within the same domain, an important factor in FMS, particularly in HMLV production lines. However, it would be beneficial from a production perspective to have a method that can quickly adapt to objects from a different domain. Hence, this experiment aims to evaluate the cross-domain transferability of the MVDeepMDP method. The policy was trained using Electronic Part 1 and PCB Layout 1, and then tested on 3D-printed blocks placed in a robot's workspace. These custom 3D-printed blocks are depicted in Figure 4.4. The blocks were designed to simulate a simplified version of real-world insertion tasks. The blocks differ from the electronic parts in overall appearance and size. The results of the experiment are detailed in Table 4.3. It can be seen that MVDeepMDP achieved a success rate a 100% in all test cases, demonstrating its resilience to cross-domain transferability. Furthermore, the average assembly time ranged from 1.68 s to 2.42 s. However, it is important to note that the blocks were specifically designed for experimentation purposes, making this task a simplified version of real-world insertion. For a more comprehensive evaluation, MVDeepMDP should be tested with a wider range of objects from different domains. Nevertheless, the results of this experiment suggest that the MVDeepMDP method has the potential to be transferred to objects from different domains, which is an important feature from a production perspective.

Figure 4.4: The presentation of the test bed with 3D-printed blocks. The blocks differ in shape and size. In this test bed, the clearance of insertion holes is 2 mm

Table 4.3: Evaluation of the proposed method in terms of transferability to objects from a different domain. As objects from the new domain, 3D printed blocks were used (see Fig. 4.4). The obtained results show that MVDeedMDP is capable of transferring the learned policy to the objects from the new domain, achieving a success rate of 100%. However, the 3D printed blocks were designed to serve as a simply test bed, thus further evaluation on more complex objects is required.

	Block 1	**Block 2**	**Block 3**	**Block 4**	**Block 5**	**Block 6**
Success Rate [%]	100	100	100	100	100	100
Mean Time [s]	1.79	2.42	1.68	1.93	1.89	1.82

4.4 Modality Mixing Mechanism

In order to analyze whether the proposed modality mixing mechanism plays a crucial role in learning unimodal encoders being able to generalize to new objects, the proposed technique is compared with other mechanisms existing in the literature. Given that MVDeepMDP is built on the concept of cross-modal VAE, a natural comparison is made with the Product-of-Experts and Mixture-of-Experts, which are widely used for integrating information from multiple modalities in latent space. Moreover, the naive approach is also considered to provide a comprehensive analysis, where the latent representation is obtained from the mean of the unimodal latent representations. To ensure a fair comparison, only the fusing mechanism is modified, while the rest of the MVDeepMDP architecture remains unchanged.

Product-of-Experts is not a new concept in mathematics, but its first application in multimodal variational autoencoders was introduced by Wu and Goodman [125]. This mechanism involves computing the joint latent representation as a product of unimodal posterior distributions. The detailed formulation of the PoE mechanism is presented in Equations (4.1) to (4.3). Since the PoE mechanism can be viewed as a special case of the gPoE mechanism with equal weights, the same learning objectives as in MVDeepMDP were utilized. This characteristic enables an unbiased comparison between the two methods.

Mixture-of-Experts for the mixing multimodal information in the latent space was proposed by Shi et al. [99]. They proposed a mechanism called the mixture-of-experts multimodal variational autoencoder. This mechanism factorizes joint posterior by weighted averaging of individual posteriors. Instead of learning individual weights for the modalities, MMVAE suggests giving equal weight to each modality present to avoid a dominant-modality issue as in PoE. Assuming the same training strategy as in the MVDeepMDP, the ELBO objective for the MVDeepMDP with the MoE mechanism is as follows:

$$\begin{aligned}\mathcal{J}_{\text{MoE}}(\mathbf{\Phi}) = \frac{1}{M}\sum_{i}^{M}\Bigg(&\mathbb{E}_{\mathbf{z}_t^i\sim q_{\phi_i}(\cdot|\mathbf{x}_t^i)}\left[\log p_\Phi\left(\mathbf{r}_t\middle|\mathbf{z}_t^i,\mathbf{a}_t\right)\right]\\&- D_{\text{KL}}\big(q_\Phi\left(\mathbf{z}_{t+1}^i\middle|\mathbf{x}_t^{1:M}\right)\parallel p_\Phi\left(\mathbf{z}_{t+1}^i\middle|\mathbf{z}_t^{1:M},\mathbf{a}_t\right)\big)\Bigg)\end{aligned} \tag{4.15}$$

Naive Approach is a simple mechanism that computes the joint latent representation as the mean of the unimodal latent representations. The formula for the naive approach is as follows:

$$\mu_{\text{naive}}(\mathbf{Z}) = \frac{1}{M}\sum_{i}^{M}\mu_i(\mathbf{z}_i) \tag{4.16}$$

$$\Sigma_{\text{naive}}(\mathbf{Z}) = \frac{1}{M}\sum_{i}^{M}\Sigma_i(\mathbf{z}_i) \tag{4.17}$$

$$\mathbf{z}_{\text{naive}} \sim \mathcal{N}\left(\mu_{\text{naive}}(\mathbf{Z}), \Sigma_{\text{naive}}(\mathbf{Z})\right) \tag{4.18}$$

Since the output of the naive approach is a Gaussian distribution, the same learning objectives those of the MVDeepMDP were utilized.

To compare the performance of the proposed method with the mechanisms mentioned above, all variants of MVDeepMDP were trained using the same procedure described in Section 4.2. Furthermore, the trained policies were evaluated based on the second test scenario (see Section 4.3.2) as it serves as a suitable benchmark to assess the transferability of the learned policy. The results of the experiment are depicted in Figure 4.5.

First, let's delve into the training process of the RL agents. It is worth noting that, apart from the MoE mechanism, all methods achieved convergence in approximately 6000 training steps. On the contrary, the MoE mechanism required more than 20000 training steps to converge. The trained policies were then tested for potential transferability to new objects within the same domain. In particular, the MVDeepMDP with the gPoE mechanism outperformed all the compared methods. Surprisingly, the naive approach yielded slightly worse results compared to the original implementation.

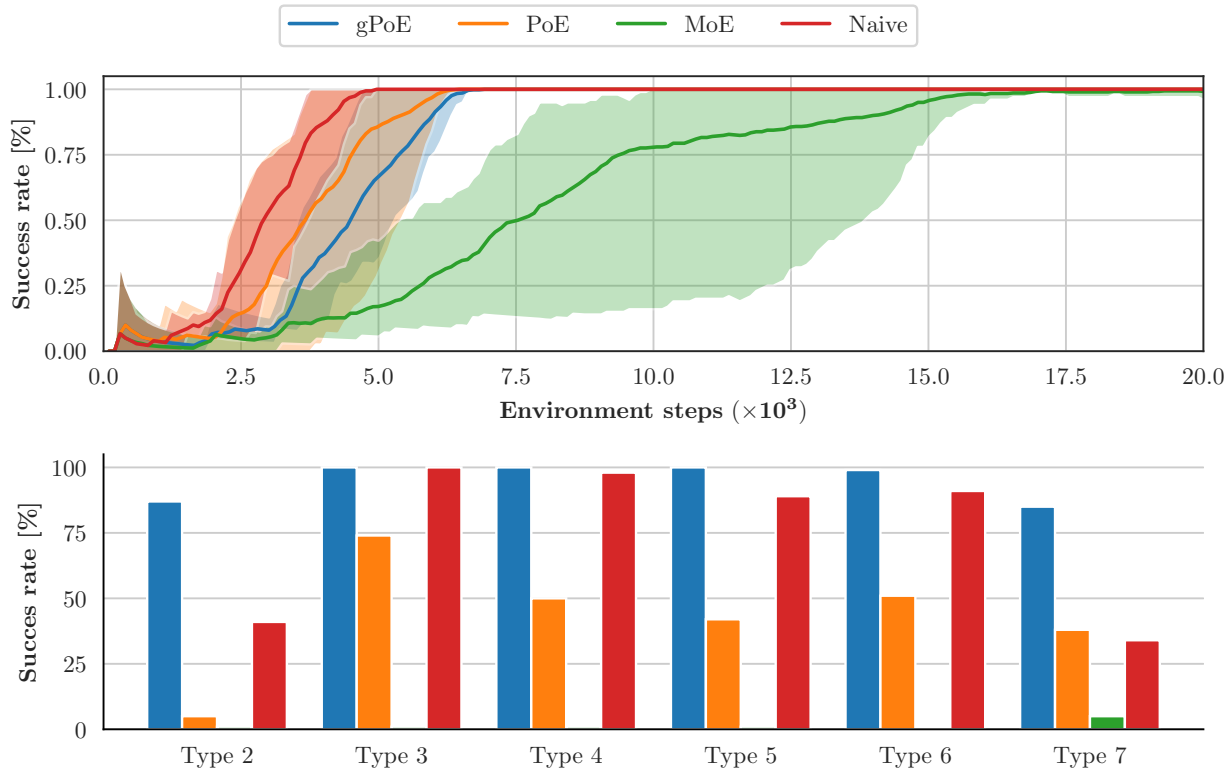


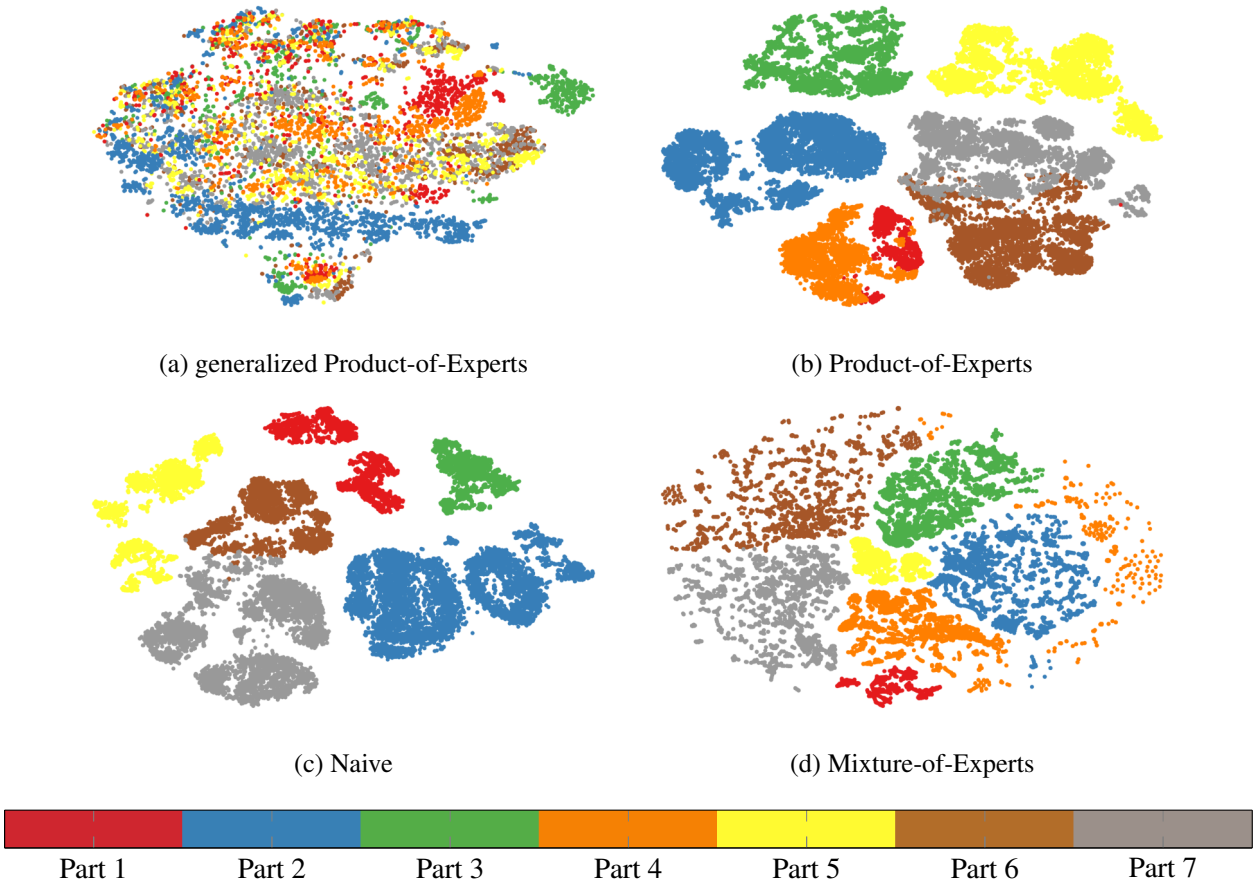
Figure 4.5: Comparison of alternative methods for fusing information from unimodal posterior distributions. For PoE and Naive, the learning objectives remains unchanged, while for MoE, the learning objective was adjusted accordingly. The results show that the proposed gPoE leads to better performance than the alternatives, as it allows to balance the impact of each modality on the final prediction. **Top:** Training results, for each method, five runs of different random seeds were performed. The shaded area represents the minimum and maximum values, while the bold line represents the mean. For a better visualization, the results are truncated at 20000 environment steps. **Bottom:** Evaluation results obtained from the second test scenario.

Conversely, the variant with the MoE mechanism failed to transfer the learned policy to unseen objects, suggesting that the MoE mechanism may not be suitable for learning multimodal dynamic latent representation due to its modified learning objective. Finally, the PoE mechanism achieved moderate performance, with success rates ranging between 40% and 70%. This can be attributed to the fact that the joint latent representation obtained by the PoE mechanism tends to be dominated by the modality with the highest weight, potentially leading to the inability to extract task-relevant features.

Nevertheless, the results demonstrate that the proposed gPoE mechanism significantly improves the generalizability of feature representation learning, thus improving the transferability of the learned policy.

4.4.1 t-SNE Visualization

The previous analysis demonstrated the benefits of the proposed gPoE mechanism over existing methods to fuse multimodal information in the latent space. To provide a more comprehensive analysis of the impact of the gPoE balancing mechanism on generalization capability, t-distributed Stochastic Neighbor Embedding (t-SNE) was used to cluster the joint multimodal latent representation of all compared methods. t-SNE is a



(a) generalized Product-of-Experts

(b) Product-of-Experts

(c) Naive

(d) Mixture-of-Experts

Figure 4.6: t-SNE visualization of the learned joint latent representations of different information fusion methods. The clusters represent different types of electronic parts. The results show that the proposed gPoE leads to more or less a homogeneous cluster, with not clear separation between the different types of electronic parts. In contrast, the other methods create more separated clusters, which indicates less effective extraction of shared information between the different modalities.

method that transforms high-dimensional Euclidean distances into conditional probabilities, capturing similarities between adjacent data points. The resulting visualization of the joint latent representation is shown in Figure 3.7, while the visualizations of the unimodal latent representations are presented in Section B.3.

Let's focus on the clusters generated by the multimodal latent representations, it can be immediately noticed that the clusters generated by the gPoE differ significantly from those generated by the remaining methods. The t-SNE output of the gPoE mechanism resembles a more or less single cluster without a distinct separation of the electronic parts, indicating potential shared representations between the electronic parts. On the contrary, the t-SNE output of the other methods shows a clear separation of the clusters. This observation led us to conclude that the weight-balancing mechanism improves the ability to learn task-relevant features, a critical factor in achieving generalization. Moreover, the results suggest that the gPoE mechanism is more effective in extracting meaningful features from the multimodal latent representation, thus improving the generalization capability of the learned policy.

4.5 Design Ablation Study

Knowing the impact of the gPoE mechanism on the generalization capability of the MVDeepMDP, let's delve into the ablation study to gain a deeper understanding of design choices incorporated into the proposed method. The ablation study involved training the MVDeepMDP model with the following modifications:

1. **No Reward Prediction:** The reward prediction objective Equation (4.13) was removed from the training procedure. Hence, the MVDeepMDP parameters were optimized using only the transition objective Equation (4.12).

2. **No Views Objectives:** The view-based objectives were removed from Equation (4.14). Consequently, the MVDeepMDP parameters were optimized only using the objectives related to the joint latent representation. Therefore, the update rule is as follows:

$$\begin{aligned}\mathcal{J}_{\mathrm{M}}(\boldsymbol{\Phi}) = \mathbb{E}_{\mathbf{z}_t^{joint}\sim q_{\boldsymbol{\Phi}}(\cdot|\mathbf{X}_t)}\left[\log p_{\boldsymbol{\Phi}}\left(r_t\middle|\mathbf{z}_t^{joint},\mathbf{a}_t\right)\right] \\ - \beta D_{\mathrm{KL}}\Big(q_{\boldsymbol{\Phi}}\left(\mathbf{z}_{t+1}^{joint}\middle|\mathbf{X}_t\right)\ \Big\|\ p_{\boldsymbol{\Phi}}\left(\mathbf{z}_{t+1}^{joint}\middle|\mathbf{Z}_t,\mathbf{a}_t\right)\Big)\end{aligned} \tag{4.19}$$

3. **No gPoE Joint Transition:** In the original approach, the joint dynamic latent representation $\mathbf{z}_{t+1}^{joint}$ was derived by fusing the unimodal dynamic latent representations using the gPoE mechanism. In this variation, the joint dynamic latent representation is defined by a single feed-forward neural network, which takes the joint latent representation $\mathbf{z}_t^{joint}$ and the action $\mathbf{a}_t$ as input. As the gPoE mechanism is not used, the transition objective Equation (4.12) is adjusted accordingly:

$$\begin{aligned}\mathcal{J}_{\mathrm{DYN}}(\boldsymbol{\Phi}) = \frac{1}{M}\sum_{i=0}^{M} D_{\mathrm{KL}}\big(q_{\boldsymbol{\phi}_i}\left(\mathbf{z}_{t+1}^{i}\middle|\mathbf{x}_t^{i}\right)\ \big\|\ p_{\boldsymbol{\phi}_i}\left(\widetilde{\mathbf{z}}_{t+1}^{i}\middle|\mathbf{z}_t^{i},\mathbf{a}_t\right)\big) \\ + D_{\mathrm{KL}}\Big(q_{\boldsymbol{\Phi}}\left(\mathbf{z}_{t+1}^{joint}\middle|\mathbf{X}_t\right)\ \Big\|\ p_{\boldsymbol{\Phi}}\left(\widetilde{\mathbf{z}}_{t+1}^{joint}\middle|\mathbf{z}_t^{joint},\mathbf{a}_t\right)\Big)\end{aligned} \tag{4.20}$$

The trained policies were evaluated using the second test scenario (see Section 4.3.2). The results of the ablation study are shown in Figure 4.7. The training process plots were generated from the training logs of five runs with different random seeds. When analyzing the training process, it can be seen that all variants converged between 6000 and 8000 environment steps. Therefore, based solely on the training process, it is challenging to determine which design choice has the most significant impact on the performance of the MVDeepMDP. However, a more insightful analysis can be derived from the evaluation results. It can be immediately observed that the most significant performance drop occurred when a single feed-forward neural network parameterized the joint dynamic latent representation. This indicates that the gPoE mechanism plays a crucial role in learning a task-relevant feature representation.

Furthermore, further analysis suggests that the reward prediction objective has a minor impact on the performance of the MVDeepMDP, as the variant without the reward prediction objective achieved the closest performance to the original implementation. Finally, the variant without view-based objectives achieved slightly lower performance compared to variants without the reward prediction objective. Hence, it can be concluded that the multimodal dynamic latent representation with the gPoE mechanism is the key component of the MVDeepMDP design.

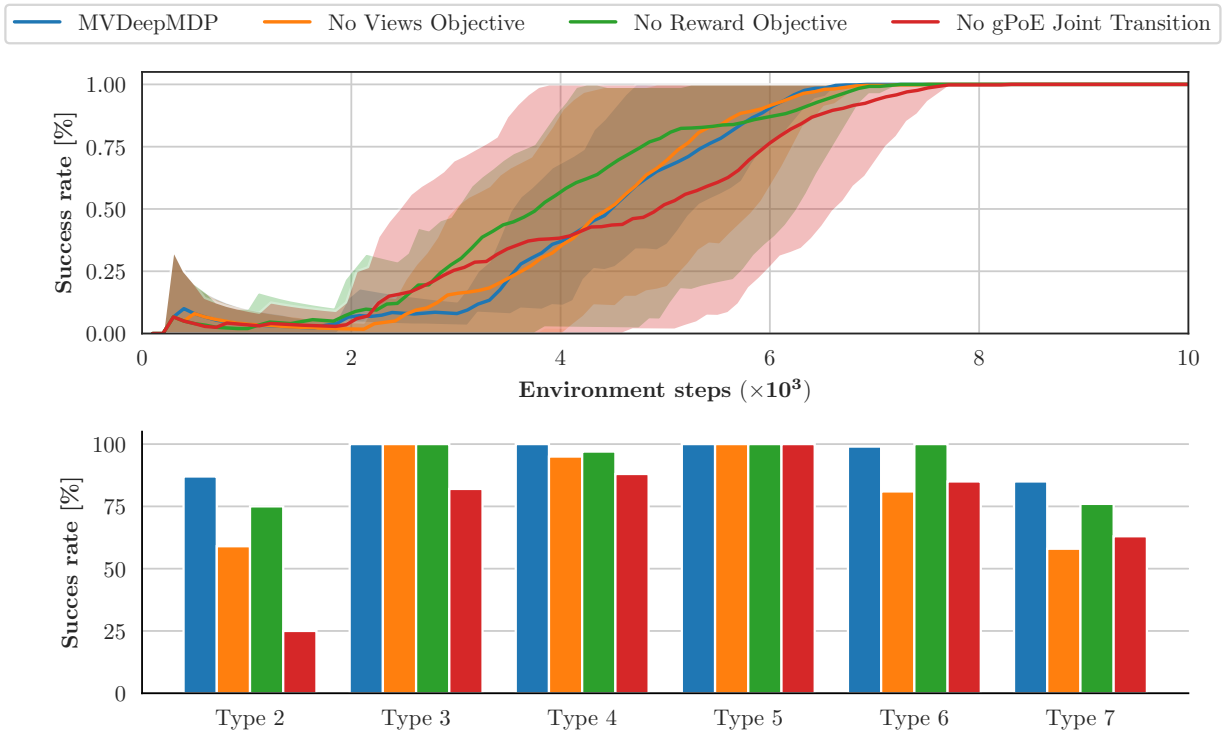


Figure 4.7: Comparison of different design choices for the proposed architecture. The results show that the most critical component is the use of the proposed gPoE to compute the joint transition distribution $p(\mathbf{z}_{t+1}^{joint} \mid \mathbf{Z}_t, \mathbf{a}_t)$. Replace the gPoE with a simple feed-forward network leads to a significant drop in performance. The results also show that the reward prediction objective only slightly improves the overall performance. **Top:** Training results, for each method, five runs of different random seeds were performed. The shaded area represents the minimum and maximum values, while the bold line represents the mean. For a better visualization, the results are truncated at 10000 environment steps. **Bottom:** Evaluation results obtained from the second test scenario.

4.6 Impact of Independently Processing State-Based Modalities

In the field of robotics, it is common for RL methods to integrate state-based modalities, such as pose and contact wrench, and preprocess them using a single feed-forward neural network. However, due to the disparate nature of these modalities in representing physical quantities, some features may not be effectively captured during training. For instance, the magnitude of the wrench vector can be several orders higher than that of the pose data, due to the difference in SI units. Haffner et al. [35] proposed the use of the *symlog* function at the beginning of the neural network as a solution to this challenge. This function is designed to scale down the magnitude of physical quantities represented by large values, such as force data, and is defined as:

$$\text{symlog}(x) \doteq \text{sign}(x) \ln\left(|x| + 1\right) \tag{4.21}$$

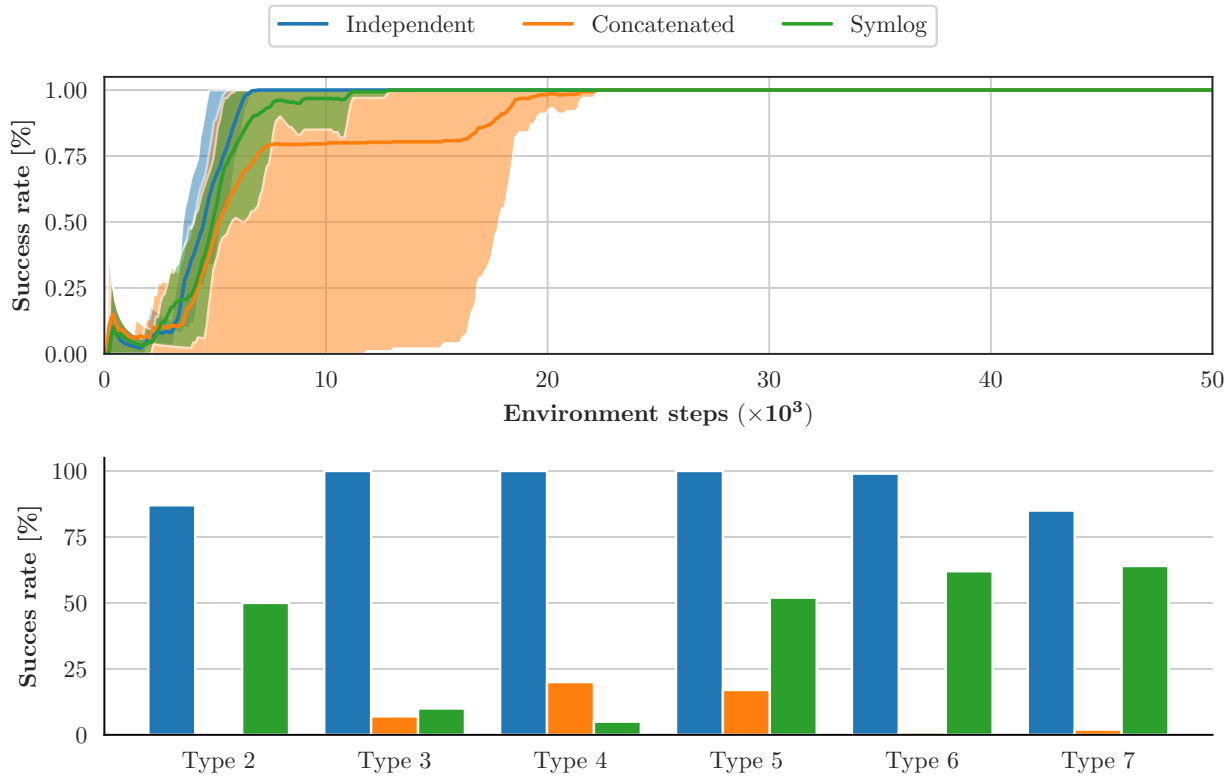


Figure 4.8: Comparison of methods for processing state modalities of different physical quantities. In all cases, the RL framework of MVDeepMDP was used and only the preprocessing of the state modalities was changed. The results show that the proposed method, which uses a separate encoder for each modality, leads to better performance than the alternatives. **Top:** Training results, for each method, five runs of different random seeds were performed. The shaded area represents the minimum and maximum values, while the bold line represents the mean. **Bottom:** Evaluation results obtained from the second test scenario.

The purpose of this experiment is to assess the impact of state-based modalities processing approaches on the performance of MVDeepMDP. Therefore, the MVDeepMDP model was trained with the following variants of state-based modalities processing:

1. **Independent Processing:** The original approach used in MVDeepMDP, where the state-based modalities are independently processed by the corresponding feed-forward encoders.

2. **Concatenated Processing:** State-based modalities are concatenated and processed by a single feed-forward encoder.

3. **Symlog Processing:** State-based modalities are concatenated, modified by the *symlog* function, and processed by a single feed-forward encoder.

The results of the experiment are depicted in Figure 4.8. Upon analysis, it was observed that MVDeepMDP, trained with independently processed state-based modalities, achieved convergence in approximately 6000 training steps. In contrast, models trained with concatenated and *symlog* processed state-based modalities required more than 10000 training steps to converge. Furthermore, the trained policies were evalu-

ated using the second test scenario (refer to Section 4.3.2), serving as a robust performance indicator for the learned policy. The findings revealed that concatenating the state-based modalities and passing them through a single feed-forward encoder significantly deteriorates the performance of the MVDeepMDP. On the other hand, applying the *symlog* function improved the performance of MVDeepMDP compared to the raw concatenation approach, albeit it remained lower than the original implementation. This suggests that independent processing of state-based modalities of different physical quantities is advantageous in extracting meaningful features, thereby improving the performance and transferability of the RL agent.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

This thesis introduces an efficient framework for robotic industrial assembly tasks based on deep reinforcement learning. At the core of this framework is the novel method called *Multimodal Variational DeepMDP*. This method enables quick adaptation of the RL agent to new objects that were not encountered during training. The proposed method was tested on a real-world use case, specifically the insertion of electronic components. This problem provides an excellent benchmark for the proposed method, as electronic parts vary in appearance, size, and number of leads, presenting a challenge for industrial robots.

The MVDeepMDP method has been integrated into a framework for robotic assembly tasks as introduced in Chapter 3. This framework enables efficient training of an RL agent using a real robotic system. Training is performed asynchronously, a crucial aspect for smooth training when the agent interacts with the real-world environment. Integration was achieved using ROS 2 middleware and Ape-X architecture. However, direct integration of ROS 2 with the PyTorch library led to performance degradation due to the thread-calling mechanism of ROS 2. Therefore, a robot-agent communication system was designed that, via the XML-RPC protocol, allows to interact with the robotic system in a non-blocking manner. Additionally, the framework includes an admittance control system, essential for industrial robotic assembly tasks, which enables safe interaction with the environment by adapting to external forces and receiving commands defined in Cartesian space. Finally, the framework introduces the environment in the context of the Markov Decision Process, providing multimodal observation to the RL agent. This observation includes two RGB images acquired from the tool-mounted vision system, the relative pose and twist of the end-effector, and the 6D force/torque sensor readings, offering comprehensive information crucial for real-world robotic tasks.

In later sections of this chapter, the proposed framework's capability was demonstrated using the Soft Actor-Critic algorithm. This algorithm was selected for its proven performance in robotic tasks. Furthermore, the MVDeepMDP method was developed on the basis of the SAC algorithm, making it the natural choice for the experiments. The experiments carried out highlighted the importance of visual observation in rapidly learning a policy capable of solving insertion tasks. The results indicated that while an agent that relied solely on state-based observation achieved a high success rate, the agent incorporating visual observation showed significantly faster convergence during training and greater resilience to initial pose

perturbation. Furthermore, the performance of the RL agent was compared to traditional methods, such as spiral and random search strategies. The results clearly demonstrated that the RL agent outperformed traditional methods, achieving a high success rate in assembly tasks under initial pose perturbation.

The framework for industrial robotic assembly tasks is a crucial aspect of the thesis, but the primary contribution lies in the MVDeepMDP method. This method was designed to enable the RL agent to quickly adapt to unseen objects. This was achieved by developing a multimodal latent dynamic model that learns the transition dynamics for each observation modality and the transition dynamics of the joint representation. The joint latent representation is generated using the generalized Product-of-Experts. Compared to other existing fusion methods, such as the Product-of-Experts, the gPoE is distinguished by its ability to learn the weights of the experts responsible for controlling the relevance of each modality. A series of experiments demonstrated that MVDeepMDP can effectively transfer trained policy to the majority of unseen objects and can also rapidly adapt to those objects for which the initial transfer failed. The performance of the proposed method was compared with state-of-the-art methods addressing generalization problems in the RL domain, such as DBC or SVEA, which serve as simple yet effective baselines. Furthermore, an ablation study was conducted to highlight the importance of each component of the MVDeepMDP method. The detailed analysis illustrated that MVDeepMDP is capable of extracting task-relevant information from the observation.

In summary, the findings of this thesis demonstrate the effectiveness of reinforcement learning as a valuable tool to address real-world challenges in the manufacturing industry. The proposed framework facilitates the training of the RL agent within actual robotic systems, enabling seamless deployment in robotic cells on production lines. Additionally, this research highlights the significance of multimodal observation in achieving efficient learning in robotic assembly tasks and high success rates. Furthermore, rapid adaptation to new objects is crucial in flexible manufacturing systems, particularly in HMLV production lines. The MVDeepMDP method successfully meets these requirements, making it a valuable solution for use cases in the manufacturing industry.

5.2 Future Work

The natural continuation of this research is to perform a thorough evaluation of the proposed framework in real-world industrial environments. Implementing a pilot project in an actual manufacturing environment would provide valuable information to improve both the framework and the MVDeepMDP method. Feedback from production lines would help identify potential bottlenecks and areas for improvement. Additionally, a detailed analysis of the performance of the framework in other manufacturing sectors, such as furniture or automotive industries, would be advantageous. These industries pose distinct challenges, such as managing large objects or complex geometries, which could benefit from the proposed framework. Finally, a future direction involves exploring the scalability of other contact-rich manipulation tasks, such as object grasping, grinding, or soldering. These tasks differ significantly from the insertion task and require different strategies and approaches. Therefore, extending the proposed framework to address these challenges would offer a comprehensive solution for contact-rich manipulation tasks in the manufacturing industry.

In terms of algorithmic development, an exciting direction for future research involves delving into the sequential architecture of the MVDeepMDP method. By adopting a sequential approach, the method would utilize the latent representation of the previous time step in computing the current latent representation. This would allow the agent to capture temporal dependencies between observations and to learn a better model of the environment dynamics. Moreover, the sequential architecture could be implemented using recurrent neural networks like Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells or by leveraging the SLAC approach [61]. The SLAC method is known for its lightweight and efficient nature, employing a simple for loop to unroll the collected sequential data during training.

Finally, it would also be valuable to explore the effects of different observation modalities on the agent's performance as a potential research direction. The current setup utilizes two RGB images acquired from the tool-mounted vision system, the relative pose and twist of the end-effector, and the wrench measurements. However, the framework is flexible and can be easily expanded to contain additional modalities, such as depth images, haptic feedback, or point clouds. Assessing the impact of these modalities on the agent's performance could involve conducting an ablation study similar to the one carried out in this thesis.

Appendix A

Hyper-parameters

The following appendix offers a detailed overview of the hyperparameters used in the algorithms in this dissertation. Firstly, it presents the hyperparameters of the SAC algorithm as shown in Table A.1, which were utilized in the experiments introduced in Chapter 3. Subsequently, it provides the hyperparameters of the MVDeepMDP algorithm as detailed in Table A.2. These hyperparameters were employed in the experiments discussed in Chapter 4.

Table A.1: SAC algorithm hyperparameters.

Parameter name	**Value**
Exploration steps	400
Replay buffer capacity	50000
Prioritized replay α	0.6
Prioritized replay β	0.4
Mini-batch size	128
Discount γ	0.95
Critic optimizer	Adam(lr=0.0003, betas=(0.9, 0.999))
Actor optimizer	Adam(lr=0.0003, betas=(0.9, 0.999))
Temperature optimizer	Adam(lr=0.0003, betas=(0.5, 0.999))
Actor update frequency	2
Actor log stddev bounds	$[-11.5, 1.6]$
Initial temperature	0.01
Target entropy	$-\frac{1}{2}\dim(\mathcal{A})$
Critic Q-function soft-update rate τ_Q	0.01

Table A.2: MVDeepMDP algorithm hyperparameters.

Parameter name	**Value**
Exploration steps	400
Replay buffer capacity	50000
Prioritized replay α	0.6
Prioritized replay β	0.4
Mini-batch size	128
Discount γ	0.95
Critic optimizer	Adam(lr=0.0003, betas=(0.9, 0.999))
Actor optimizer	Adam(lr=0.0003, betas=(0.9, 0.999))
Temperature optimizer	Adam(lr=0.0003, betas=(0.5, 0.999))
Actor update frequency	2
Actor log stddev bounds	$[-11.5, 1.6]$
Initial temperature	0.01
Target entropy	$-\frac{1}{2}\dim(\mathcal{A})$
Critic Q-function soft-update rate τ_Q	0.01
MVDeepMDP optimizer	AdamW(lr=0.0003, betas=(0.9, 0.999), weight_decay=0.01)
Reward decoder stddev	0.2
Latent representation dimension	128
Transition models β coefficient	1.0

Appendix B

Additional Results

B.1 Performance Against Conventional Methods - Remaining Parts

This section provides a comparison of the RL agent's performance with conventional methods for the remaining electronic parts. The corresponding results are presented in the following tables and align with the findings outlined in Section 3.5.5. Across the various electronic parts, the RL agent consistently achieved a success rate close to 100% in the majority of test cases, whereas the conventional methods demonstrated strong performance only in Test Case 1.

Table B.1: Comparison of the RL agent and the conventional methods against the initial pose disturbances. The results are shown for the Electronic Part 2.

	Success rate [%]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	**100**	**98**	**84**	**100**	**99**	**98**	**79**
Straight down	98	0	0	61	26	0	0
Random search	75	24	12	65	26	22	4
Spiral search	85	21	7	57	27	13	2

	Mean time [s]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	0.81	1.32	4.02	0.87	1.51	1.57	4.80
Straight down	0.67	-	-	0.88	1.08	-	-
Random search	1.19	4.00	3.82	2.52	2.72	4.86	5.57
Spiral search	1.00	5.27	6.16	2.35	2.06	4.84	5.47

Table B.2: Comparison of the RL agent and the conventional methods against the initial pose disturbances. The results are shown for the Electronic Part 3.

	Success rate [%]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	**100**	**98**	**84**	**100**	**99**	**98**	**79**
Straight down	96	0	0	40	19	0	0
Random search	85	28	17	53	24	15	5
Spiral search	96	25	9	58	22	15	1

	Mean time [s]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	0.81	1.32	4.02	0.87	1.51	1.57	4.80
Straight down	0.79	-	-	1.04	1.39	-	-
Random search	1.13	2.58	4.16	2.58	3.17	3.99	6.11
Spiral search	1.52	3.47	3.90	3.39	3.99	5.16	6.51

Table B.3: Comparison of the RL agent and the conventional methods against the initial pose disturbances. The results are shown for the Electronic Part 4.

	Success rate [%]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	**100**	**94**	**57**	**100**	**92**	**96**	**56**
Straight down	64	0	0	24	5	0	0
Random search	63	19	10	34	17	8	3
Spiral search	92	21	9	32	19	6	2

	Mean time [s]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	0.81	2.32	5.37	1.07	2.57	2.48	6.06
Straight down	1.22	-	-	1.00	1.18	-	-
Random search	1.41	2.61	4.19	2.88	3.19	5.77	3.54
Spiral search	1.66	3.84	4.76	2.68	2.96	4.97	8.46

Table B.4: Comparison of the RL agent and the conventional methods against the initial pose disturbances. The results are shown for the Electronic Part 5.

	Success rate [%]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	**100**	**90**	**55**	**100**	**95**	**84**	**87**
Straight down	90	0	0	2	0	0	0
Random search	70	20	5	57	33	16	2
Spiral search	70	23	6	56	24	15	2

	Mean time [s]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	0.68	2.63	6.88	0.78	2.07	4.38	2.86
Straight down	0.95	-	-	1.99	-	-	-
Random search	0.92	3.59	3.91	2.25	2.50	2.77	2.62
Spiral search	1.13	5.11	4.13	2.73	3.10	5.51	6.88

Table B.5: Comparison of the RL agent and the conventional methods against the initial pose disturbances. The results are shown for the Electronic Part 6.

	Success rate [%]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	**100**	**99**	**66**	**100**	**100**	**100**	**75**
Straight down	90	0	0	2	0	0	0
Random search	89	22	9	47	20	11	2
Spiral search	99	25	10	52	24	10	4

	Mean time [s]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	0.78	1.91	6.39	0.98	1.71	1.96	5.76
Straight down	0.95	-	-	1.99	-	-	-
Random search	1.16	4.01	7.15	2.61	2.79	4.25	6.51
Spiral search	0.84	5.38	4.92	3.43	2.88	5.75	3.21

Table B.6: Comparison of the RL agent and the conventional methods against the initial pose disturbances. The results are shown for the Electronic Part 7.

	Success rate [%]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	**100**	**98**	**71**	**100**	**97**	**95**	**68**
Straight down	90	0	0	2	0	0	0
Random search	58	21	8	22	14	9	2
Spiral search	47	20	9	15	17	5	0

	Mean time [s]						
	Test case 1	Test case 2	Test case 3	Test case 4	Test case 5	Test case 6	Test case 7
RL Agent	0.85	2.37	6.32	1.04	1.91	2.41	6.87
Straight down	0.95	-	-	1.99	-	-	-
Random search	1.39	3.01	2.38	1.99	1.53	4.25	4.88
Spiral search	2.12	5.08	4.77	2.95	2.15	5.33	-

B.2 First Test Scenario: Robustness - Additional Data

In order to further evaluate the robustness to background perturbation of the MVDeepMDP algorithm, an additional experiment was conducted. In this study, the algorithms were trained using Electronic Part 2 and PCB Layout 1 (refer to Figure 3.7a). Subsequently, the trained models were commanded to insert Electronic Part 2 into both PCBA Layout 1 (refer to Figure 4.2) and PCB Layout 3 (refer to Figure 3.7c). In contrast to the previous experiment discussed in Section 4.3.1, both PCBs in this experiment share the same laminate color but have different electrical connections layout. The results are presented in Table B.7. Most of the methods successfully handled the introduced background perturbations, with only DrQ, PAD, and SAC failing to achieve a 100% success rate. This suggests that transferring the learned policy to a PCB with a similar laminate color is less challenging than transferring it to a PCB with a different laminate color. Nevertheless, this experiment further demonstrated the robustness of the MVDeepMDP algorithm.

Table B.7: Comparison of benchmark methods and MVDeepMDP to handle background perturbations. The presented results are for the Electronic Part 2. For each method, the success rate and mean time are shown. Most of the methods successfully handled the introduced background perturbations, with only DrQ, PAD, and SAC failing to achieve a 100% success rate.

	Success rate [%]		**Mean time [s]**	
	PCBA Layout 1	PCB Layout 3	PCBA Layout 1	PCB Layout 3
SAC	6	14	8.18	7.15
DrQ	45	98	8.17	3.31
SVEA	**100**	**100**	**1.80**	**1.73**
PAD	**100**	5	2.14	15.62
DeepMDP	**100**	**100**	1.84	3.39
DBC	**100**	**100**	1.81	2.42
MVDeepMDP	**100**	**100**	1.85	2.69

B.3 t-SNE Visualization of Unimodal Latent Representations

In this section, the additional t-SNE visualization of the unimodal latent representations is presented.

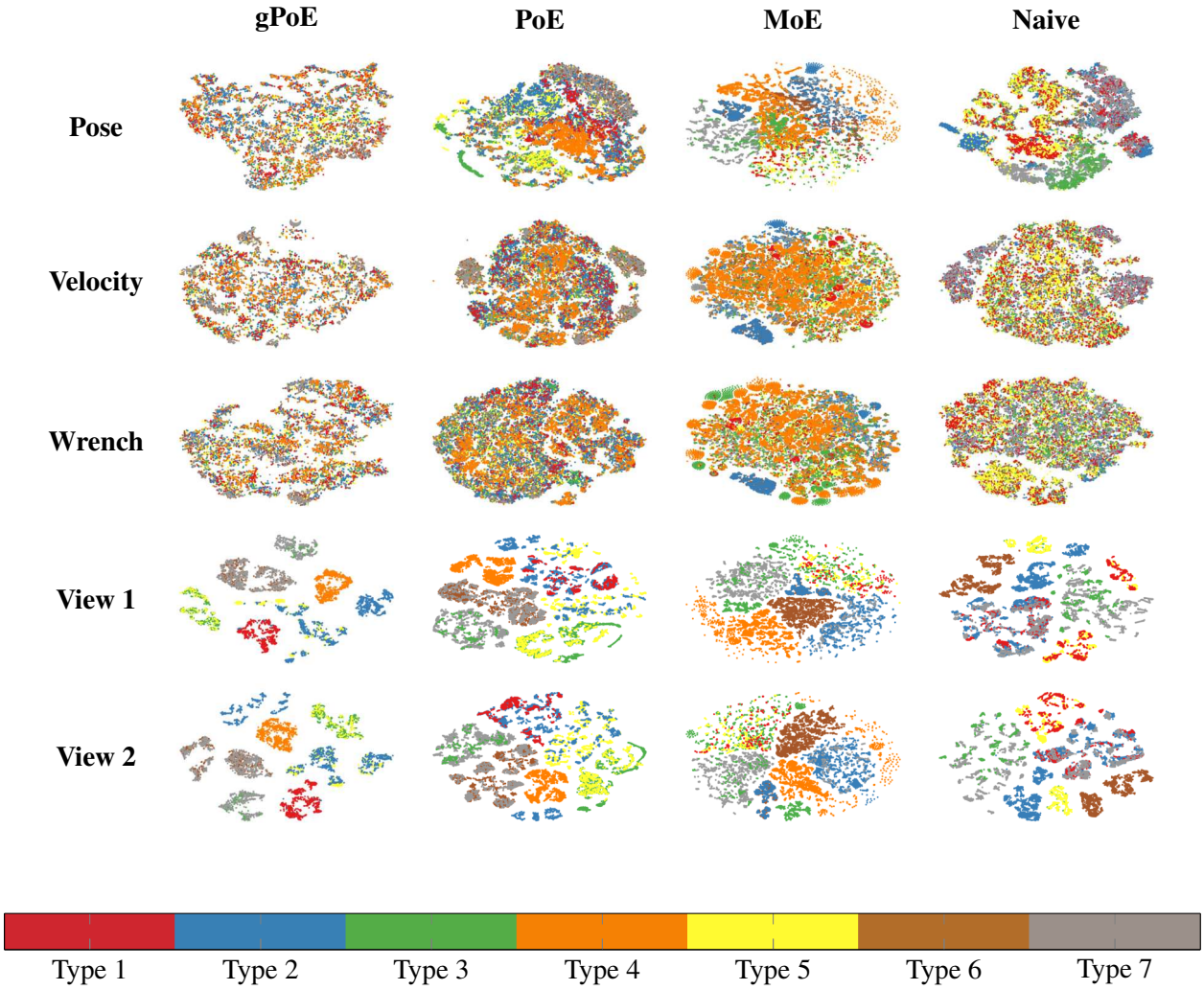


Figure B.1: t-SNE visualization of the latent space for unimodal encoders.

Bibliography

[1] Kuk-Hyun Ahn, Minwoo Na, and Jae-Bok Song. Robotic assembly strategy via reinforcement learning based on force and visual information. *Robotics and Autonomous Systems*, 164:104399, 2023.

[2] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep Variational Information Bottleneck. In *5th International Conference on Learning Representations*, 2017.

[3] Amy Zhang and Rowan Thomas McAllister and Roberto Calandra and Yarin Gal and Sergey Levine. Learning Invariant Representations for Reinforcement Learning without Reconstruction. In *9th International Conference on Learning Representations*, 2021.

[4] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *CoRR*, abs/1607.06450, 2016.

[6] Grzegorz Bartyzel. Multimodal Variational DeepMDP: An Efficient Approach for Industrial Assembly in High-Mix, Low-Volume Production. *IEEE Robotics and Automation Letters*, 9(12):11297–11304, 2024.

[7] Richard Bellman. On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952.

[8] Richard Bellman. A Markovian Decision Process. *Indiana University Mathematics Journal*, 6:679–684, 1957.

[9] Cristian Camilo Beltran-Hernandez, Damien Petit, Ixchel Georgina Ramirez-Alpizar, Takayuki Nishi, Shinichi Kikuchi, Takamitsu Matsubara, and Kensuke Harada. Learning Force Control for Contact-Rich Manipulation Tasks With Rigid Position-Controlled Robots. *IEEE Robotics and Automation Letters*, 5(4):5709–5716, 2020.

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *CoRR*, abs/1606.01540, 2016.

[11] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Proceedings of Robotics: Science and Systems XIX*, 2023.

[12] Yanshuai Cao and David J Fleet. Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions. *Modern Nonparametrics 3: Automating the Learning Pipeline Workshop at NIPS*, 2014.

[13] Carles Gelada and Saurabh Kumar and Jacob Buckman and Ofir Nachum and Marc G. Bellemare. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2170–2179. PMLR, 09–15 Jun 2019.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1597–1607. PMLR, 2020.

[15] John J. Craig and Marc H. Raibert. A systematic method of hybrid position/force control of a manipulator. In *COMPSAC 79. Proceedings. Computer Software and The IEEE Computer Society's Third International Applications Conference, 1979.*, pages 446–451, 1979.

[16] Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning*, pages 465–472, 2011.

[17] Denis Yarats and Ilya Kostrikov and Rob Fergus. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. In *9th International Conference on Learning Representations*, 2021.

[18] Andreas Doerr, Christian Daniel, Martin Schiegg, Duy Nguyen-Tuong, Stefan Schaal, Marc Toussaint, and Sebastian Trimpe. Probabilistic Recurrent State-Space Models. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1279–1288. PMLR, 2018.

[19] Jiameng Fan and Wenchao Li. DRIBO: Robust Deep Reinforcement Learning via Multi-View Information Bottleneck. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu,

and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 6074–6102. PMLR, 2022.

[20] Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and Regularization in DQN. *CoRR*, abs/1810.00123, 2018.

[21] Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. Learning Robust Representations via Multi-View Information Bottleneck. In *8th International Conference on Learning Representations*, 2020.

[22] Yanqiong Fei and Xifang Zhao. An Assembly Process Modeling and Analysis for Robotic Multiple Peg-in-hole. *Journal of Intelligent and Robotic Systems*, 36(2):175–189, Feb 2003.

[23] Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi S. Jaakkola. Learning Task Informed Abstractions. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 3480–3491. PMLR, 2021.

[24] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1582–1591. PMLR, 2018.

[25] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael J. Black, and Bernhard Schölkopf. From Variational to Deterministic Autoencoders. In *8th International Conference on Learning Representations*, 2020.

[26] Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal Difference Variational Auto-Encoder. In *7th International Conference on Learning Representations*, 2019.

[27] Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aäron van den Oord. Shaping Belief States with Generative Environment Models for RL. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 13475–13487, 2019.

[28] Grzegorz Bartyzel and Wojciech Półchłopek and Dominik Rzepka. Reinforcement Learning With Stereo-View Observation for Robust Electronic Component Robotic Insertion. *Journal of Intelligent & Robotic Systems*, 109(3):57, Oct 2023.

[29] David Ha and Jürgen Schmidhuber. World Models. *CoRR*, abs/1803.10122, 2018.

[30] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to Walk Via Deep Reinforcement Learning. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, *Proceedings of Robotics: Science and Systems XV*, 2019.

[31] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications, 2018.

[32] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *8th International Conference on Learning Representations*, 2020.

[33] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2555–2565. PMLR, 2019.

[34] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models. In *9th International Conference on Learning Representations*, 2021.

[35] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models. *CoRR*, abs/2301.04104, 2023.

[36] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-Supervised Policy Adaptation during Deployment. In *9th International Conference on Learning Representations*, 2021.

[37] Nicklas Hansen and Xiaolong Wang. Generalization in Reinforcement Learning by Soft Data Augmentation. In *Proceedings of 2021 IEEE International Conference on Robotics and Automation*, pages 13611–13617. IEEE, 2021.

[38] Olivier J. Hénaff. Data-Efficient Image Recognition with Contrastive Predictive Coding. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 4182–4192. PMLR, 2020.

[39] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *5th International Conference on Learning Representations*, 2017.

[40] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[41] Neville Hogan. Impedance Control: An Approach to Manipulation. In *1984 American Control Conference*, pages 304–313, 1984.

[42] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed Prioritized Experience Replay. In *6th International Conference on Learning Representations*, 2018.

[43] Zhimin Hou, Haiming Dong, Kuangen Zhang, Quan Gao, Ken Chen, and Jing Xu. Knowledge-Driven Deep Deterministic Policy Gradient for Robotic Multiple Peg-in-Hole Assembly Tasks. In *Proceedings of 2018 IEEE International Conference on Robotics and Biomimetics*, pages 256–261, 2018.

[44] Zhimin Hou, Zhihu Li, Chenwei Hsu, Kuangen Zhang, and Jing Xu. Fuzzy Logic-Driven Variable Time-Scale Prediction-Based Reinforcement Learning for Robotic Multiple Peg-in-Hole Assembly . *IEEE Transactions on Automation Science and Engineering*, 19(1):218–229, 2022.

[45] Shouren Huang, Kenichi Murakami, Yuji Yamakawa, Taku Senoo, and Masatoshi Ishikawa. Fast peg-and-hole alignment using visual compliance. In *Proceedings of 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 286–292, 2013.

[46] Tadanobu Inoue, Giovanni De Magistris, Asim Munawar, Tsuyoshi Yokoya, and Ryuki Tachibana. Deep reinforcement learning for high precision assembly tasks. In *Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 819–825, 2017.

[47] Iñigo Iturrate, Etienne Roberge, Esben Hallundbæk Østergaard, Vincent Duchaine, and Thiusius Rajeeth Savarimuthu. Improving the Generalizability of Robot Assembly Tasks Learned from Demonstration via CNN-based Segmentation. In *Proceedings of 2019 IEEE 15th International Conference on Automation Science and Engineering*, pages 553–560, 2019.

[48] M. Jinno, F. Ozaki, T. Yoshimi, K. Tatsuno, M. Takahashi, M. Kanda, Y. Tamada, and S. Nagataki. Development of a force controlled robot for grinding, chamfering and polishing. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 2, pages 1455–1460 vol.2, 1995.

[49] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual Reinforcement Learning for Robot Control. In *Proceedings of 2019 IEEE International Conference on Robotics and Automation*, pages 6023–6029, 2019.

[50] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, Nov 1999.

[51] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[52] Oussama Khatib. Inertial Properties in Robotic Manipulation: An Object-Level Framework. *The International Journal of Robotics Research*, 14(1):19–36, 1995.

[53] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations*, 2015.

[54] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised Learning with Deep Generative Models. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589, 2014.

[55] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations*, 2014.

[56] Oliver Kroemer, Scott Niekum, and George Konidaris. A Review of Robot Learning for Manipulation: Challenges and Representations and and Algorithms. *Journal of Machine Learning Research*, 22(30):1–82, 2021.

[57] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Proceedings of 2010 International Joint Conference on Neural Networks*, pages 1–8, 2010.

[58] Sascha Lange, Martin Riedmiller, and Arne Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. In *Proceedings of 2012 International Joint Conference on Neural Networks*, pages 1–8, 2012.

[59] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement Learning with Augmented Data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33*, 2020.

[60] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 5639–5650. PMLR, 2020.

[61] Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33*, 2020.

[62] Michelle A. Lee, Yuke Zhu, Peter Zachares, Matthew Tan, Krishnan Srinivasan, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making Sense of Vision and Touch: Learning Multimodal Representations for Contact-Rich Tasks. *IEEE Transactions on Robotics*, 36(3):582–596, 2020.

[63] Mihee Lee and Vladimir Pavlovic. Private-Shared Disentangled Multimodal VAE for Learning of Latent Representations. In *Proceedings of 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1692–1700. Computer Vision Foundation / IEEE, 2021.

[64] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17:39:1–39:40, 2016.

[65] Rui Li, Robert Platt Jr., Wenzhen Yuan, Andreas ten Pas, Nathan Roscup, Mandayam A. Srinivasan, and Edward H. Adelson. Localization and manipulation of small parts using GelSight tactile sensing. In *Proceedings of 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3988–3993, 2014.

[66] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for Distributed Reinforcement Learning.

In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3053–3062. PMLR, 10–15 Jul 2018.

[67] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2016.

[68] Zhenyu Liu, Ke Wang, Daxin Liu, Qide Wang, and Jianrong Tan. A Motion Planning Method for Visual Servoing Using Deep Reinforcement Learning in Autonomous Robotic Assembly. *IEEE/ASME Transactions on Mechatronics*, 28(6):3513–3524, 2023.

[69] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations*, 2019.

[70] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning. In *Proceedings of 2024 IEEE International Conference on Robotics and Automation*, pages 16961–16969, 2024.

[71] Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Aparicio Ojea, Alice M. Agogino, Aviv Tamar, and Pieter Abbeel. Reinforcement Learning on Variable Impedance Controller for High-Precision Robotic Assembly. In *Proceedings of 2019 IEEE International Conference on Robotics and Automation*, pages 3080–3087, 2019.

[72] Jianlan Luo, Oleg Sushkov, Rugile Pevceviciute, Wenzhao Lian, Chang Su, Mel Vecerík, Ning Ye, Stefan Schaal, and Jonathan Scholz. Robust Multi-Modal Policies for Industrial Assembly via Reinforcement Learning and Demonstrations: A Large-Scale Study. In Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh, editors, *Proceedings of Robotics: Science and Systems XVII*, 2021.

[73] Yanqin Ma, De Xu, and Fangbo Qin. Efficient Insertion Control for Precision Assembly Based on Demonstration Learning and Reinforcement Learning. *IEEE Transactions on Industrial Informatics*, 17(7):4492–4502, 2021.

[74] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[75] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. In *Proceedings of Robotics: Science and Systems XIII*, 2017.

[76] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

[77] Jeffrey Mahler, Florian T. Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based net-

work of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *Proceedings of 2016 IEEE International Conference on Robotics and Automation*, pages 1957–1964. IEEE, 2016.

[78] P. Marbach and J.N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, 2001.

[79] Roberto Martín-Martín, Michelle A. Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable Impedance Control in End-Effector Space: An Action Space for Reinforcement Learning in Contact-Rich Tasks. In *Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1010–1017, 2019.

[80] Jeremy A. Marvel, Roger Bostelman, and Joe Falco. Multi-Robot Assembly Strategies and Metrics. *ACM Comput. Surv.*, 51(1), jan 2018.

[81] Matthew T. Mason. Compliance and Force Control for Computer Controlled Manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):418–432, 1981.

[82] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb 2015.

[83] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention Bottlenecks for Multimodal Fusion. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34*, pages 14200–14213, 2021.

[84] Nicklas Hansen and Hao Su and Xiaolong Wang. Stabilizing Deep Q-Learning with ConvNets and Vision Transformers under Data Augmentation. In M'A. Ranzato, A. Beygelzimer, Y.N. Dauphin, P. Liang, and J.W. Vaughan, editors, *Advances in Neural Information Processing Systems 34*, pages 3680–3693, 2021.

[85] Gaurav Pandey and Ambedkar Dukkipati. Variational methods for conditional multimodal deep learning. In *Proceedings of 2017 International Joint Conference on Neural Networks*, pages 308–315. IEEE, 2017.

[86] Hyeonjun Park, Ji-Hun Bae, Jae-Han Park, Moon-Hong Baeg, and Jaeheung Park. Intuitive peg-in-hole assembly strategy with a compliant manipulator. In *Proceedings of 2013 IEEE International Symposium on Robotics*, pages 1–5, 2013.

[87] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner,

Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, volume 32. Curran Associates, Inc., 2019.

[88] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-Modal Fusion Transformer for End-to-End Autonomous Driving. In *Proceedings of 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087. Computer Vision Foundation / IEEE, 2021.

[89] Marc H. Raibert and John J. Craig. Hybrid Position/Force Control of Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 103(2):126–133, 06 1981.

[90] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for Activation Functions. 2018.

[91] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge and MA and USA, 2018.

[92] Korntham Sathirakul and Robert H. Sturges. Jamming conditions for multiple peg-in-hole assemblies. *Robotica*, 16(3):329–345, 1998.

[93] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations*, 2016.

[94] Stefan Scherzinger, Arne Roennau, and Rüdiger Dillmann. Forward Dynamics Compliance Control (FDCC): A new approach to cartesian compliance for robotic manipulators. In *Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4568–4575, 2017.

[95] Gerrit Schoettler, Ashvin Nair, Jianlan Luo, Shikhar Bahl, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5548–5555, 2020.

[96] Gerrit Schoettler, Ashvin Nair, Juan Aparicio Ojea, Sergey Levine, and Eugen Solowjow. Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks. In *Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 9728–9735, 2020.

[97] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, Dec 2020.

[98] Homayoun Seraji. Adaptive admittance control: an approach to explicit force control in compliant motion. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2705–2712 vol.4, 1994.

[99] Yuge Shi, Siddharth Narayanaswamy, Brooks Paige, and Philip H. S. Torr. Variational Mixture-of-Experts Autoencoders for Multi-Modal Deep Generative Models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15692–15703, 2019.

[100] Yunlei Shi, Zhaopeng Chen, Yansong Wu, Dimitri Henkel, Sebastian Riedel, Hongxu Liu, Qian Feng, and Jianwei Zhang. Combining Learning from Demonstration with Learning by Exploration to Facilitate Contact-Rich Tasks. In *Proceedings of 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1062–1069, 2021.

[101] Satinder P. Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1):123–158, Mar 1996.

[102] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning Structured Output Representation using Deep Conditional Generative Models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, volume 28. Curran Associates, Inc., 2015.

[103] Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational Overfitting in Reinforcement Learning. In *8th International Conference on Learning Representations*, 2020.

[104] Oren Spector and Dotan Di Castro. InsertionNet - A Scalable Solution for Insertion. *IEEE Robotics and Automation Letters*, 6(3):5509–5516, 2021.

[105] Oren Spector, Vladimir Tchuiev, and Dotan Di Castro. InsertionNet 2.0: Minimal Contact Multi-Step Insertion Using Multimodal Multiview Sensory Input. In *Proceedings of 2022 IEEE International Conference on Robotics and Automation)*, pages 6330–6336, 2022.

[106] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling Representation Learning from Reinforcement Learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 9870–9879. PMLR, 2021.

[107] Thomas M. Sutter, Imant Daunhawer, and Julia E. Vogt. Multimodal Generative Learning Utilizing Jensen-Shannon-Divergence. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33*, 2020.

[108] Thomas M. Sutter, Imant Daunhawer, and Julia E. Vogt. Generalized Multimodal ELBO. In *9th International Conference on Learning Representations*, 2021.

[109] Richard S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3(1):9–44, Aug 1988.

[110] Richard S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In Bruce Porter and Raymond Mooney, editors, *Machine Learning Proceedings 1990*, pages 216–224. Morgan Kaufmann, San Francisco (CA), 1990.

[111] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, volume 12. MIT Press, 1999.

[112] Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Joint Multimodal Learning with Deep Generative Models. In *5th International Conference on Learning Representations*, 2017.

[113] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

[114] Joshua C. Triyonoputro, Weiwei Wan, and Kensuke Harada. Quickly Inserting Pegs into Uncertain Holes using Multi-view Images and Deep Network Trained on Synthetic Data. In *Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5792–5799, 2019.

[115] Tuomas Haarnoja and Aurick Zhou and Pieter Abbeel and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1861–1870. PMLR, 2018.

[116] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *CoRR*, abs/1807.03748, 2018.

[117] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, volume 30. Curran Associates, Inc., 2017.

[118] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothörl, Todd Hester, and Jon Scholz. A Practical Approach to Insertion with Variable Socket Position Using Deep Reinforcement Learning. In *Proceedings of 2019 IEEE International Conference on Robotics and Automation*, pages 754–760, 2019.

[119] Chen Wang, Shaoxiong Wang, Branden Romero, Filipe Veiga, and Edward Adelson. SwingBot: Learning Physical Features from In-hand Tactile Exploration for Dynamic Swing-up Manipulation. In *Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5633–5640, 2020.

[120] Kaimeng Wang, Yu Zhao, and Ichiro Sakuma. Learning Robotic Insertion Tasks From Human Demonstration. *IEEE Robotics and Automation Letters*, 8(9):5815–5822, 2023.

[121] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[122] Daniel E. Whitney. Force Feedback Control of Manipulator Fine Motions. *Journal of Dynamic Systems, Measurement, and Control*, 99(2):91–97, 06 1977.

[123] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

[124] Zheng Wu, Yichen Xie, Wenzhao Lian, Changhao Wang, Yanjiang Guo, Jianyu Chen, Stefan Schaal, and Masayoshi Tomizuka. Zero-Shot Policy Transfer with Disentangled Task Representation of Meta-Reinforcement Learning. In *Proceedings of 2023 IEEE International Conference on Robotics and Automation*, pages 7169–7175, 2023.

[125] Wu, Mike and Goodman, Noah. Multimodal Generative Models for Scalable Weakly-Supervised Learning. In S. Bengio, H.M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5580–5590, 2018.

[126] Liang Xie, Hongxiang Yu, Yinghao Zhao, Haodong Zhang, Zhongxiang Zhou, Minhang Wang, Yue Wang, and Rong Xiong. Learning to Fill the Seam by Vision: Sub-millimeter Peg-in-hole on Unseen Shapes in Real World. In *Proceedings of 2022 IEEE International Conference on Robotics and Automation*, pages 2982–2988, 2022.

[127] Jing Xu, Zhimin Hou, Wei Wang, Bohao Xu, Kuangen Zhang, and Ken Chen. Feedback Deep Deterministic Policy Gradient With Fuzzy Reward for Robotic Multiple Peg-in-Hole Assembly Tasks. *IEEE Transactions on Industrial Informatics*, 15(3):1658–1667, 2019.

[128] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2Image: Conditional Image Generation from Visual Attributes. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 776–791. Springer, 2016.

[129] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement Learning with Prototypical Representations. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 11920–11931. PMLR, 2021.

[130] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning. In *10th International Conference on Learning Representations*, 2022.

[131] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving Sample Efficiency in Model-Free Reinforcement Learning from Images. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10674–10681, May 2021.

[132] Cunjun Yu, Zhongang Cai, Hung Pham, and Quang-Cuong Pham. Siamese Convolutional Neural Network for Sub-millimeter-accurate Camera Pose Estimation and Visual Servoing. In *Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 935–941, 2019.

[133] Chengjie Yuan, Yunlei Shi, Qian Feng, Chunyang Chang, Michael Liu, Zhaopeng Chen, Alois Christian Knoll, and Jianwei Zhang. Sim-to-Real Transfer of Robotic Assembly with Visual Inputs Using CycleGAN and Force Control. In *Proceedings of 2022 IEEE International Conference on Robotics and Biomimetics*, pages 1426–1432, 2022.

[134] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A Dissection of Overfitting and Generalization in Continuous Reinforcement Learning. *CoRR*, abs/1806.07937, 2018.

[135] Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural Environment Benchmarks for Reinforcement Learning. *CoRR*, abs/1811.06032, 2018.

[136] Xiang Zhang, Changhao Wang, Lingfeng Sun, Zheng Wu, Xinghao Zhu, and Masayoshi Tomizuka. Efficient Sim-to-real Transfer of Contact-Rich Manipulation Skills with Online Admittance Residual Learning. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229, pages 1621–1639. PMLR, 06–09 Nov 2023.

[137] Zihao Zhao, Anusha Nagabandi, Kate Rakelly, Chelsea Finn, and Sergey Levine. MELD: Meta-Reinforcement Learning from Images via Latent State Models. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *Proceedings of The 4th Conference on Robot Learning*, volume 155, pages 1246–1261. PMLR, 2020.

[138] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229, pages 2165–2183. PMLR, 2023.

[139] K.J Åström. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

[140] Íñigo Elguea-Aguinaco, Antonio Serrano-Muñoz, Dimitrios Chrysostomou, Ibai Inziarte-Hidalgo, Simon Bøgh, and Nestor Arana-Arexolaleiba. A review on reinforcement learning for contact-rich robotic manipulation tasks. *Robotics and Computer-Integrated Manufacturing*, 81:102517, 2023.