TECHNICAL UNIVERSITY OF MUNICH

SCHOOL OF ENGINEERING AND DESIGN
DEPARTMENT OF MECHANICAL ENGINEERING

Chair of Sensor Based Robotic Systems and Intelligent Assistance Systems

Master's thesis in
Mechatronics, Robotics and Biomechanical Engineering M.Sc.

Guided Reinforcement Learning with Vision Feedback

Author:	Baran Özer
Examiner:	Prof. Dr. Alin Albu-Schäffer
Supervisors:	Dr. Abhishek Padalkar, Dr. João Silvério
Submission Date:	23.04.2026

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 23.04.2026 Baran Özer

Acknowledgments

I would like to sincerely thank the German Aerospace Center (DLR) for providing me with the opportunity to conduct this research. The resources and environment provided have been crucial in allowing me to explore and advance my work.

I am especially grateful to Abhishek Padalkar and João Silvério for their guidance, support, and mentorship throughout this project. Their expertise and encouragement were essential in overcoming both the theoretical and practical challenges faced during this research. I am also grateful to Antonin Raffin, whose expertise and willingness to help were a constant support throughout this thesis.

Abstract

This thesis investigates the combined use of probabilistic trajectory guidance and visual representation learning for contact-rich robotic assembly tasks employing Reinforcement Learning (RL). The aim is to improve the robustness and efficiency of RL by integrating learnt visual cues into Kernelized Guided Reinforcement Learning (KGRL), a framework that combines RL with Kernelized Movement Primitives (KMP). KMP encodes and generalizes motion trajectories into a probabilistic reference distribution, while KGRL employs these trajectory priors to guide policy learning via null space actions, whose magnitude is governed by the demonstration variance, enabling uncertainty-aware exploration. Task-specific Variational Autoencoders (VAEs) are designed to produce low-dimensional visual latent representations that improve the state representation of the reinforcement learning agent in varying initial conditions and geometric uncertainty.

This thesis introduces a Vision-augmented Kernelized Guided Reinforcement Learning (Vision-KGRL) framework, evaluated on three representative assembly tasks: Peg Insertion, Gear Meshing, and Nut Threading within the NVIDIA Isaac Lab simulation environment, using the Factory and Forge benchmark environments. The research findings indicate that KGRL can be seamlessly augmented with latent visual features while preserving the faster convergence over unconstrained RL reported in prior work, with this trend becoming more pronounced in higher-dimensional action spaces.

A thorough series of ablation studies methodically examines the individual and collective impacts of visual features, force feedback, KMP-based trajectory guidance, and uncertainty-aware exploration through null space actions. The evaluation compares standard reinforcement learning and KGRL under varying observation modalities, including proprioceptive-only, force-augmented, vision-augmented, and combined force and vision settings, while also examining the impact of action space through a comparison of 4D (position and yaw) and full 6D Cartesian control, across the aforementioned assembly tasks. In addition to improved success rates and convergence behavior, the experiments show that KMP-guided policies have significantly lower interaction forces on the environment, implying smoother and more stable task execution. This property makes the approach especially promising for use on real robotic systems, where limiting contact forces is critical for safe operation.

Overall, the thesis illustrates that Vision-KGRL is an efficient approach for contact-rich robotic assembly. It shows that combining learned visual representations with trajectory priors markedly improves learning stability and effectiveness in complex manipulation tasks.

Keywords: Guided Reinforcement Learning; Movement Primitives; Learning from Demonstration; Visual Representation Learning; Variational Autoencoders; Reinforcement Learning; Contact-Rich Assembly; Robot Manipulation.

Contents

1. Introduction

Robotic assembly is a key part of industrial automation and still remains a challenging problem in robotics. Contact-rich tasks such as peg insertion, gear meshing, and nut threading require precise alignment, stable force regulation, and accurate motion control under tight geometric tolerances. Small errors in perception or modeling can cause jamming, too much contact force, or task failure. Impedance control and hybrid position-force control are two classical methods that provide stable interaction in structured environments by directly controlling motion and contact forces [1, 2]. Nonetheless, these techniques rely on precise modeling and meticulously adjusted control parameters, which constrains their resilience in the presence of geometric uncertainties and environmental changes [2, 3].

Reinforcement learning (RL) offers a data-driven approach by deriving policies directly from interactions instead of depending on explicit analytical models of contact dynamics [4]. In robotic control, initial actor-critic and model-based methodologies illustrated the feasibility of learning continuous control policies directly from experience [4]. Subsequent work extended deep RL to contact-rich assembly tasks, incorporating proprioceptive and force-based observations to learn insertion behaviors without explicit contact modeling [5, 6]. Off-policy algorithms such as Soft Actor-Critic (SAC) further improved stability and sample efficiency in continuous control through entropy regularization and replay mechanisms [7].

Even with these improvements, using pure RL for precision assembly is still challenging. Contact-rich manipulation involves discontinuous dynamics at contact transitions, complex force interactions, and safety-critical exploration. In practice, collecting data is expensive, designing reward functions is not trivial, and unconstrained exploration can lead to unstable or unsafe behaviors [6, 8]. These challenges often lead to slow convergence and high variance in performance when policies are learned without structural guidance.

Imitation-guided reinforcement learning uses structural priors based on demonstrations to make RL exploration safer and more efficient. Kernelized Guided Reinforcement Learning (KGRL) [9], built upon Null-Space Kernelized Movement Primitives (NS-KMP) [10, 11], constrains policy learning using uncertainty-informed movement primitives derived from demonstrations. RL does not freely explore the entire action space; instead, it operates within the soft null space of a demonstration-derived trajectory distribution. The variance in the demonstrations influences exploration and facilitates state-dependent modulation of the reference trajectory. This probabilistic guidance accelerates and stabilizes policy learning in contact-rich manipulation compared to unconstrained RL.

Despite these advantages, current KGRL formulations mostly use proprioceptive and force-based state representations. Visual perception, which is essential for pre-contact alignment and geometric reasoning, is not explicitly integrated into the demonstration-constrained learning process.

At the same time, vision-based RL techniques generally depend on high-dimensional RGB images, which offer abundant perceptual information but are computationally intensive and susceptible to fluctuations in object appearance, lighting, or background, thereby complicating generalization to real-world scenarios [12].

Moreover, vision-driven RL frequently lacks structured guidance from demonstrations, requiring substantial data collection and exploration in contact-rich settings. This thesis mitigates this limitation by incorporating learned compact visual representations into the KGRL framework and methodically examining the impact of various sensory modalities on policy training. The primary objective is to integrate uncertainty-aware guidance derived from demonstrations with perception-driven state representations, while systematically comparing proprioceptive-only, force-augmented, and vision-augmented observation spaces. The aim is to improve robustness, data efficiency, and task performance in contact-rich assembly by integrating visual information into a systematic imitation-guided RL framework.

The NVIDIA Isaac Lab simulation framework [13] was used for all of the experiments. It is a GPU-accelerated robotics simulation platform made for large-scale RL. The assembly tasks examined in this thesis are taken from the Isaac Lab Factory and Forge benchmark environments [14, 15], which provide high-fidelity contact-rich manipulation scenarios including PegInsert, GearMesh, and NutThread. The Factory environments provide proprioceptive state observations, while the Forge environments enhance the observation space by integrating force measurements, allowing evaluation of force-augmented policies.

To encode visual information from these simulation environments, three Variational Autoencoders (VAEs) with a latent dimension of 16 were trained for the PegInsert, GearMesh, and NutThread tasks. The learned latent representations were integrated into the policy input within both standard SAC and KGRL frameworks.

A systematic empirical evaluation was performed in simulation across the three tasks and multiple random seeds. The study compares standard SAC and KGRL across observation modalities and action spaces, encompassing proprioceptive-only observations (Factory), proprioceptive and force-based observations (Forge), and their corresponding vision-augmented extensions, as well as reduced 4D control (position and yaw) and full 6D Cartesian control. The evaluation assesses learning dynamics, convergence behavior, safety, and task success rates.

The results of the study indicate that incorporating visual perception into a demonstration-guided RL framework enhances performance and robustness in contact-rich robotic assembly tasks. In the opposite direction, they also show that the guidance provided by the demonstrations improves training efficiency of the visuomotor policies, particularly in higher dimensional action spaces. Together, these observations suggest that visual perception and trajectory guidance that takes uncertainty into account are two important parts of safe and efficient reinforcement learning for precision assembly

The primary contributions of this thesis are summarized as follows:

- An extension of Kernelized Guided Reinforcement Learning (KGRL) with multimodal observations for contact-rich manipulation. The framework incorporates proprioception, force sensing, and compact visual latent representations obtained from task-specific

Variational Autoencoders into an RL policy constrained by demonstration-derived movement primitives.

- An empirical validation of enhanced safety[1] and robustness, illustrating that demonstration-guided KGRL exhibits lower variance across random seeds and reduced contact forces compared to standard RL baselines across all evaluated tasks, environments, and action space configurations.
- A systematic analysis of the dimensionality of the action space, comparing reduced 4D control (position and yaw) with full 6D Cartesian control in terms of learning performance, stability, and safety. The results indicate that KGRL achieves a significantly higher performance than unconstrained RL in the 6D setting.
- A systematic multimodal ablation study that assesses the individual and combined effects of visual and force observations in four observation configurations: proprioceptive-only, force-augmented, vision-augmented, and combined force and vision settings.
- A comprehensive large-scale evaluation in NVIDIA Isaac Lab, which includes systematic experiments across Factory and Forge simulation environments on three assembly tasks of increasing difficulty: PegInsert, GearMesh, and NutThread.

[1] Throughout this thesis, the term *safety* refers to reduced contact forces during RL training and evaluation. Higher contact forces risk damaging the robot or the assembled components, and policies that produce lower contact forces are therefore considered safer.

2. Related Work

2.1. Classical Control in Contact-rich Robotic Assembly

Early robotic assembly research used model-based control strategies to address contact uncertainties in tasks like peg-in-hole insertion [16]. Passive compliance techniques were used to correct minor misalignments and minimize insertion forces. Although useful in certain situations, these techniques have limited adaptability beyond their initial design.

Force-based control methods, such as impedance control [1] and hybrid position-force control [2], adjust stiffness and force in specific directions to regulate the robot-environment interaction. These methods provide reliable insertion and alignment in structured settings, but they rely heavily on accurate contact geometry and material property predictions. Their performance suffers in the presence of model uncertainty or environmental variations.

Pitchandi et al. [17] and Hou et al. [18] used visual servoing with image feedback to improve alignment. Vision-based pose estimation improves pre-contact alignment by reconstructing geometric attributes from camera data [19]. However, visual-only techniques often require precise calibration and stable environmental conditions.

Multi-modal control methods with vision and force sensing have been proposed to improve robustness [20], [3]. In these systems, vision enables coarse alignment, while force feedback allows for precise adjustments during contact. Despite improved performance, traditional multi-modal approaches continue to rely on predefined control rules and precise system modeling.

Classical control methods provide stability and accuracy in regulated environments, but they lack the adaptability required for dynamic and changing assembly conditions. These constraints drive the evolution of learning-based approaches.

2.2. Learning-Based Methodologies for Contact-Rich Manipulation

2.2.1. Reinforcement Learning

Reinforcement learning (RL) allows control policies to be designed through direct interaction with the environment rather than using explicit analytical models of contact dynamics. Modeling discontinuous contact transitions and complex force interactions is particularly challenging in contact-rich manipulation. RL offers a flexible alternative to traditional control techniques [21].

Preliminary techniques included actor-critic and model-based reinforcement learning algorithms with proprioceptive and force-based state representations [5, 6]. Although these methods showed that contact-aware behaviors can be learned through interaction, they

frequently had low sampling efficiency and sensitivity to sensitivity to reward function design.

Off-policy methodologies, such as Soft Actor-Critic (SAC) [7], improve stability and data efficiency in continuous control environments by using entropy regularization and replay buffers. Pure reinforcement learning in contact-rich tasks often requires extensive exploration and online interaction. In sparse-reward contexts, unregulated exploration can lead to delayed convergence and dangerous actions [8].

These limitations motivate the incorporation of demonstrations to guide exploration and improve data efficiency.

2.2.2. Learning from Demonstration and Movement Primitives

Learning from Demonstration (LfD) enables robots to acquire policies from expert-provided examples [22]. Demonstrations typically consist of state-action trajectories collected during task execution, with the goal of learning a policy that reproduces the demonstrated behavior.

A common approach to policy learning in LfD is behavior cloning, which formulates the problem as supervised learning over observed state-action pairs. A parameterized policy is trained to map states to corresponding expert actions using standard regression or classification objectives.

Despite its simplicity and effectiveness, behavior cloning is restricted by its dependence on the distribution of observed states. Specifically, performance can degrade when the policy confronts states that were not present in the demonstration data, which can lead to cumulative errors under distributional shift [23].

Trajectory-level representations, including Dynamic Movement Primitives (DMP) [24], Probabilistic Movement Primitives (ProMP) [25], and Kernelized Movement Primitives (KMP) [11], cover entire motion trajectories in addition to direct state-action mappings.

DMPs define motions as stable dynamical systems that use learned forcing functions. ProMPs improve this formulation by preserving a probability distribution across trajectory parameters, allowing for conditioning and variability modeling. KMP defines trajectory learning as a kernel-based nonparametric regression problem that allows for smooth interpolation and adaptation to new task constraints while maintaining demonstration statistics.

Movement primitives provide structured representations of demonstrated motions, but do not naturally address exploration or long-term optimization in unseen situations, particularly in contact-rich settings where in-contact behaviors are difficult to capture accurately from demonstrations alone.

2.3. Guided Reinforcement Learning Utilizing Structural Priors

Many approaches employ structural priors in reinforcement learning to enhance safety and sampling efficiency in contact-rich manipulation. These structural priors can take different forms such as engineered controllers [21], simulation-trained policies [12], human

demonstrations and corrections [26, 27, 28], or probabilistic representations obtained from demonstrations [29, 9].

Zhang et al. [21] integrated visual and force feedback into a residual reinforcement learning framework for robotic assembly. The engineered force controller functions as a structural prior, which reduces exploration complexity and enhances training safety by limiting the policy to learn only corrective actions. However, the baseline controller is manually calibrated and may lack generalizability across diverse contact dynamics and material characteristics.

Guo et al. [12] introduced SPARR, which integrates a simulation-trained base policy with a vision-conditioned residual policy trained on a real robot. The base policy provides guided exploration, while the residual policy corrects for sim-to-real discrepancies. Nonetheless, the residual policy is unconstrained by any structure from demonstrations, which could produce unsafe behaviors. Moreover, the framework relies on the base policy achieving a sufficient zero-shot success rate, since residual policy training depends on successful base policy rollouts rather than human demonstrations.

Luo et al. [26] introduced SERL, a software suite designed for sample-efficient real-world robotic reinforcement learning, which integrates demonstration data with off-policy RL to develop contact-rich manipulation policies directly on physical robots utilizing visual observations and sparse rewards. Luo et al. [27] developed HIL-SERL, which integrates human corrections in the training process. Human operators provide demonstrations and intervene to correct failures. Their framework achieves near-perfect success rates in dexterous assembly tasks within one to two hours of practical training. Stranghöner et al. [28] introduced SHaRe-RL, which integrates the human-in-the-loop reinforcement learning paradigm within structured manipulation primitives for industrial connector insertion. Scripted primitives manage repetitive phases, whereas learning focuses on the unpredictable insertion phase, significantly enhancing performance compared to unstructured human-in-the-loop baselines. However, none of these approaches constrain exploration by employing the uncertainty uncertainty structure of the demonstrations.

Davchev et al. [29] introduced residual learning from demonstration (rLfD), integrating dynamic movement primitives (DMPs) with model-free reinforcement learning for contact-rich insertion tasks. A DMP base policy is acquired from demonstrations, while a residual RL policy directly learns corrective actions in task space. They claim that implementing residual corrections directly in task space facilitates local exploratory behavior, such as jiggling, which is especially effective for contact-rich insertion. The framework exhibits enhanced sample efficiency and generalization relative to pure RL baselines. However, exploration is shaped by unstructured Gaussian noise rather than the uncertainty structure of the demonstrations.

Padalkar et al. [9] introduced Kernelized Guided Reinforcement Learning (KGRL), which constrains policy learning by using uncertainty-aware movement primitives obtained from demonstrations. The methodology employs Null-Space Kernelized Movement Primitives (NS-KMP), as presented by Silvério et al. [10]. Reinforcement learning policy in KGRL operates within the null space of a trajectory model obtained from demonstrations. Exploration is shaped by the variance of the demonstrations, which leads to state-dependent modifications of the reference trajectory while adhering to certain constraints. However, the original KGRL

formulation considers only proprioception and interaction forces as the policy observation space, without incorporating any form of visual perception.

Unlike residual reinforcement learning and simulation-based methods, KGRL builds structure at the trajectory level through probabilistic representations obtained from demonstrations, facilitating safer exploration with reduced contact forces during the learning process. Both techniques aim to enhance stability and data efficiency in contact-rich manipulation by integrating prior structure into reinforcement learning.

2.4. Multi-Modal Reinforcement Learning for Manipulation in Contact-Rich Environments

Several recent studies have directly integrated visual and force inputs into reinforcement learning frameworks for robotic assembly tasks, thereby improving accuracy and resilience.

Zhang et al. [21] used visual and force data in their residual reinforcement learning framework. The baseline controller uses force inputs to ensure alignment and control contact during assembly, whereas the residual reinforcement learning policy learns corrective actions to reduce modeling inaccuracies and environmental variations by using visual information.

Ahn et al. [30] proposed an assembly method that combines visual and force data using two neural network-driven trajectory generators. The image-based trajectory generator generates motion commands from camera images, whereas the force-based trajectory generator uses measured force and robot state data. The force-based network is pre-trained using imitation learning based on hand-guided demonstrations, and then optimized with Soft Actor-Critic reinforcement learning. The outputs from the two generators are combined to determine the trajectory of the robot. Experiments with peg-in-hole tasks show that successful assembly is possible despite significant initial positional and orientational inconsistencies, as well as diverse part geometries.

Jin et al. [31] presented a vision-force-integrated curriculum learning approach for contact-rich robotic manipulation. Their approach incorporates visual and force elements into a curriculum learning framework, gradually increasing task difficulty. The trained policy is directly transferred from simulation to physical hardware, with no adjustments.

Tang et al. [32] proposed a visual-tactile fusion approach with Soft Actor-Critic reinforcement learning for peg-in-hole assembly environments. Their method combines RGB images, depth maps, tactile force data, and robotic body pose information using an autoencoder-based fusion network to produce a multimodal latent representation. The latent representation is used to train the SAC policy. Experimental results show improved assembly success rates, shorter task completion times, and higher success rates and task efficiency when compared to unimodal baselines in various scenarios.

Luo et al. [33] integrated pre-trained visual features obtained from a Variational Autoencoder with human demonstrations in their multimodal DDPGfD algorithm, where demonstrations fill the replay buffer and human corrections resolve failures. Nevertheless, the policy is not explicitly guided by demonstration trajectories, which could lead to unsafe behaviors during RL exploration.

Tang et al. [34] proposed IndustReal, a multimodal assembly framework that integrates vision-based pose estimation with robot proprioception, where object poses are estimated using a Mask R-CNN perception pipeline. Despite promising sim-to-real results, the lack of force/torque feedback means the policy cannot directly sense contact forces, which may limit robustness in tasks necessitating precise force regulation during insertion.

Guo et al. [12] integrated high dimensional raw RGB image observations into a vision-based residual RL policy to capture fine-grained visual details, at the expense of handling dense visual inputs that increase computational complexity and demonstrate sensitivity to variations in lighting, texture, and scene appearance.

2.5. Research Gap and Thesis Positioning

The literature on contact-rich robotic assembly identifies three main research directions. Classical control provides stability through analytical models and predefined control laws, but it lacks adaptability in unpredictable situations and depends heavily on model accuracy. Pure reinforcement learning allows for flexible policy optimization through interaction; however, it frequently encounters challenges associated with dangerous exploration and sample inefficiency in scenarios involving significant contact. Imitation-guided reinforcement learning attempts to close this gap by utilizing structure derived from demonstrations.

Kernelized Guided Reinforcement Learning (KGRL) [9], based on Null-Space Kernelized Movement Primitives (NS-KMP) [10], constrains policy learning through uncertainty-aware trajectory distributions, improving safety and convergence over unconstrained reinforcement learning. Nonetheless, current KGRL formulations rely primarily on proprioceptive and force-based state representations, with no explicit incorporation of learned visual features.

Concurrently, vision-based reinforcement learning methods typically use high-dimensional visual observations to improve perception and alignment. Although these methods provide a wealth of sensory input, they typically lack structured, demonstration-based constraints, necessitating extensive data collection and exploration in contact-rich environments.

A research gap appears at the intersection of these research directions. Demonstration-guided methods allow for structured and uncertainty-aware exploration, but they operate primarily without learned visual representations. Vision-based RL methods incorporate perception, but rarely uses probabilistic guidance derived from demonstrations. The combination of visual perception and uncertainty-aware, demonstration-guided RL remains insufficiently explored.

This thesis fills that gap by combining KGRL with learnt compact visual feature representations. The goal is to maintain the safety and sample efficiency of Reinforcement Learning, while allowing for perception-driven adjustments in contact-rich robotic assembly tasks.

3. Background

This chapter gives an overview of the theoretical and mathematical ideas that are needed to explain the proposed method. It introduces probabilistic modeling of demonstrations, kernelized movement primitives and their extensions, and reinforcement learning preliminaries for continuous control. The presented concepts establish a common notation and provide the tools used in the methodology chapter.

3.1. Learning from Demonstrations: Probabilistic Modeling

3.1.1. Gaussian Mixture Models (GMMs)

A Gaussian Mixture Model (GMM) is a probabilistic model that represents a data distribution as a weighted sum of multiple Gaussian components [35]. For a data point $\boldsymbol{x} \in \mathbb{R}^d$, the probability density function of a GMM is defined as

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{3.1}$$

where K denotes the number of Gaussian components, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the mean vector and covariance matrix of the k-th Gaussian component, and π_k are the mixing coefficients. The mixing coefficients satisfy

$$\pi_k \geq 0, \qquad \sum_{k=1}^{K} \pi_k = 1. \tag{3.2}$$

Each Gaussian component represents a specific region of the data distribution, and the mixing coefficients define the contribution of each component to the overall probability density. By integrating numerous Gaussian components, GMMs may describe complicated and multimodal distributions in a compact and flexible way.

In robotic skill learning, GMMs are commonly used to simulate the joint distribution of task-relevant variables, such as position and orientation extracted from the demonstrations [36]. This probabilistic formulation captures variability across demonstrations and provides the foundation for generating smooth trajectories through Gaussian Mixture Regression, which is discussed in the following section.

3.1.2. Gaussian Mixture Regression (GMR)

Gaussian Mixture Regression (GMR) extends Gaussian Mixture Models by enabling the prediction of output variables given an input by conditioning of a joint probability distribution

[35, 37]. GMR uses a mixture of multivariate Gaussian components to represent the joint distribution of input and output variables rather than modeling them separately.

Let $\boldsymbol{x} \in \mathbb{R}^{d_x}$ denote the input vector and $\boldsymbol{y} \in \mathbb{R}^{d_y}$ the corresponding output vector. Their joint distribution is modeled as

$$p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \;\middle|\; \begin{bmatrix} \boldsymbol{\mu}_x^{(k)} \\ \boldsymbol{\mu}_y^{(k)} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx}^{(k)} & \boldsymbol{\Sigma}_{xy}^{(k)} \\ \boldsymbol{\Sigma}_{yx}^{(k)} & \boldsymbol{\Sigma}_{yy}^{(k)} \end{bmatrix} \right), \tag{3.3}$$

where π_k are the mixing coefficients, $\boldsymbol{\mu}_x^{(k)}$ and $\boldsymbol{\mu}_y^{(k)}$ are the mean vectors for the input and output dimensions, and the covariance matrix is partitioned into input, output, and cross-covariance blocks.

For each Gaussian component, the conditional distribution of the output $\boldsymbol{y}$ given an input $\boldsymbol{x}$ is obtained using the standard conditioning rule for multivariate Gaussian distributions. The conditional mean and covariance of component k are given by

$$\boldsymbol{\mu}_{y|x}^{(k)} = \boldsymbol{\mu}_y^{(k)} + \boldsymbol{\Sigma}_{yx}^{(k)} \left(\boldsymbol{\Sigma}_{xx}^{(k)} \right)^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_x^{(k)} \right), \tag{3.4}$$

$$\boldsymbol{\Sigma}_{y|x}^{(k)} = \boldsymbol{\Sigma}_{yy}^{(k)} - \boldsymbol{\Sigma}_{yx}^{(k)} \left(\boldsymbol{\Sigma}_{xx}^{(k)} \right)^{-1} \boldsymbol{\Sigma}_{xy}^{(k)}. \tag{3.5}$$

The final regression result is obtained by combining the conditional distributions of all components, weighted by their posterior responsibilities. The conditional output distribution is thus expressed as

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{k=1}^{K} h_k(\boldsymbol{x}) \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{\mu}_{y|x}^{(k)}, \boldsymbol{\Sigma}_{y|x}^{(k)} \right), \tag{3.6}$$

where the responsibility of component k is defined as

$$h_k(\boldsymbol{x}) = \frac{\pi_k \, \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_x^{(k)}, \boldsymbol{\Sigma}_{xx}^{(k)})}{\sum_{j=1}^{K} \pi_j \, \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_x^{(j)}, \boldsymbol{\Sigma}_{xx}^{(j)})}. \tag{3.7}$$

The responsibilities act as normalized weighting factors, indicating how much each Gaussian component contributes to the prediction at each given input. As a result, GMR generates smooth and continuous regression outputs while directly describing uncertainty using expected covariance.

For computational efficiency, some applications may benefit from approximating the conditional mixture $p(\boldsymbol{y} \mid \boldsymbol{x})$ with a single Gaussian distribution. Moment matching [35] allows for a common approximation by matching the global mean and covariance of a mixture to those of an equivalent Gaussian distribution.

Given component means $\boldsymbol{\mu}_{y|x}^{(k)}$ and covariances $\boldsymbol{\Sigma}_{y|x}^{(k)}$ with responsibilities $h_k(\boldsymbol{x})$, the moment-matched mean and covariance are computed as

$$\boldsymbol{\mu}_{y|x} = \sum_{k=1}^{K} h_k(\boldsymbol{x}) \boldsymbol{\mu}_{y|x}^{(k)}, \tag{3.8}$$

$$\boldsymbol{\Sigma}_{y|x} = \sum_{k=1}^{K} h_k(\boldsymbol{x}) \left(\boldsymbol{\Sigma}_{y|x}^{(k)} + (\boldsymbol{\mu}_{y|x}^{(k)} - \boldsymbol{\mu}_{y|x})(\boldsymbol{\mu}_{y|x}^{(k)} - \boldsymbol{\mu}_{y|x})^\top \right). \tag{3.9}$$

This approximation smoothes away multimodal features in the original mixture, but gives a compact and computationally efficient representation suitable for real-time regression or control applications.

In robotic skill learning, GMR is commonly used to build motion trajectories from demonstrations by conditioning on a phase variable like time [36]. This formulation replicates both the expected trajectory and its associated unpredictability, making GMR ideal for learning continuous motions from a small number of demonstrations.

3.2. Kernelized Movement Primitives (KMPs)

Kernelized Movement Primitives (KMPs) provide a nonparametric framework for learning and reproducing robot motions from demonstrations [11]. The central idea is to begin with a probabilistic reference trajectory extracted from demonstrations and then construct a trajectory model whose predicted distribution matches that reference trajectory as closely as possible. In contrast to methods that require manually chosen basis functions in the final prediction formula, KMP reformulates the problem in a kernel form. This makes it possible to model complex motion patterns while remaining applicable to high dimensional inputs.

Let the demonstrated dataset be defined as

$$\{\{\boldsymbol{s}_{n,h}, \boldsymbol{\xi}_{n,h}\}_{n=1}^{N}\}_{h=1}^{H}, \tag{3.10}$$

where $\boldsymbol{s}_{n,h} \in \mathbb{R}^{\mathcal{I}}$ denotes the input and $\boldsymbol{\xi}_{n,h} \in \mathbb{R}^{\mathcal{O}}$ denotes the corresponding output. Here, $\mathcal{I}$ and $\mathcal{O}$ are the dimensions of the input and output spaces, respectively, while H and N denote the number of demonstrations and the trajectory length.

A probabilistic reference trajectory is first extracted from the demonstrations. For each input $\boldsymbol{s}_n$, the corresponding reference output is modeled as a Gaussian distribution

$$\mathcal{P}_{\mathbf{r}}(\boldsymbol{\xi}|\boldsymbol{s}_n) = \mathcal{N}(\boldsymbol{\xi}|\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n), \tag{3.11}$$

where $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$ are the mean and covariance of the reference trajectory at $\boldsymbol{s}_n$. This reference trajectory encodes not only the average demonstrated behavior, but also the variability observed across demonstrations.

KMP starts from the following parametric representation of the trajectory:

$$\boldsymbol{\xi}(\boldsymbol{s}) = \boldsymbol{\Theta}(\boldsymbol{s})^\top \boldsymbol{w}, \tag{3.12}$$

where $\boldsymbol{w} \in \mathbb{R}^{B\mathcal{O}}$ is a weight vector, and $\boldsymbol{\Theta}(\boldsymbol{s}) \in \mathbb{R}^{B\mathcal{O}\times\mathcal{O}}$ is defined by

$$\boldsymbol{\Theta}(\boldsymbol{s}) = \begin{bmatrix} \boldsymbol{\varphi}(\boldsymbol{s}) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\varphi}(\boldsymbol{s}) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\varphi}(\boldsymbol{s}) \end{bmatrix}, \tag{3.13}$$

with $\boldsymbol{\varphi}(s) \in \mathbb{R}^B$ denoting a B dimensional basis function vector.

The weight vector is assumed to follow a Gaussian distribution,

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w), \tag{3.14}$$

where $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ are unknown. This assumption induces a Gaussian distribution over the trajectory itself:

$$\boldsymbol{\xi}(s) \sim \mathcal{N}\left(\boldsymbol{\Theta}(s)^\top \boldsymbol{\mu}_w, \boldsymbol{\Theta}(s)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(s)\right). \tag{3.15}$$

Accordingly, for each input s_n, the parametric trajectory distribution is written as

$$\mathcal{P}_\mathbf{p}(\boldsymbol{\xi}|s_n) = \mathcal{N}\left(\boldsymbol{\xi} \mid \boldsymbol{\Theta}(s_n)^\top \boldsymbol{\mu}_w, \boldsymbol{\Theta}(s_n)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(s_n)\right). \tag{3.16}$$

The aim of KMP is to determine $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ such that the parametric trajectory distribution matches the reference trajectory distribution. This is formulated by minimizing the Kullback-Leibler divergence between the two distributions:

$$J_{\text{ini}}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) = \sum_{n=1}^{N} D_{\text{KL}}\left(\mathcal{P}_\mathbf{p}(\boldsymbol{\xi}|s_n) \parallel \mathcal{P}_\mathbf{r}(\boldsymbol{\xi}|s_n)\right). \tag{3.17}$$

Since both distributions are Gaussian, (3.17) can be rewritten as

$$\begin{aligned} J_{\text{ini}}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) = \sum_{n=1}^{N} \frac{1}{2}\Big(& \log|\hat{\boldsymbol{\Sigma}}_n| - \log\left|\boldsymbol{\Theta}(s_n)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(s_n)\right| - \mathcal{O} \\ & + \text{Tr}\left(\hat{\boldsymbol{\Sigma}}_n^{-1} \boldsymbol{\Theta}(s_n)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(s_n)\right) \\ & + \left(\boldsymbol{\Theta}(s_n)^\top \boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right)^\top \hat{\boldsymbol{\Sigma}}_n^{-1} \left(\boldsymbol{\Theta}(s_n)^\top \boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right)\Big). \end{aligned} \tag{3.18}$$

After removing constant terms $\log|\hat{\boldsymbol{\Sigma}}_n|$, $\mathcal{O}$ and the coefficient $\frac{1}{2}$, this optimization can be decomposed into a mean minimization subproblem and a covariance minimization subproblem. The two can therefore be solved separately.

3.2.1. KMP Mean Prediction

The mean related part of the objective is

$$J_{\text{ini}}(\boldsymbol{\mu}_w) = \sum_{n=1}^{N} \left(\boldsymbol{\Theta}(s_n)^\top \boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right)^\top \hat{\boldsymbol{\Sigma}}_n^{-1} \left(\boldsymbol{\Theta}(s_n)^\top \boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right). \tag{3.19}$$

To avoid overfitting, a regularization term is added. The resulting optimization problem becomes

$$J(\boldsymbol{\mu}_w) = \sum_{n=1}^{N} \left(\boldsymbol{\Theta}(s_n)^\top \boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right)^\top \hat{\boldsymbol{\Sigma}}_n^{-1} \left(\boldsymbol{\Theta}(s_n)^\top \boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right) + \lambda \boldsymbol{\mu}_w^\top \boldsymbol{\mu}_w, \tag{3.20}$$

where $\lambda > 0$ is a regularization coefficient.

This cost has the form of a covariance weighted least squares problem. The role of $\hat{\boldsymbol{\Sigma}}_n^{-1}$ is especially important. Points with small covariance are trusted more strongly and therefore penalize deviations more heavily, whereas points with larger covariance permit greater flexibility in the fitted trajectory. In this way, the optimization naturally accounts for the variability present in the demonstrations.

By using the dual formulation, the optimal solution is

$$\boldsymbol{\mu}_w^* = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu}, \tag{3.21}$$

where

$$\begin{aligned}\boldsymbol{\Phi} &= [\boldsymbol{\Theta}(\boldsymbol{s}_1)\ \boldsymbol{\Theta}(\boldsymbol{s}_2)\ \cdots\ \boldsymbol{\Theta}(\boldsymbol{s}_N)],\\ \boldsymbol{\Sigma} &= \text{blockdiag}(\hat{\boldsymbol{\Sigma}}_1,\ \hat{\boldsymbol{\Sigma}}_2,\ \ldots,\ \hat{\boldsymbol{\Sigma}}_N),\\ \boldsymbol{\mu} &= [\hat{\boldsymbol{\mu}}_1^\top\ \hat{\boldsymbol{\mu}}_2^\top\ \cdots\ \hat{\boldsymbol{\mu}}_N^\top]^\top.\end{aligned} \tag{3.22}$$

For a new query input $\boldsymbol{s}^*$, the predicted mean is obtained by substituting $\boldsymbol{\mu}_w^*$ into the parametric trajectory model:

$$\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{\Theta}(\boldsymbol{s}^*)^\top\boldsymbol{\mu}_w^* = \boldsymbol{\Theta}(\boldsymbol{s}^*)^\top\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu}. \tag{3.23}$$

At this stage, the prediction still depends on explicit basis functions through $\boldsymbol{\Theta}(\boldsymbol{s})$. The next step is therefore to eliminate these basis functions directly by using the kernel trick.

The inner product of the basis functions is defined as

$$\boldsymbol{\varphi}(\boldsymbol{s}_i)^\top\boldsymbol{\varphi}(\boldsymbol{s}_j) = k(\boldsymbol{s}_i, \boldsymbol{s}_j), \tag{3.24}$$

where $k(\cdot,\cdot)$ is a kernel function. Based on the structure of $\boldsymbol{\Theta}(\boldsymbol{s})$, this implies

$$\boldsymbol{\Theta}(\boldsymbol{s}_i)^\top\boldsymbol{\Theta}(\boldsymbol{s}_j) = \begin{bmatrix} k(\boldsymbol{s}_i,\boldsymbol{s}_j) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & k(\boldsymbol{s}_i,\boldsymbol{s}_j) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & k(\boldsymbol{s}_i,\boldsymbol{s}_j) \end{bmatrix}, \tag{3.25}$$

which can be rewritten compactly as the matrix valued kernel

$$\boldsymbol{k}(\boldsymbol{s}_i, \boldsymbol{s}_j) = \boldsymbol{\Theta}(\boldsymbol{s}_i)^\top\boldsymbol{\Theta}(\boldsymbol{s}_j) = k(\boldsymbol{s}_i, \boldsymbol{s}_j)\boldsymbol{I}_{\mathcal{O}}, \tag{3.26}$$

where $\boldsymbol{I}_{\mathcal{O}}$ is the $\mathcal{O}$ dimensional identity matrix.

The full kernel matrix is then defined as

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{s}_1,\boldsymbol{s}_1) & \boldsymbol{k}(\boldsymbol{s}_1,\boldsymbol{s}_2) & \cdots & \boldsymbol{k}(\boldsymbol{s}_1,\boldsymbol{s}_N) \\ \boldsymbol{k}(\boldsymbol{s}_2,\boldsymbol{s}_1) & \boldsymbol{k}(\boldsymbol{s}_2,\boldsymbol{s}_2) & \cdots & \boldsymbol{k}(\boldsymbol{s}_2,\boldsymbol{s}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{k}(\boldsymbol{s}_N,\boldsymbol{s}_1) & \boldsymbol{k}(\boldsymbol{s}_N,\boldsymbol{s}_2) & \cdots & \boldsymbol{k}(\boldsymbol{s}_N,\boldsymbol{s}_N) \end{bmatrix}, \tag{3.27}$$

and the query dependent kernel row is written as

$$\boldsymbol{k}^* = [\boldsymbol{k}(\boldsymbol{s}^*,\boldsymbol{s}_1)\ \boldsymbol{k}(\boldsymbol{s}^*,\boldsymbol{s}_2)\ \cdots\ \boldsymbol{k}(\boldsymbol{s}^*,\boldsymbol{s}_N)]. \tag{3.28}$$

With these definitions, the mean prediction can be expressed entirely in kernel form:

$$\mathbb{E}[\boldsymbol{\xi}(s^*)] = \boldsymbol{k}^* \left(\boldsymbol{K} + \lambda\boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\mu}. \tag{3.29}$$

This equation is the final mean prediction formula of KMP. It shows that the prediction depends only on kernel evaluations between inputs and no longer requires explicit manipulation of the basis functions themselves. This is the key step that turns the method into a practical nonparametric regression framework.

3.2.2. KMP Covariance Prediction

The covariance related part of the KL objective is

$$J_{\text{ini}}(\boldsymbol{\Sigma}_w) = \sum_{n=1}^{N} \Big(-\log \Big| \boldsymbol{\Theta}(s_n)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(s_n) \Big| + \text{Tr}\left(\hat{\boldsymbol{\Sigma}}_n^{-1} \boldsymbol{\Theta}(s_n)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(s_n) \right) \Big). \tag{3.30}$$

As in the mean case, a regularization term is introduced. Using $\text{Tr}(\boldsymbol{\Sigma}_w)$ as a relaxed penalty leads to

$$J(\boldsymbol{\Sigma}_w) = \sum_{n=1}^{N} \Big(-\log \Big| \boldsymbol{\Theta}(s_n)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(s_n) \Big| + \text{Tr}\left(\hat{\boldsymbol{\Sigma}}_n^{-1} \boldsymbol{\Theta}(s_n)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(s_n) \right) \Big) + \lambda \text{Tr}(\boldsymbol{\Sigma}_w). \tag{3.31}$$

Taking the derivative of this objective with respect to $\boldsymbol{\Sigma}_w$ and setting it to zero yields

$$\sum_{n=1}^{N} \Big(-\boldsymbol{\Sigma}_w^{-1} + \boldsymbol{\Theta}(s_n) \hat{\boldsymbol{\Sigma}}_n^{-1} \boldsymbol{\Theta}(s_n)^\top \Big) + \lambda \boldsymbol{I} = 0. \tag{3.32}$$

Using the compact notation (3.22), the optimal covariance is

$$\boldsymbol{\Sigma}_w^* = N \left(\boldsymbol{\Phi} \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi}^\top + \lambda \boldsymbol{I} \right)^{-1}. \tag{3.33}$$

Applying the Woodbury identity leads to

$$\mathbb{D}[\boldsymbol{\xi}(s^*)] = \boldsymbol{\Theta}(s^*)^\top \boldsymbol{\Sigma}_w^* \boldsymbol{\Theta}(s^*). \tag{3.34}$$

$$\mathbb{D}[\boldsymbol{\xi}(s^*)] = N \boldsymbol{\Theta}(s^*)^\top \left(\boldsymbol{\Phi} \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi}^\top + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{\Theta}(s^*). \tag{3.35}$$

$$\mathbb{D}[\boldsymbol{\xi}(s^*)] = \frac{N}{\lambda} \boldsymbol{\Theta}(s^*)^\top \left(\boldsymbol{I} - \boldsymbol{\Phi} \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma} \right)^{-1} \boldsymbol{\Phi}^\top \right) \boldsymbol{\Theta}(s^*). \tag{3.36}$$

Using the kernel definitions (3.26) and (3.27), this expression can also be rewritten in a purely kernel based form:

$$\mathbb{D}[\boldsymbol{\xi}(s^*)] = \frac{N}{\lambda} \left(\boldsymbol{k}(s^*, s^*) - \boldsymbol{k}^* \left(\boldsymbol{K} + \lambda \boldsymbol{\Sigma} \right)^{-1} \boldsymbol{k}^{*\top} \right). \tag{3.37}$$

This is the final covariance prediction formula of KMP. Together with the mean prediction, KMP provides both a nominal trajectory and an uncertainty description around that trajectory, which is especially useful in motion adaptation and probabilistic skill representation.

3.3. Null-Space Kernelized Movement Primitives (NS-KMPs)

In conventional KMP, trajectory modulation is achieved by augmenting the reference trajectory distribution with an additional point $(\bar{s}, \bar{\mu}, \bar{\Sigma})$. When $\bar{\Sigma}$ is sufficiently small, the reproduced trajectory is forced to pass close to $\bar{\mu}$ at $\bar{s}$ [11]. Although effective, this approach requires recomputing and inverting the matrix $(K + \lambda\Sigma)$ whenever a new point is added, which is computationally expensive in online settings [10].

Null-Space KMP (NS-KMP) addresses this limitation by reformulating trajectory modulation through a null space structure. This allows incorporating secondary objectives without modifying the kernel matrix, thus avoiding repeated matrix inversions.

3.3.1. Null-space formulation as a least-squares problem

Starting from the original KMP objective, an additional term is introduced to bias the solution towards a desired weight vector $\hat{w}$:

$$\mu_w^* = \arg\min_{\mu_w} \left(\Phi^\top \mu_w - \mu\right)^\top \Sigma^{-1} \left(\Phi^\top \mu_w - \mu\right) + \alpha \mu_w^\top \mu_w + \beta \left(\mu_w - \hat{w}\right)^\top \left(\mu_w - \hat{w}\right). \quad (3.38)$$

This formulation corresponds to a regularized weighted least squares problem with an additional quadratic penalty enforcing proximity to $\hat{w}$. Taking the derivative and setting it to zero yields

$$\mu_w^* = \left(\Phi\Sigma^{-1}\Phi^\top + (\alpha + \beta) I\right)^{-1} \left(\Phi\Sigma^{-1}\mu + \beta\hat{w}\right). \quad (3.39)$$

Defining $\lambda = \alpha + \beta$, the solution can be rewritten as

$$\mu_w^* = \Phi \left(\Phi^\top \Phi + \lambda\Sigma\right)^{-1} \mu + \beta \left(\Phi\Sigma^{-1}\Phi^\top + \lambda I\right)^{-1} \hat{w}. \quad (3.40)$$

Applying the Woodbury identity, the inverse term becomes

$$\left(\Phi\Sigma^{-1}\Phi^\top + \lambda I\right)^{-1} = \frac{1}{\lambda}\left[I - \Phi \left(\Phi^\top \Phi + \lambda\Sigma\right)^{-1} \Phi^\top\right]. \quad (3.41)$$

Substituting back into (3.40) gives

$$\mu_w^* = \underbrace{\Phi \left(\Phi^\top \Phi + \lambda\Sigma\right)^{-1}}_{(\Phi^\top)^\dagger} \mu + \frac{\beta}{\lambda}\left[I - \underbrace{\Phi \left(\Phi^\top \Phi + \lambda\Sigma\right)^{-1}}_{(\Phi^\top)^\dagger} \Phi^\top\right] \hat{w}. \quad (3.42)$$

This expression corresponds to a classical least-squares solution with a null space component, where the second term projects $\hat{w}$ onto the null space of $\Phi^\top$.

For the remainder, it is assumed without loss of generality that $\beta/\lambda = 1$.

3.3.2. Kernelized null-space solution

The expected trajectory at a query input s^* is obtained as

$$\mathbb{E}[\boldsymbol{\xi}(s^*)] = \boldsymbol{\Phi}(s^*)^\top \boldsymbol{\mu}_w^*. \tag{3.43}$$

Substituting (3.42) yields

$$\begin{aligned}\mathbb{E}[\boldsymbol{\xi}(s^*)] = \boldsymbol{\Phi}(s^*)^\top \boldsymbol{\Phi} \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\mu} \\ + \left[\boldsymbol{\Phi}(s^*)^\top - \boldsymbol{\Phi}(s^*)^\top \boldsymbol{\Phi} \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\Phi}^\top\right] \hat{w}.\end{aligned} \tag{3.44}$$

Applying the kernel trick leads to

$$\mathbb{E}[\boldsymbol{\xi}(s^*)] = \boldsymbol{k}^* \left(\boldsymbol{K} + \lambda \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\mu} + \left[\boldsymbol{\Phi}(s^*)^\top - \boldsymbol{k}^* \left(\boldsymbol{K} + \lambda \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\Phi}^\top\right] \hat{w}. \tag{3.45}$$

To express the null-space term in trajectory space, the desired weights are mapped to target outputs:

$$\hat{\boldsymbol{\xi}} = \hat{\boldsymbol{\Phi}}^\top \hat{w}. \tag{3.46}$$

Using the right pseudo-inverse,

$$\hat{w} = \hat{\boldsymbol{\Phi}} \left(\hat{\boldsymbol{\Phi}}^\top \hat{\boldsymbol{\Phi}}\right)^{-1} \hat{\boldsymbol{\xi}}. \tag{3.47}$$

Substituting into (3.45) yields

$$\mathbb{E}[\boldsymbol{\xi}(s^*)] = \boldsymbol{k}^* \boldsymbol{\Psi} \boldsymbol{\mu} + \left(\hat{\boldsymbol{k}}^* - \boldsymbol{k}^* \boldsymbol{\Psi} \hat{\boldsymbol{K}}\right) \left(\underline{\boldsymbol{K}}\right)^{-1} \hat{\boldsymbol{\xi}}, \tag{3.48}$$

with

$$\boldsymbol{\Psi} = \left(\boldsymbol{K} + \lambda \boldsymbol{\Sigma}\right)^{-1}, \quad \hat{\boldsymbol{k}}^* = \boldsymbol{\Theta}(s^*)^\top \hat{\boldsymbol{\Phi}}, \quad \hat{\boldsymbol{K}} = \boldsymbol{\Phi}^\top \hat{\boldsymbol{\Phi}}, \quad \underline{\boldsymbol{K}} = \hat{\boldsymbol{\Phi}}^\top \hat{\boldsymbol{\Phi}}. \tag{3.49}$$

For a single secondary target and a squared-exponential kernel, $\underline{\boldsymbol{K}} = \boldsymbol{I}$, resulting in

$$\mathbb{E}[\boldsymbol{\xi}(s^*)] = \boldsymbol{k}^* \boldsymbol{\Psi} \boldsymbol{\mu} + \left(\hat{\boldsymbol{k}}^* - \boldsymbol{k}^* \boldsymbol{\Psi} \hat{\boldsymbol{K}}\right) \hat{\boldsymbol{\xi}}. \tag{3.50}$$

3.3.3. Interpretation

The final expression corresponds to the original KMP prediction augmented with a null-space term. This additional term enables trajectory modulation without modifying the kernel matrix. Since $\boldsymbol{\Psi}$ depends only on the reference distribution, it can be computed offline.

Unlike classical null space projectors, the resulting projector is not strictly idempotent. Its behavior depends on the covariance $\boldsymbol{\Sigma}$, which determines how strongly the null space component influences the trajectory. As $\lambda \boldsymbol{\Sigma} \to \boldsymbol{0}$, the projector approaches an exact null space projector. For larger covariance values, the modulation becomes softer, allowing variability in directions with higher uncertainty.

This formulation provides an efficient mechanism for trajectory adaptation, where modulation is governed by the data distribution, preserving structure in low-variance regions while allowing flexibility in high-variance regions.

3.4. Reinforcement Learning Preliminaries

This section describes the reinforcement learning (RL) formalism used throughout the thesis and defines the notation needed to represent Kernelized Guided Reinforcement Learning (KGRL). The formulation follows the classical RL treatment in [38].

3.4.1. Markov Decision Process

An RL problem is commonly modeled as a Markov Decision Process (MDP), defined by the tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma), \tag{3.51}$$

where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $p(s_{t+1} \mid s_t, a_t)$ the transition dynamics, $r(s_t, a_t)$ the reward function, and $\gamma \in [0, 1)$ the discount factor [38].

The Markov property is satisfied by the transition dynamics, which indicates that the subsequent state is solely determined by the current state and current action,

$$p(s_{t+1} \mid s_0, a_0, \ldots, s_t, a_t) = p(s_{t+1} \mid s_t, a_t). \tag{3.52}$$

At each discrete time step t, the agent observes a state $s_t \in \mathcal{S}$, samples an action $a_t \in \mathcal{A}$ from a policy $\pi(a \mid s_t)$, receives a scalar reward $r_t = r(s_t, a_t)$, and transitions to a successor state $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.

A stochastic policy $\pi(a \mid s)$ produces a distribution over trajectories $\tau = (s_0, a_0, s_1, a_1, \ldots)$. The objective is to find a policy that maximizes the expected discounted return

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \, r_t \right], \tag{3.53}$$

where the expectation is taken over trajectories generated by sampling $a_t \sim \pi(\cdot \mid s_t)$ and $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.

Two central quantities for evaluating a policy π are the state-value function

$$V^\pi(s_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \,\middle|\, s_t \right], \tag{3.54}$$

and the state-action value function (Q-function)

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \,\middle|\, s_t, a_t \right]. \tag{3.55}$$

These two functions are related by $V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi}[Q^\pi(s_t, a_t)]$, and the Q-function satisfies the Bellman equation

$$Q^\pi(s_t, a_t) = \mathbb{E}\Big[r(s_t, a_t) + \gamma \, \mathbb{E}_{a_{t+1} \sim \pi(\cdot \mid s_{t+1})}[Q^\pi(s_{t+1}, a_{t+1})] \Big], \tag{3.56}$$

where $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.

The Bellman equation is the foundation of the majority of practical RL algorithms and formally expresses the recursive relationship between the values of a state-action pair and the values of its successors.

The optimal policy π^* maximizes the expected return for all states $s \in \mathcal{S}$, defined as $\pi^* = \arg\max_\pi V^\pi(s)$. The corresponding optimal value functions are $V^*(s) = \max_\pi V^\pi(s)$ and $Q^*(s,a) = \max_\pi Q^\pi(s,a)$, and the optimal policy can be recovered greedily via

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \; Q^*(s,a). \tag{3.57}$$

This arg max is intractable to compute directly in continuous control settings due to the fact that $\mathcal{A}$ is a continuous set. Actor-critic methods resolve this problem by preserving an explicit parametric policy (actor) that is updated to optimize a parametric value function (critic), both of which are approximated by neural networks [38].

3.4.2. Soft Actor-Critic

All reinforcement learning policies in this thesis are optimized using Soft Actor-Critic (SAC) [7], an off-policy actor-critic algorithm that combines off-policy learning from a replay buffer with a maximum entropy objective. Rather than maximizing the expected discounted return alone, SAC augments the objective with the entropy of the policy at each visited state:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \Big(r(s_t, a_t) + \alpha \, \mathcal{H}\big(\pi(\cdot \mid s_t)\big) \Big) \right], \tag{3.58}$$

where $\alpha > 0$ is the temperature parameter that regulates the relative importance of the entropy term, and $\mathcal{H}(\pi(\cdot \mid s_t)) = -\mathbb{E}_{a_t \sim \pi}[\log \pi(a_t \mid s_t)]$ is the Shannon entropy of the policy at state s_t. The entropy term encourages the agent to explore the state-action space thoroghly and prevents premature convergence to a suboptimal deterministic policy. In the limit $\alpha \to 0$, the standard maximum reward objective is recovered.

The soft state-value function under this objective is defined as

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi}\big[Q^\pi(s_t, a_t) - \alpha \log \pi(a_t \mid s_t) \big], \tag{3.59}$$

and the corresponding soft Q-function satisfies the soft Bellman equation

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \, \mathbb{E}_{s_{t+1} \sim p}\big[V^\pi(s_{t+1}) \big], \tag{3.60}$$

where $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.

Compared to the standard Bellman equation in (3.56), the soft state-value function introduces an entropy term, which encourages the policy to maintain high uncertainty.

In practice, SAC uses function approximators for both the Q-function and the policy. A parametric state-value network $V_\psi(s_t)$, a soft Q-network $Q_\theta(s_t, a_t)$, and a stochastic actor $\pi_\phi(a_t \mid s_t)$ are trained jointly from transitions sampled from a replay buffer $\mathcal{D}$. The value network is trained to minimize

$$J_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} \Big(V_\psi(s_t) - \mathbb{E}_{a_t \sim \pi_\phi}\big[Q_\theta(s_t, a_t) - \alpha \log \pi_\phi(a_t \mid s_t) \big] \Big)^2 \right], \tag{3.61}$$

and the Q-network is trained by minimizing the soft Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\Big(Q_\theta(s_t,a_t) - \hat{Q}(s_t,a_t)\Big)^2\right], \tag{3.62}$$

where $\hat{Q}(s_t,a_t) = r(s_t,a_t) + \gamma\,\mathbb{E}_{s_{t+1}\sim p}\Big[V_{\bar{\psi}}(s_{t+1})\Big]$ is the soft Bellman target and $V_{\bar{\psi}}$ is a target value network whose parameters $\bar{\psi}$ are kept as an exponential moving average of ψ to stabilize training.

The actor is updated by minimizing the expected KL divergence between the policy and the Q-function [7].

Using the reparameterization trick, actions are sampled as $a_t = f_\phi(\epsilon_t; s_t)$ with $\epsilon_t \sim \mathcal{N}(0, I)$, which allows gradients to flow through the sampled action.

The actor objective is

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D},\,\epsilon_t\sim\mathcal{N}}\left[\alpha\log\pi_\phi(a_t \mid s_t) - Q_\theta(s_t,a_t)\right]. \tag{3.63}$$

To mitigate overestimation bias in the Q-function, SAC maintains two independent Q-networks and uses the minimum of their outputs in both (3.61) and (3.63) [7].

The target value network parameters are updated via Polyak averaging,

$$\bar{\psi} \leftarrow \tau\,\psi + (1-\tau)\,\bar{\psi}, \tag{3.64}$$

with smoothing coefficient $\tau \ll 1$.

The full training procedure and convergence analysis are described in [7].

4. Methodology

This chapter describes the proposed learning framework and its implementation for contact rich robotic assembly. The presentation builds upon the probabilistic skill representations and reinforcement learning preliminaries introduced in Chapter 3. The proposed method in this thesis is Vision-Augmented Kernelized Guided Reinforcement Learning (Vision-KGRL), which extends KGRL by incorporating additional visual observation modalities.

To establish the full formulation, the chapter first introduces the linearly constrained null space KMP (LC-NS-KMP) formulation from which KGRL is derived [9]. It then formulates KGRL as a trajectory generation mechanism in which a reinforcement learning policy modulates an imitation learned trajectory through null space actions. Finally, the Vision-KGRL formulation used in this thesis is presented by extending KGRL with visual latent representations.

4.1. LC-NS-KMP Formulation

As introduced in Chapter 3, KMP provides a covariance weighted reproduction of a probabilistic reference trajectory learned from demonstrations, while NS-KMP extends this formulation by introducing a null space modulation term. KGRL in [9] is derived from a more general formulation, namely Linearly Constrained Null Space Kernelized Movement Primitives (LC-NS-KMP). This formulation combines three elements in a single optimization problem: reproduction of the demonstrated trajectory, modulation through a desired null space target, and satisfaction of linear inequality constraints.

In this chapter, the KMP input is denoted by $\boldsymbol{s} \in \mathbb{R}^I$ and the trajectory output by $\boldsymbol{\eta} \in \mathbb{R}^O$. In this thesis, $\boldsymbol{s}$ represents normalized time in the interval $[0, 1]$, where $\boldsymbol{s} = 0$ corresponds to the start and $\boldsymbol{s} = 1$ to the end of the trajectory. The variable $\boldsymbol{\eta}$ denotes the task space quantity to be predicted. The demonstrations are written as

$$\mathcal{D} = \left\{ \{\boldsymbol{s}_{n,m}, \boldsymbol{\eta}_{n,m}\}_{n=1}^{N} \right\}_{m=1}^{M}, \tag{4.1}$$

where N is the trajectory length and M is the number of demonstrations.

Following [9], a joint probability distribution over input and output is modeled using a Gaussian mixture model,

$$\mathcal{P}(\boldsymbol{s}, \boldsymbol{\eta}) \sim \sum_{c=1}^{C} p_c \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{4.2}$$

and Gaussian mixture regression yields the reference trajectory distribution

$$\mathcal{T}_r = \{\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^{N}, \tag{4.3}$$

where $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$ denote the reference mean and covariance at sample index n.

A parametric trajectory representation consistent with KMP is given by

$$\boldsymbol{\eta}(\boldsymbol{s}) = \boldsymbol{\Theta}(\boldsymbol{s})^\top \boldsymbol{w}, \tag{4.4}$$

where $\boldsymbol{w} \in \mathbb{R}^{BO}$ is a weight vector and $\boldsymbol{\Theta}(\boldsymbol{s}) \in \mathbb{R}^{BO \times O}$ is defined as

$$\boldsymbol{\Theta}(\boldsymbol{s}) = \begin{bmatrix} \boldsymbol{\varphi}(\boldsymbol{s}) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\varphi}(\boldsymbol{s}) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\varphi}(\boldsymbol{s}) \end{bmatrix}, \tag{4.5}$$

with $\boldsymbol{\varphi}(\boldsymbol{s}) \in \mathbb{R}^B$ denoting a B-dimensional basis function vector.

The compact notation introduced for KMP in Chapter 3 is used throughout this chapter:

$$\boldsymbol{\Phi} = \left[\boldsymbol{\Theta}(\boldsymbol{s}_1), \boldsymbol{\Theta}(\boldsymbol{s}_2), \ldots, \boldsymbol{\Theta}(\boldsymbol{s}_N)\right], \qquad \boldsymbol{\mu} = \left[\hat{\boldsymbol{\mu}}_1^\top, \ldots, \hat{\boldsymbol{\mu}}_N^\top\right]^\top, \tag{4.6}$$

$$\boldsymbol{\Sigma} = \text{blockdiag}\left(\hat{\boldsymbol{\Sigma}}_1, \hat{\boldsymbol{\Sigma}}_2, \ldots, \hat{\boldsymbol{\Sigma}}_N\right). \tag{4.7}$$

Starting from the constrained KMP mean optimization problem, an additional quadratic term is introduced to keep the solution close to a desired weight vector $\hat{\boldsymbol{\mu}}_w$. The resulting LC-NS-KMP optimization problem is

$$\arg\min_{\boldsymbol{\mu}_w} \sum_{n=1}^{N} \frac{1}{2} \left(\boldsymbol{\Theta}^\top(\boldsymbol{s}_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right)^\top \hat{\boldsymbol{\Sigma}}_n^{-1} \left(\boldsymbol{\Theta}^\top(\boldsymbol{s}_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right) + \frac{1}{2}\lambda \boldsymbol{\mu}_w^\top \boldsymbol{\mu}_w + \frac{1}{2}\beta \left(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w\right)^\top \left(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w\right) \tag{4.8}$$

subject to

$$\boldsymbol{g}_{n,f}^\top \boldsymbol{\eta}(\boldsymbol{s}_n) \geq c_{n,f}, \qquad \forall f \in \{1, 2, \ldots, F\}, \qquad \forall n \in \{1, 2, \ldots, N\}. \tag{4.9}$$

Here, $\lambda > 0$ is the regularization parameter inherited from KMP, $\beta > 0$ controls the strength of the null space modulation, F is the number of linear constraints, and $\boldsymbol{g}_{n,f}$ together with $c_{n,f}$ defines the corresponding constraint hyperplane at sample n. The first two terms reproduce the demonstrated trajectory in the same covariance aware manner as KMP, while the third term softly biases the solution toward a desired one $\hat{\boldsymbol{\mu}}_w$.

To solve (4.8) under the inequality constraints in (4.9), nonnegative Lagrange multipliers $\alpha_{n,f} \geq 0$ are introduced. The Lagrangian becomes

$$\begin{aligned} L(\boldsymbol{\mu}_w, \boldsymbol{\alpha}) = &\sum_{n=1}^{N} \frac{1}{2} \left(\boldsymbol{\Theta}^\top(\boldsymbol{s}_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right)^\top \hat{\boldsymbol{\Sigma}}_n^{-1} \left(\boldsymbol{\Theta}^\top(\boldsymbol{s}_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n\right) \\ &+ \frac{1}{2}\lambda \boldsymbol{\mu}_w^\top \boldsymbol{\mu}_w + \frac{1}{2}\beta \left(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w\right)^\top \left(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w\right) \\ &- \sum_{n=1}^{N} \sum_{f=1}^{F} \alpha_{n,f} \left(\boldsymbol{g}_{n,f}^\top \boldsymbol{\Theta}(\boldsymbol{s}_n)^\top \boldsymbol{\mu}_w - c_{n,f}\right). \end{aligned} \tag{4.10}$$

Using matrix notation, the same expression can be written as

$$L(\boldsymbol{\mu}_w, \boldsymbol{\alpha}) = \frac{1}{2}\left(\boldsymbol{\Phi}^\top \boldsymbol{\mu}_w - \boldsymbol{\mu}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\Phi}^\top \boldsymbol{\mu}_w - \boldsymbol{\mu}\right) + \frac{1}{2}\lambda \boldsymbol{\mu}_w^\top \boldsymbol{\mu}_w + \frac{1}{2}\beta\left(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w\right)^\top \left(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w\right) - \boldsymbol{\alpha}^\top \bar{\boldsymbol{G}}^\top \boldsymbol{\Phi}^\top \boldsymbol{\mu}_w - \boldsymbol{\alpha}^\top \bar{\boldsymbol{C}}, \tag{4.11}$$

where

$$\boldsymbol{G}_n = \begin{bmatrix} \boldsymbol{g}_{n,1} & \boldsymbol{g}_{n,2} & \dots & \boldsymbol{g}_{n,F} \end{bmatrix}, \qquad \forall n \in \{1, 2, \dots, N\}, \tag{4.12}$$

$$\bar{\boldsymbol{G}} = \text{blockdiag}\left(\boldsymbol{G}_1, \boldsymbol{G}_2, \dots, \boldsymbol{G}_N\right), \tag{4.13}$$

$$\boldsymbol{\alpha} = \left[\alpha_{1,1}, \alpha_{1,2}, \dots, \alpha_{1,F}, \dots, \alpha_{N,1}, \dots, \alpha_{N,F}\right]^\top, \tag{4.14}$$

and

$$\bar{\boldsymbol{C}} = \left[\boldsymbol{C}_1^\top, \boldsymbol{C}_2^\top, \dots, \boldsymbol{C}_N^\top\right]^\top, \qquad \boldsymbol{C}_n = \left[c_{n,1}, c_{n,2}, \dots, c_{n,F}\right]^\top. \tag{4.15}$$

Setting the derivative of (4.11) with respect to $\boldsymbol{\mu}_w$ to zero yields

$$\left(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top + \gamma \boldsymbol{I}\right)\boldsymbol{\mu}_w^* = \boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \beta\hat{\boldsymbol{\mu}}_w + \boldsymbol{\Phi}\bar{\boldsymbol{G}}\boldsymbol{\alpha}, \tag{4.16}$$

Which results in,

$$\boldsymbol{\mu}_w^* = \left(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top + \gamma \boldsymbol{I}\right)^{-1}\left(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \beta\hat{\boldsymbol{\mu}}_w + \boldsymbol{\Phi}\bar{\boldsymbol{G}}\boldsymbol{\alpha}\right). \tag{4.17}$$

Using the Woodbury identity, (4.17) can be rewritten as

$$\boldsymbol{\mu}_w^* = \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + \frac{\beta}{\gamma}\left(\boldsymbol{I} - \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Phi}^\top\right)\hat{\boldsymbol{\mu}}_w, \tag{4.18}$$

where

$$\boldsymbol{A} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \gamma\boldsymbol{\Sigma})^{-1}, \qquad \gamma = \lambda + \beta. \tag{4.19}$$

Substituting $\boldsymbol{\mu}_w^*$ into (4.11) and (4.4) yields

$$\tilde{L}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \bar{\boldsymbol{G}}^\top \boldsymbol{\Sigma}\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + \left(2\boldsymbol{\mu}^\top \boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}} - \beta\hat{\boldsymbol{\mu}}_w^\top \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}} + \bar{\boldsymbol{C}}^\top\right)\boldsymbol{\alpha} + \text{const}, \tag{4.20}$$

and

$$\mathbb{E}\left(\boldsymbol{\eta}(\boldsymbol{s}^*)\right) = \boldsymbol{\Theta}(\boldsymbol{s}^*)\left(\boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + \frac{\beta}{\gamma}\left(\boldsymbol{I} - \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Phi}^\top\right)\hat{\boldsymbol{\mu}}_w\right), \tag{4.21}$$

respectively, where

$$\mathcal{A} = -\frac{1}{2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \frac{\gamma}{2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}. \tag{4.22}$$

To express the null space term directly in trajectory space, a desired output

$$\boldsymbol{\xi} = \hat{\boldsymbol{\Phi}}^\top \hat{\boldsymbol{\mu}}_w \tag{4.23}$$

is introduced. The corresponding desired weight vector can be estimated using the right pseudo inverse of $\hat{\boldsymbol{\Phi}}^\top$ as

$$\hat{\boldsymbol{\mu}}_w = \hat{\boldsymbol{\Phi}}\left(\hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\Phi}}\right)^{-1}\boldsymbol{\xi}. \tag{4.24}$$

Substituting (4.24) into (4.20) and (4.21) yields

$$\tilde{L}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \bar{\boldsymbol{G}}^\top \boldsymbol{\Sigma} \boldsymbol{A} \mathcal{A} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} \boldsymbol{\alpha} + \left(2\boldsymbol{\mu}^\top \boldsymbol{A} \mathcal{A} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} - \beta \boldsymbol{\xi}^\top \left(\hat{\boldsymbol{\Phi}}^\top \hat{\boldsymbol{\Phi}} \right)^{-1} \hat{\boldsymbol{\Phi}}^\top \boldsymbol{\Phi} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} + \bar{\boldsymbol{C}}^\top \right) \boldsymbol{\alpha} + \text{const}, \tag{4.25}$$

and

$$\mathbb{E}\left(\boldsymbol{\eta}(s^*)\right) = \boldsymbol{\Theta}(s^*) \left(\boldsymbol{\Phi} \boldsymbol{A} \boldsymbol{\mu} + \boldsymbol{\Phi} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} \boldsymbol{\alpha} + \frac{\beta}{\gamma} \left(\boldsymbol{I} - \boldsymbol{\Phi} \boldsymbol{A} \boldsymbol{\Phi}^\top \right) \hat{\boldsymbol{\Phi}} \left(\hat{\boldsymbol{\Phi}}^\top \hat{\boldsymbol{\Phi}} \right)^{-1} \boldsymbol{\xi} \right). \tag{4.26}$$

Applying the kernel treatment introduced for KMP in Chapter 3, with

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}(s_1, s_1) & \dots & \boldsymbol{k}(s_1, s_N) \\ \vdots & \ddots & \vdots \\ \boldsymbol{k}(s_N, s_1) & \dots & \boldsymbol{k}(s_N, s_N) \end{bmatrix}, \qquad \boldsymbol{k}^* = \left[\boldsymbol{k}(s^*, s_1), \dots, \boldsymbol{k}(s^*, s_N) \right], \tag{4.27}$$

$$\boldsymbol{k}(s_i, s_j) = k(s_i, s_j) \boldsymbol{I}, \qquad \underline{\boldsymbol{K}} = \hat{\boldsymbol{\Phi}}^\top \hat{\boldsymbol{\Phi}}, \qquad \hat{\boldsymbol{K}} = \boldsymbol{\Phi}^\top \hat{\boldsymbol{\Phi}}, \qquad \hat{k}^* = \boldsymbol{\Theta}(s^*)^\top \hat{\boldsymbol{\Phi}}, \tag{4.28}$$

(4.25) and (4.26) become

$$\tilde{L}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \bar{\boldsymbol{G}}^\top \boldsymbol{\Sigma} \boldsymbol{A} \mathcal{A} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} \boldsymbol{\alpha} + \left(2\boldsymbol{\mu}^\top \boldsymbol{A} \mathcal{A} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} - \beta \boldsymbol{\xi}^\top \underline{\boldsymbol{K}}^{-1} \hat{\boldsymbol{K}} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} + \bar{\boldsymbol{C}}^\top \right) \boldsymbol{\alpha} + \text{const}, \tag{4.29}$$

and

$$\mathbb{E}\left(\boldsymbol{\eta}(s^*)\right) = \boldsymbol{k}^* \boldsymbol{A} \boldsymbol{\mu} + \boldsymbol{k}^* \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} \boldsymbol{\alpha} + \frac{\beta}{\gamma} \left(\hat{\boldsymbol{k}}^* - \boldsymbol{k}^* \boldsymbol{A} \hat{\boldsymbol{K}} \right) \underline{\boldsymbol{K}}^{-1} \boldsymbol{\xi}, \tag{4.30}$$

where

$$\boldsymbol{A} = (\boldsymbol{K} + \gamma \boldsymbol{\Sigma})^{-1}, \qquad \mathcal{A} = -\frac{1}{2} \boldsymbol{K} \boldsymbol{\Sigma}^{-1} \boldsymbol{K} - \frac{\gamma}{2} \boldsymbol{K}. \tag{4.31}$$

This kernelized expression shows the three roles of the model clearly. The first term reproduces the demonstrated trajectory, the second term enforces the linear constraints, and the third term introduces the null space modulation in a covariance aware manner.

For convenience, define

$$\mathcal{B}_1 = \bar{\boldsymbol{G}}^\top \boldsymbol{\Sigma} \boldsymbol{A} \mathcal{A} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}}, \qquad \mathcal{B}_2 = 2\boldsymbol{\mu}^\top \boldsymbol{A} \mathcal{A} \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} + \beta \boldsymbol{\xi}^\top \underline{\boldsymbol{K}}^{-1} \hat{\boldsymbol{K}} (-\boldsymbol{A}) \boldsymbol{\Sigma} \bar{\boldsymbol{G}} + \bar{\boldsymbol{C}}^\top. \tag{4.32}$$

Then (4.29) becomes a quadratic function in $\boldsymbol{\alpha}$, and the optimal Lagrange multipliers are obtained by solving

$$\arg\max_{\boldsymbol{\alpha}} \quad \boldsymbol{\alpha}^\top \mathcal{B}_1 \boldsymbol{\alpha} + \mathcal{B}_2 \boldsymbol{\alpha}, \qquad \text{s.t.} \quad \boldsymbol{\alpha} \geq 0. \tag{4.33}$$

After solving for $\boldsymbol{\alpha}$, (4.30) yields a trajectory prediction that incorporates the demonstrated behavior, the desired modulation, and the linear inequality constraints in a unified form.

In this thesis, no linear inequality constraints are used, as the primary objective is to study the effect of demonstration-based guidance on policy learning rather than bounding exploration through hard constraints. Consequently, the constrained correction term $\boldsymbol{k}^* \boldsymbol{A} \boldsymbol{\Sigma} \bar{\boldsymbol{G}} \boldsymbol{\alpha}$ is inactive in the implemented system, which corresponds to $\boldsymbol{\alpha} = \boldsymbol{0}$. Nevertheless, the LC-NS-KMP derivation is included here because it is the general formulation from which KGRL is derived in [9].

4.2. Kernelized Guided Reinforcement Learning

KGRL uses reinforcement learning to generate the null space target that modulates the imitation learned trajectory. Instead of outputting the final robot command directly, the policy produces a null space action,

$$\boldsymbol{\xi}_t \sim \pi(\boldsymbol{\xi} \mid \boldsymbol{q}_t), \tag{4.34}$$

where $\boldsymbol{q}_t$ is the policy input at time step t. This action is then passed through the LC-NS-KMP predictor. As a result, reinforcement learning does not act directly in the trajectory output space. Rather, it influences the output through a modulation term that is filtered by the covariance structure of the demonstrations.

The reinforcement learning observation is denoted by $\boldsymbol{o}_t \in \mathcal{O}$ and the policy input by $\boldsymbol{q}_t \in \mathcal{Q}$. From the reinforcement learning formalism introduced in Chapter 3, the state corresponds to the selected observation vector $\boldsymbol{q}_t$, and the action corresponds to the null space action $\boldsymbol{\xi}_t$. The policy in (4.34) is optimized with SAC, while the sampled null space action is transformed into the actual task space command through the trajectory model.

Substituting the policy output into the general LC-NS-KMP expression in (4.30) gives the KGRL prediction

$$\mathbb{E}\big(\boldsymbol{\eta}(\boldsymbol{s}^*)\big) = \boldsymbol{k}^* \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{k}^* \boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + \frac{\beta}{\gamma}\left(\hat{\boldsymbol{k}}^* - \boldsymbol{k}^* \boldsymbol{A}\hat{\boldsymbol{K}}\right)\boldsymbol{\xi}. \tag{4.35}$$

Since no linear inequality constraints are used in this thesis, the implemented KGRL formulation reduces to

$$\mathbb{E}\big(\boldsymbol{\eta}(\boldsymbol{s}^*)\big) = \boldsymbol{k}^* \boldsymbol{A}\boldsymbol{\mu} + \frac{\beta}{\gamma}\left(\hat{\boldsymbol{k}}^* - \boldsymbol{k}^* \boldsymbol{A}\hat{\boldsymbol{K}}\right)\boldsymbol{\xi}. \tag{4.36}$$

Equation (4.36) is the KGRL prediction equation used in this thesis. The first term defines the nominal trajectory prior extracted from demonstrations, while the second term allows reinforcement learning to adapt that trajectory through null space actions. Because the modulation is shaped by the demonstration covariance, exploration becomes structured rather than arbitrary. Regions with low demonstrated variance resist large deviations, whereas regions with higher uncertainty allow more freedom for adaptation.

This property is particularly relevant for contact rich assembly. Direct reinforcement learning in Cartesian task space may lead to unsafe or inefficient exploration, especially near contact transitions. In contrast, KGRL restricts the policy to operate through null space actions that are mapped into smooth trajectory modifications that are compatible with the learned skill prior. The reinforcement learning policy therefore refines an existing motion representation instead of replacing it.

The action sampled from the policy is transformed into the actual task space command through (4.36), and the resulting command is then applied to the robot. In this way, the reinforcement learning problem is defined over null space actions, while execution remains grounded in the imitation learned trajectory model.

4.3. Vision Augmented KGRL

The proposed method extends KGRL by incorporating visual information into the policy input while preserving the trajectory generation mechanism defined in (4.36). The LC-NS-KMP structure remains unchanged. Only the policy state is augmented with a learned compact visual representation. This results in Vision-KGRL, where perception is integrated at the policy level while trajectory generation remains governed by the probabilistic skill model.

4.3.1. Denoising Variational Autoencoder for Visual Latent Representation

To incorporate visual information into the policy in a compact and noise-resistant manner, a denoising variational autoencoder (VAE) [39] is utilized to map high-dimensional RGB observations to a low-dimensional latent representation.

Let

$$\boldsymbol{o}_t^{\text{img}} \in \mathbb{R}^{H\times W\times C} \tag{4.37}$$

denote the clean RGB image observation at time step t, where H, W, and C denote the image height, width, and number of channels. A latent variable

$$\boldsymbol{z}_t \in \mathbb{R}^{d_z} \tag{4.38}$$

is introduced to represent the image in a compact form, where d_z denotes the dimensionality of the latent space.

In the denoising setting, the encoder does not operate on the clean image directly. Instead, a stochastically augmented version of the image is generated as

$$\tilde{\boldsymbol{o}}_t^{\text{img}} \sim p_{\text{aug}}\left(\tilde{\boldsymbol{o}}_t^{\text{img}} \mid \boldsymbol{o}_t^{\text{img}}\right), \tag{4.39}$$

where $p_{\text{aug}}(\cdot)$ denotes an image corruption or augmentation process. The encoder takes $\tilde{\boldsymbol{o}}_t^{\text{img}}$ as input, while the decoder is trained to reconstruct the clean image $\boldsymbol{o}_t^{\text{img}}$. This formulation encourages the latent variable to capture stable and task-relevant structure rather than simulation-specific perturbations.

Following the standard variational autoencoder formulation [39], a prior distribution over the latent variable is defined as

$$p(\boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{z}_t \mid \boldsymbol{0}, \boldsymbol{I}). \tag{4.40}$$

The encoder defines an approximate posterior distribution over the latent variable conditioned on the augmented image,

$$q_\phi\left(\boldsymbol{z}_t \mid \tilde{\boldsymbol{o}}_t^{\text{img}}\right) = \mathcal{N}\left(\boldsymbol{z}_t \;\middle|\; \boldsymbol{\mu}_\phi\left(\tilde{\boldsymbol{o}}_t^{\text{img}}\right), \operatorname{diag}\left(\boldsymbol{\sigma}_\phi^2\left(\tilde{\boldsymbol{o}}_t^{\text{img}}\right)\right)\right), \tag{4.41}$$

where $\boldsymbol{\mu}_\phi(\cdot), \boldsymbol{\sigma}_\phi^2(\cdot) \in \mathbb{R}^{d_z}$ and ϕ denotes the parameters of the encoder neural network. Thus, the encoder maps the augmented image to the mean and diagonal variance of a Gaussian distribution in latent space.

The decoder defines the likelihood of the clean image given the latent variable as

$$p_\psi\left(\boldsymbol{o}_t^{\text{img}} \mid \boldsymbol{z}_t\right) = \mathcal{N}\left(\boldsymbol{o}_t^{\text{img}} \;\middle|\; f_\psi(\boldsymbol{z}_t),\, \sigma_o^2 \boldsymbol{I}\right), \tag{4.42}$$

where $f_\psi(\cdot)$ denotes the decoder neural network with parameters ψ, and σ_o^2 is a fixed observation variance.

The VAE model is trained by maximizing the evidence lower bound (ELBO) [39], which for each sample t is given by

$$\mathcal{L}_{\text{ELBO}}^{(t)} = \mathbb{E}_{q_\phi(\boldsymbol{z}_t \mid \tilde{\boldsymbol{o}}_t^{\text{img}})}\left[\log p_\psi\left(\boldsymbol{o}_t^{\text{img}} \mid \boldsymbol{z}_t\right)\right] - D_{\text{KL}}\left(q_\phi\left(\boldsymbol{z}_t \mid \tilde{\boldsymbol{o}}_t^{\text{img}}\right) \parallel p(\boldsymbol{z}_t)\right). \tag{4.43}$$

A detailed derivation of the ELBO from the marginal likelihood can be found in [39].

To enable gradient-based optimization, the stochastic sampling from the encoder is expressed using the reparameterization trick [39]:

$$\boldsymbol{z}_t = \boldsymbol{\mu}_\phi\left(\tilde{\boldsymbol{o}}_t^{\text{img}}\right) + \boldsymbol{\sigma}_\phi\left(\tilde{\boldsymbol{o}}_t^{\text{img}}\right) \odot \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{4.44}$$

where $\odot$ denotes elementwise multiplication. This formulation allows gradients to propagate through the sampling operation.

In practice, training is performed by minimizing the negative ELBO over a dataset of N samples:

$$\mathcal{L}_{\text{VAE}} = -\frac{1}{N}\sum_{t=1}^{N} \mathcal{L}_{\text{ELBO}}^{(t)}. \tag{4.45}$$

With the Gaussian decoder and fixed variance, the reconstruction term corresponds to a squared error loss up to constant factors. Therefore, the objective used in this thesis is written as

$$\mathcal{L}_{\text{VAE}} = \frac{1}{N}\sum_{t=1}^{N}\left[\left\|\boldsymbol{o}_t^{\text{img}} - f_\psi(\boldsymbol{z}_t)\right\|_2^2 + D_{\text{KL}}\left(q_\phi\left(\boldsymbol{z}_t \mid \tilde{\boldsymbol{o}}_t^{\text{img}}\right) \parallel p(\boldsymbol{z}_t)\right)\right]. \tag{4.46}$$

Since both the approximate posterior and the prior are Gaussian, the KL divergence admits a closed-form expression:

$$D_{\text{KL}}\left(q_\phi(\boldsymbol{z}_t \mid \tilde{\boldsymbol{o}}_t^{\text{img}}) \parallel p(\boldsymbol{z}_t)\right) = \frac{1}{2}\sum_{j=1}^{d_z}\left(\mu_{t,j}^2 + \sigma_{t,j}^2 - \log\sigma_{t,j}^2 - 1\right), \tag{4.47}$$

where $\mu_{t,j}$ and $\sigma_{t,j}^2$ denote the jth components of the encoder outputs.

Substituting (4.47) into (4.46), the final training objective used in this thesis is given by

$$\mathcal{L}_{\text{VAE}} = \frac{1}{N}\sum_{t=1}^{N}\left[\left\|\boldsymbol{o}_t^{\text{img}} - f_\psi(\boldsymbol{z}_t)\right\|_2^2 + \frac{1}{2}\sum_{j=1}^{d_z}\left(\mu_{t,j}^2 + \sigma_{t,j}^2 - \log\sigma_{t,j}^2 - 1\right)\right]. \tag{4.48}$$

Algorithm 1 summarizes the VAE training procedure. The encoder and decoder are trained by minimizing $\mathcal{L}_{\text{VAE}}$ across a dataset of images obtained from the simulation environments. Durimg inference, the encoder weights are fixed, and the mean of the encoder distribution serves as the visual latent representation throughout RL policy training and inference.

Algorithm 1 VAE Training for Visual Latent Representation

Initialize encoder parameters ϕ and decoder parameters ψ
repeat
  Sample a mini-batch $\{\boldsymbol{o}_t^{\text{img}}\}$
  Generate augmented observations $\tilde{\boldsymbol{o}}_t^{\text{img}} \sim p_{\text{aug}}(\tilde{\boldsymbol{o}}_t^{\text{img}} \mid \boldsymbol{o}_t^{\text{img}})$
  Encoder: compute $\boldsymbol{\mu}_\phi(\tilde{\boldsymbol{o}}_t^{\text{img}})$ and $\boldsymbol{\sigma}_\phi(\tilde{\boldsymbol{o}}_t^{\text{img}})$
  Sample latent via reparameterization: $\boldsymbol{z}_t = \boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  Decoder: reconstruct clean image $f_\psi(\boldsymbol{z}_t)$
  Compute $\mathcal{L}_{\text{VAE}}$ using (4.46)
  Update ϕ, ψ by gradient descent on $\mathcal{L}_{\text{VAE}}$
until convergence

4.3.2. Vision Augmented KGRL Policy

During inference, the clean RGB observation $\boldsymbol{o}_t^{\text{img}} \in \mathbb{R}^{H\times W\times C}$ is processed by the trained encoder to produce a latent representation

$$\boldsymbol{z}_t = \boldsymbol{\mu}_\phi\left(\boldsymbol{o}_t^{\text{img}}\right), \tag{4.49}$$

where $\boldsymbol{\mu}_\phi(\cdot)$ denotes the mean of the encoder distribution.

The latent vector $\boldsymbol{z}_t$ forms the visual component of the policy observation, as defined in (5.4). It encodes task-relevant geometric and spatial information from the scene in a compact form, which is not directly available from proprioceptive or force measurements.

If vision is enabled, the policy input is augmented with the visual latent and is written as

$$\boldsymbol{q}_t^{\text{aug}} = \left[\boldsymbol{q}_t^{\text{prop}}, \boldsymbol{q}_t^{\text{force}}, \boldsymbol{s}_t, \boldsymbol{z}_t\right]. \tag{4.50}$$

The visual latent is therefore incorporated only through the policy input. Using the augmented observation $\boldsymbol{q}_t^{\text{aug}}$, the policy produces a null space action $\boldsymbol{\xi}_t$ according to

$$\boldsymbol{\xi}_t \sim \pi(\boldsymbol{\xi} \mid \boldsymbol{q}_t^{\text{aug}}). \tag{4.51}$$

Consistent with the KGRL formulation in (4.36), the resulting trajectory mean at query point $\boldsymbol{s}^*$ can be written as

$$\mathbb{E}\big(\boldsymbol{\eta}(\boldsymbol{s}^*)\big) = \boldsymbol{k}^* \boldsymbol{A}\boldsymbol{\mu} + \frac{\beta}{\gamma}\left(\hat{\boldsymbol{k}}^* - \boldsymbol{k}^* \boldsymbol{A}\hat{\boldsymbol{K}}\right)\boldsymbol{\xi} \tag{4.52}$$

Consequently, the trajectory model is not directly influenced by the latent variable $\boldsymbol{z}_t$. Rather, it modifies the policy input, which in turn influences the null space action produced by the policy. In this way, visual information is incorporated at the policy level, while the underlying KGRL and NS-KMP trajectory generation mechanism remains unchanged.

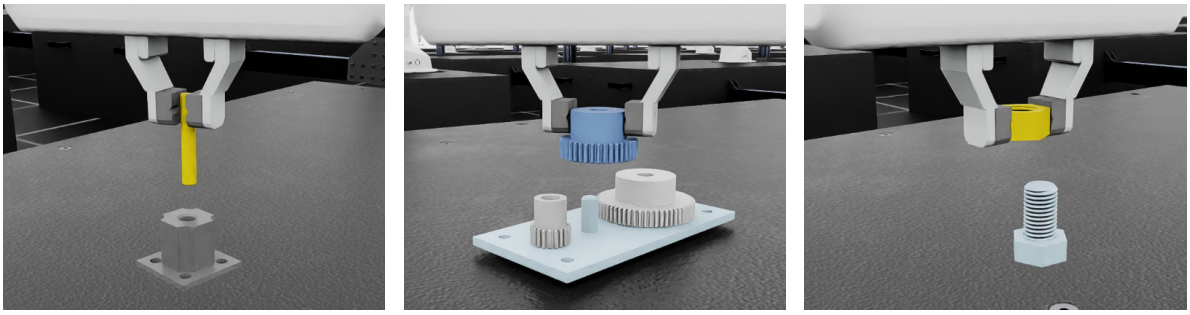
The full Vision-KGRL training procedure is summarized in Algorithm 2.

Algorithm 2 Vision-KGRL

Collect demonstrations $\mathcal{D} = \{\{\boldsymbol{s}_{n,m}, \boldsymbol{\eta}_{n,m}\}_{n=1}^{N}\}_{m=1}^{M}$
Configure hyperparameters λ, β and select kernel $k(\cdot, \cdot)$
Specify planning horizon h
Fit joint distribution $\mathcal{P}(\boldsymbol{s}, \boldsymbol{\eta})$ to demonstrations
Train VAE encoder $\boldsymbol{\mu}_\phi$ via Algorithm 1 and freeze weights
Initialize RL policy $\pi(\boldsymbol{\xi} \mid \boldsymbol{q})$ and its parameters
for each timestep $n = 1, \ldots, N$ **do**
    At state $\boldsymbol{s}_n$, retrieve reference distribution $\mathcal{T}_r \leftarrow \{\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i\}_{i=n}^{n+h}$
    Obtain visual latent: $\boldsymbol{z}_n = \boldsymbol{\mu}_\phi(\boldsymbol{o}_n^{\text{img}})$
    Form augmented policy input: $\boldsymbol{q}_n^{\text{aug}} = [\boldsymbol{q}_n^{\text{prop}}, \boldsymbol{q}_n^{\text{force}}, \boldsymbol{s}_n, \boldsymbol{z}_n]$
    Sample null space action: $\boldsymbol{\xi}_n \sim \pi(\boldsymbol{\xi} \mid \boldsymbol{q}_n^{\text{aug}})$
    Evaluate $\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}_n))$ using (4.36)
    Execute $\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}_n))$ on the robot
    Observe reward r_n
    Update RL policy π
end for

5. Experimental Setup

This thesis uses the PegInsert, GearMesh, and NutThread environments from NVIDIA Isaac Lab [13] as baselines for the experiments. The corresponding environment images are shown in Figure 5.1.



(a) (b) (c)

Figure 5.1.: NVIDIA Isaac Lab environments used in this thesis: (a) PegInsert, (b) GearMesh, (c) NutThread.

The experiments were conducted on these three contact-rich assembly tasks. Each task was evaluated in two environment families, *Factory* and *Forge*. Factory provides proprioceptive observations only, whereas Forge additionally provides force-torque measurements and includes force-related reward terms and action penalty. All training runs used $N_{\text{env}} = 128$ parallel simulation environments. The simulator time step was set to $\Delta t_{\text{sim}} = 1/120\,\text{s}$. With control decimation $d = 8$, actions were applied at

$$\Delta t_{\text{ctrl}} = d\,\Delta t_{\text{sim}} = 1/15\,\text{s}, \tag{5.1}$$

corresponding to a control rate of 15 Hz.

Table 5.1 summarizes the core simulation and episode settings.

Table 5.1.: Simulation and episode configuration used for all experiments.

Setting	Value
Simulator time step Δt_{sim}	0.0083333 s
Control decimation d	8
Control time step Δt_{ctrl}	0.0666667 s
Number of parallel environments N_{env}	128
Episode length (PegInsert)	10 s
Episode length (GearMesh)	10 s
Episode length (NutThread)	15 s

Action Spaces

Two action configurations were evaluated:

- 4D action space: Cartesian translation in (x, y, z) with yaw rotation.
- 6D action space: Cartesian translation in (x, y, z) with $(\text{roll}, \text{pitch}, \text{yaw})$ rotation.

For both action configurations, the policy action $\boldsymbol{a}_t$ was normalized to the bounded range $[-1, 1]^D$, where $D \in \{4, 6\}$ is the action dimension. This normalization was identical for pure RL baselines and KGRL variants.

The action dimensionality was kept consistent across SAC and KGRL variants within each comparison.

For KGRL variants, the normalized policy action represents a null-space action whose effect on the commanded motion is additionally scaled by task-specific scaling factors applied to ξ, which are specified in Section 5.4.

Initial Pose Variations

At the beginning of each episode, the initial poses of both the end-effector and the target (fixed asset) are sampled from uniform distributions. The initial roll and pitch for the end-effector are set to 180° and 0°, respectively. This ensures that the end-effector is always facing down before the simulation starts.

The positional noise and task-specific yaw angles are uniformly sampled across the predefined ranges specified in Table 5.2. These settings are identical for both the Factory and Forge environments.

5.1. Policy State Definitions and Modality Configurations

This thesis evaluates policies under different observation modality configurations. The policy observation is built from proprioception, force sensing, and vision based latent features.

Table 5.2.: Initial Position and Yaw Variations for the End-Effector and Target

Task	Component	Position Variation (x, y, z)	Yaw Range
PegInsert	End-Effector	$[\pm 0.02, \pm 0.02, \pm 0.01]$ m	$[-45°, 45°]$
	Target	$[\pm 0.05, \pm 0.05, \pm 0.05]$ m	$[0°, 360°]$
GearMesh	End-Effector	$[\pm 0.02, \pm 0.02, \pm 0.01]$ m	$[-45°, 45°]$
	Target	$[\pm 0.05, \pm 0.05, \pm 0.05]$ m	$[0°, 15°]$
NutThread	End-Effector	$[\pm 0.02, \pm 0.02, \pm 0.01]$ m	$[90°, 120°]$
	Target	$[\pm 0.05, \pm 0.05, \pm 0.05]$ m	$[120°, 150°]$

Let the TCP pose be denoted by a position $\boldsymbol{x}_t^{\text{tcp}} \in \mathbb{R}^3$ and a unit quaternion $\boldsymbol{q}_t^{\text{tcp}} \in \mathbb{R}^4$. Let the task target (fixed asset) pose be denoted by $\boldsymbol{x}^{\text{tar}} \in \mathbb{R}^3$ and $\boldsymbol{q}^{\text{tar}} \in \mathbb{R}^4$. The proprioceptive observation used in this thesis is defined as

$$\boldsymbol{o}_t^{\text{prop}} = \left[\underbrace{\boldsymbol{x}_t^{\text{tcp}} - \boldsymbol{x}^{\text{tar}}}_{\in \mathbb{R}^3}, \underbrace{\boldsymbol{q}_t^{\text{tcp}}}_{\in \mathbb{R}^4} \right] \in \mathbb{R}^7. \tag{5.2}$$

Force sensing is available only in the Forge test cases:

$$\boldsymbol{o}_t^{\text{force}} = [f_x, f_y, f_z] \in \mathbb{R}^3. \tag{5.3}$$

Vision features are included if vision is enabled. Let $\boldsymbol{o}_t^{\text{img}}$ denote the image observation at time step t. A task specific VAE encoder f_θ produces a latent vector

$$\boldsymbol{o}_t^{\text{vision}} = \boldsymbol{z}_t = f_\theta(\boldsymbol{o}_t^{\text{img}}) \in \mathbb{R}^{16}. \tag{5.4}$$

In addition, all policies receive a normalized time variable $\boldsymbol{s}_t \in [0, 1]$ that represents the progression within the episode. Depending on the experiment, final policy observation is defined as

$$\boldsymbol{q}_t \in \left\{ \left[\boldsymbol{s}_t, \boldsymbol{o}_t^{\text{prop}}\right], \left[\boldsymbol{s}_t, \boldsymbol{o}_t^{\text{prop}}, \boldsymbol{o}_t^{\text{force}}\right], \left[\boldsymbol{s}_t, \boldsymbol{o}_t^{\text{prop}}, \boldsymbol{o}_t^{\text{vision}}\right], \left[\boldsymbol{s}_t, \boldsymbol{o}_t^{\text{prop}}, \boldsymbol{o}_t^{\text{force}}, \boldsymbol{o}_t^{\text{vision}}\right] \right\}, \tag{5.5}$$

where $\boldsymbol{s}_t \in [0, 1]$ denotes the normalized time variable representing the progression within the episode.

Consequently, the policy input dimension depends on the enabled modalities and satisfies

$$\dim(\boldsymbol{q}_t) \in \{8,\ 11,\ 24,\ 27\}. \tag{5.6}$$

The normalized time variable $\boldsymbol{s}_t$ is included for both the standard SAC baselines and the KGRL variants, as removing it from the observation significantly reduces performance in both cases. For KGRL, it also serves as the query input for the KMP trajectory model, where it is a fundamental requirement since the KMP mean and covariance are explicit functions of $\boldsymbol{s}_t$. An ablation study confirming the importance of this input is provided in Appendix A.1.

5.2. Reward Functions

This section documents the reward definitions used in the Factory and Forge environments.

5.2.1. Factory Reward Function

The Factory reward function only includes a sparse success reward:

$$r_{\text{Factory}} = r_{\text{success}}. \tag{5.7}$$

Let the position error between the held asset and the task target (fixed asset) be defined as

$$e_{\text{pos}} = \|\boldsymbol{p}_{\text{held}} - \boldsymbol{p}_{\text{target}}\|_2. \tag{5.8}$$

The success reward is then defined as

$$r_{\text{success}} = \begin{cases} 4 & \text{if } e_{\text{pos}} < 0.04, \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

5.2.2. Forge Reward Function

The Forge reward extends the Factory reward function with additional penalty terms:

$$r_{\text{Forge}} = r_{\text{Factory}} + r_{\text{action\_penalty}} + r_{\text{contact\_penalty}} \tag{5.10}$$

The action penalty discourages excessively large motion commands and is defined as

$$r_{\text{action\_penalty}} = -0.001 \left(e_{\text{pos}}^{\text{act}} + e_{\text{rot}}^{\text{act}} \right). \tag{5.11}$$

The translational component is defined as

$$e_{\text{pos}}^{\text{act}} = \frac{\|\Delta \boldsymbol{p}\|_2}{\varepsilon_{\text{pos}}}, \tag{5.12}$$

$$\Delta \boldsymbol{p} = \boldsymbol{x}_t^{\text{tcp}} - \boldsymbol{x}_t^{\text{cmd}}, \tag{5.13}$$

where $\boldsymbol{x}_t^{\text{tcp}}$ denotes the current TCP position and $\boldsymbol{x}_t^{\text{cmd}}$ denotes the commanded target position.

For the 4D action space, the rotational component penalizes yaw error

$$e_{\text{rot}}^{\text{act}} = \frac{|\Delta \psi|}{\varepsilon_{\text{rot}}}, \tag{5.14}$$

$$\Delta \psi = \psi_t^{\text{tcp}} - \psi_t^{\text{cmd}}. \tag{5.15}$$

For the 6D action space, the rotational penalty additionally includes roll and pitch errors

$$e_{\text{rot}}^{\text{act}} = \frac{|\Delta \phi|}{\varepsilon_{\text{rot}}} + \frac{|\Delta \theta|}{\varepsilon_{\text{rot}}} + \frac{|\Delta \psi|}{\varepsilon_{\text{rot}}}, \tag{5.16}$$

$$\Delta\phi = \phi_t^{\text{tcp}} - \phi_t^{\text{cmd}}, \qquad \Delta\theta = \theta_t^{\text{tcp}} - \theta_t^{\text{cmd}}. \tag{5.17}$$

The normalization coefficients were set to

$$\varepsilon_{\text{pos}} = 0.02, \qquad \varepsilon_{\text{rot}} = 0.097. \tag{5.18}$$

The contact penalty discourages excessive forces:

$$r_{\text{contact\_penalty}} = -\max(0, \|\boldsymbol{f}_{\text{smooth}}\|_2 - \varepsilon_{\text{contact}}), \tag{5.19}$$

with $\varepsilon_{\text{contact}} = 7.5$ and $\boldsymbol{f}_{\text{smooth}}$ the smoothed force measurement.

5.3. Reinforcement Learning Configuration

All policies were trained using Soft Actor-Critic (SAC). The training hyperparameters used in all experiments are listed in Table 5.3.

Table 5.3.: Soft Actor-Critic hyperparameters used in all experiments.

Hyperparameter	Value
Policy	`MlpPolicy`
Replay buffer size	1,000,000
Batch size	256
Learning rate	4×10^{-4}
Learning starts	5,000 steps
Training frequency	1 step
Gradient steps per update	512
Number of critics	2
Network architecture	[512, 256, 128]
Activation function	ELU
Observation normalization	Enabled
Optimizer	AdamW (Optax)
Policy delay	4

5.4. Demonstration Modeling and KMP Configuration

For KGRL variants, demonstrations were modeled using Kernelized Movement Primitives (KMP). For each task, a trajectory distribution was constructed over normalized time s, producing a mean trajectory $\hat{\boldsymbol{\mu}}(s)$ and covariance $\hat{\boldsymbol{\Sigma}}(s)$.

5.4.1. Task-Specific KMP Parameters

The KMP inference parameters used for each task are reported in Table 5.4, where h denotes the prediction horizon and w_{pos}, w_{rot} denote the task-specific scaling weights applied to the translational and rotational components of ξ, respectively.

Table 5.4.: Task-specific KMP inference parameters.

Task	KMP Δt	h	w_{pos}	w_{rot}
PegInsert (4D)	1/150	15	150.0	15.0
PegInsert (6D)	1/150	15	150.0	15.0
GearMesh (4D)	1/150	15	250.0	25.0
GearMesh (6D)	1/150	15	250.0	25.0
NutThread (4D)	1/225	25	200.0	10.0
NutThread (6D)	1/225	25	200.0	10.0

The reinforcement learning policy outputs a normalized action $\boldsymbol{a}_t \in [-1,1]^D$, where $D \in \{4,6\}$ is the action dimension. For KGRL variants, this action is interpreted as a null-space action ξ and scaled before being passed to the LC-NS-KMP module.

Let

$$\boldsymbol{a}_t = [a_{t,x},\, a_{t,y},\, a_{t,z},\, a_{t,\alpha},\, a_{t,\beta},\, a_{t,\gamma}]$$

denote the policy action, where the first three components correspond to translational corrections and the last three components correspond to rotational corrections.

The scaled null-space action $\boldsymbol{\xi}_t$ is defined as

$$\boldsymbol{\xi}_t = \begin{bmatrix} w_{\text{pos}}\, a_{t,x} \\ w_{\text{pos}}\, a_{t,y} \\ w_{\text{pos}}\, a_{t,z} \\ w_{\text{rot}}\, a_{t,\alpha} \\ w_{\text{rot}}\, a_{t,\beta} \\ w_{\text{rot}}\, a_{t,\gamma} \end{bmatrix}, \tag{5.20}$$

where w_{pos} and w_{rot} are the task-specific scaling weights reported in Table 5.4, governing the relative magnitude of translational and rotational corrections that $\boldsymbol{\xi}$ applies to the KMP reference trajectory.

5.5. Pose Representation with KMP

To construct trajectory distributions with KMP, each demonstrated TCP pose is represented in a relative 6D form consisting of a translational and a rotational component. The KMP model predicts these components as functions of normalized time $\boldsymbol{s} \in [0,1]$.

5.5.1. Translational Component of KMP

The translational component is defined as the relative position between the TCP and the target,

$$\boldsymbol{p}_{\text{rel}}(t) = \boldsymbol{p}_{\text{tcp}}(t) - \boldsymbol{p}_{\text{Target}}. \tag{5.21}$$

To introduce uncertainty into the environment, target position is defined as a perturbed version of the nominal fixed asset position,

$$\boldsymbol{p}_{\text{target}} = \boldsymbol{p}_{\text{target}_{\text{nominal}}} + \boldsymbol{\eta}, \tag{5.22}$$

where $\boldsymbol{\eta}$ denotes the initial position noise applied at the beginning of each episode,

$$\eta_i = 0.001\,\epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0,1), \qquad i \in \{x, y, z\}. \tag{5.23}$$

The relative position $\boldsymbol{p}_{\text{rel}}(t) \in \mathbb{R}^3$ serves as the demonstration data for the translational part of KMP.

During inference, KMP predicts a 3D translational output,

$$\hat{\boldsymbol{y}}_p(\boldsymbol{s}) = \hat{\boldsymbol{p}}_{\text{rel}}(\boldsymbol{s}) \in \mathbb{R}^3, \tag{5.24}$$

which represents the estimated relative position.

The absolute TCP position is then reconstructed as

$$\hat{\boldsymbol{p}}_{\text{tcp}}(\boldsymbol{s}) = \boldsymbol{p}_{\text{target}} + \hat{\boldsymbol{y}}_p(\boldsymbol{s}). \tag{5.25}$$

5.5.2. Rotational Component of KMP

The rotational component is defined relative to a reference orientation $\boldsymbol{q}_{\text{ref}}$,

$$\boldsymbol{q}_{\text{rel}}(t) = \boldsymbol{q}_{\text{ref}}^{-1} \otimes \boldsymbol{q}_{\text{tcp}}(t). \tag{5.26}$$

The reference orientation is defined as

$$\boldsymbol{q}_{\text{ref}} = \boldsymbol{q}(\pi,\, 0,\, \psi_{\text{ref}}), \tag{5.27}$$

with

$$\psi_{\text{ref}} = \begin{cases} 0, & \text{PegInsert and GearMesh,} \\ -40^\circ, & \text{NutThread.} \end{cases} \tag{5.28}$$

To obtain a minimal 3D representation, the relative quaternion is mapped to a 3D rotation vector using the logarithmic map,

$$\boldsymbol{r}(t) = \log\big(\boldsymbol{q}_{\text{rel}}(t)\big) \in \mathbb{R}^3, \tag{5.29}$$

where $\boldsymbol{r}(t)$ corresponds to the axis-angle representation of the rotation, with its direction indicating the rotation axis and its magnitude the rotation angle.

The vector $\boldsymbol{r}(t)$ serves as the demonstration data for the rotational part of KMP.

During inference, KMP predicts

$$\hat{\boldsymbol{y}}_r(\boldsymbol{s}) = \hat{\boldsymbol{r}}(\boldsymbol{s}) \in \mathbb{R}^3, \tag{5.30}$$

which represents the estimated relative rotation in axis-angle form.

This prediction is mapped back to a quaternion using the exponential map,

$$\boldsymbol{q}_{\text{rel}}(\boldsymbol{s}) = \exp\big(\hat{\boldsymbol{y}}_r(\boldsymbol{s})\big), \tag{5.31}$$

and the TCP orientation is reconstructed as

$$\boldsymbol{q}_{\text{tcp}}(\boldsymbol{s}) = \boldsymbol{q}_{\text{ref}} \otimes \boldsymbol{q}_{\text{rel}}(\boldsymbol{s}). \tag{5.32}$$

In the 4D action space setting, roll and pitch are fixed to their reference values. Therefore, the relative rotation is dominated by a rotation about the z axis of the reference frame, and the logarithmic map yields

$$\boldsymbol{r}(t) = \begin{bmatrix} 0 \\ 0 \\ \Delta\psi(t) \end{bmatrix}, \tag{5.33}$$

where the third component represents the relative yaw deviation.

5.5.3. Combined KMP Output Representation

For each normalized time input $\boldsymbol{s} \in [0, 1]$, the complete KMP prediction is formed by stacking the translational and rotational outputs:

$$\hat{\boldsymbol{y}}(\boldsymbol{s}) = \begin{bmatrix} \hat{\boldsymbol{y}}_p(\boldsymbol{s}) \\ \hat{\boldsymbol{y}}_r(\boldsymbol{s}) \end{bmatrix} = \begin{bmatrix} \hat{x}(\boldsymbol{s}) \\ \hat{y}(\boldsymbol{s}) \\ \hat{z}(\boldsymbol{s}) \\ \hat{r}_x(\boldsymbol{s}) \\ \hat{r}_y(\boldsymbol{s}) \\ \hat{r}_z(\boldsymbol{s}) \end{bmatrix}. \tag{5.34}$$

The first three components correspond to the translational part, that is, the predicted relative TCP position, whereas the last three components correspond to the rotational part represented in axis-angle form.

5.6. Vision Configuration

In the vision augmented experiments, image observations were encoded into a compact latent representation and appended to the policy observation, as introduced in Section 5.1. For each task, a task specific variational autoencoder was trained offline and its encoder was later used during reinforcement learning to produce a latent vector of dimension 16. During policy training, the encoder weights remained fixed.

5.6.1. VAE Network Architecture

The architecture of the variational autoencoder is illustrated in Fig. 5.2. It follows a symmetric convolutional encoder-decoder design operating on RGB images of resolution 256×256.

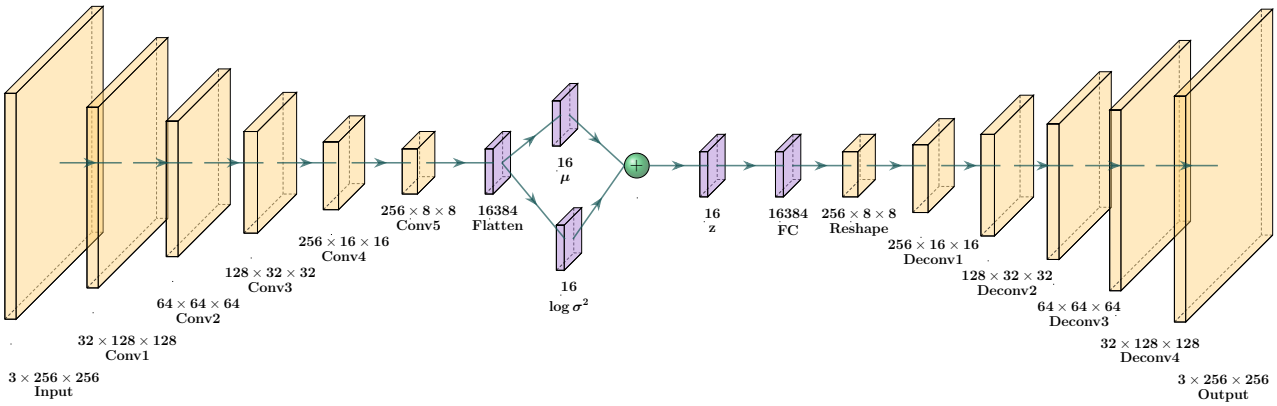


Figure 5.2.: Architecture of the variational autoencoder used for visual feature extraction.

The encoder is composed of five convolutional layers, each with a kernel size of 4, a stride of 2, and a padding of 1. These layers gradually decrease the spatial resolution as the number of feature channels increases. Each layer is succeeded by a Leaky ReLU activation and Group Normalization with 8 groups. The mean $\boldsymbol{\mu} \in \mathbb{R}^{16}$ and the log variance $\log \boldsymbol{\sigma}^2 \in \mathbb{R}^{16}$ are obtained by flattening and mapping the final feature map to the parameters of the latent distribution. The reparameterization trick is then employed to obtain a latent sample $\boldsymbol{z}$.

The image is reconstructed to its original resolution by the decoder, which mirrors the encoder with transposed convolutional layers. Intermediate decoder layers also employ Group Normalization and Leaky ReLU, while the final layer employs a sigmoid activation to generate outputs within the $[0, 1]$ range.

The visual latent representation is exclusively the encoder mean $\boldsymbol{\mu}$, which is appended to the policy observation during reinforcement learning. This results in a vision feature that is compact, while simultaneously maintaining a manageable downstream policy input dimension.

5.6.2. VAE Hyperparameters

The variational autoencoder was trained separately for the PegInsert, GearMesh, and NutThread tasks. The hyperparameters used for all VAE training runs are reported in Table 5.5.

Table 5.5.: VAE hyperparameters used for visual representation learning.

Parameter	Value
Number of training images	130,000
Encoder-decoder depth	5 convolutional layers ($k = 4$, $s = 2$)
Activation function	Leaky ReLU
Loss function	L2 (MSE)
Optimizer	Adam, lr $= 1 \times 10^{-3}$
Batch size	128
Epochs	400
Augmentations	Horizontal flip (50%), brightness shift (-15 to 15, 30%), Gaussian blur (σ = 0-0.7, 30%)

These settings were chosen to provide stable reconstruction quality while learning a compact latent representation that could later be integrated into the reinforcement learning policy. The VAE training curves are reported in Chapter 6.

5.7. Camera Placement Configuration

Task-specific camera placement presets were used in the experiments to ensure consistent and appropriate viewpoints for vision-based tasks. The camera configurations that were used for each environment are summarized in Table 5.6.

Table 5.6.: Camera placement and configuration for each task.

Task	Position (m)	Roll (°)	Pitch (°)	Yaw (°)
PegInsert	(0.08, 0.0, 0.00)	-20	0	0
GearMesh	(0.0, 0.0, 0.05)	0	0	90
NutThread	(0.0, 0.0, 0.05)	0	0	90

These settings define the camera position and orientation relative to the task environments to provide a consistent view of the manipulated objects. The resolution of each camera was fixed at 256×256 pixels to balance computational cost and visual detail during training and evaluation.

PegInsert NutThread GearMesh

Figure 5.3.: Task specific camera placements used in the vision based experiments. The viewpoints were chosen to provide consistent visual observations of the manipulated objects and interaction regions for each task.

5.8. Method Configurations Evaluated

The experiments evaluate multiple reinforcement learning configurations across three tasks (PegInsert, GearMesh, NutThread), two simulation environments (Factory and Forge), two action spaces (4D and 6D), and four learning variants (SAC, Vision-SAC, KGRL, Vision-KGRL).

Table 5.7.: Evaluated experimental configurations.

Category	Options
Tasks	PegInsert, GearMesh, NutThread
Environments	Factory, Forge
Action spaces	4D, 6D
Learning variants	SAC, Vision-SAC, KGRL, Vision-KGRL

The combination of these factors results in a total of 48 experimental cases.

6. Results

This chapter presents the empirical evaluation of the proposed vision-augmented Kernelized Guided Reinforcement Learning (KGRL) framework. The results are structured as follows. First, the learned KMP trajectory distributions are illustrated. Second, the VAE training curves are reported. Finally, the learning performance of all policy variants is analyzed across the three assembly tasks: PegInsert, GearMesh, and NutThread.

All reported values correspond to mean performance over 10 random seeds. Standard deviation across seeds is reported to quantify statistical variability.

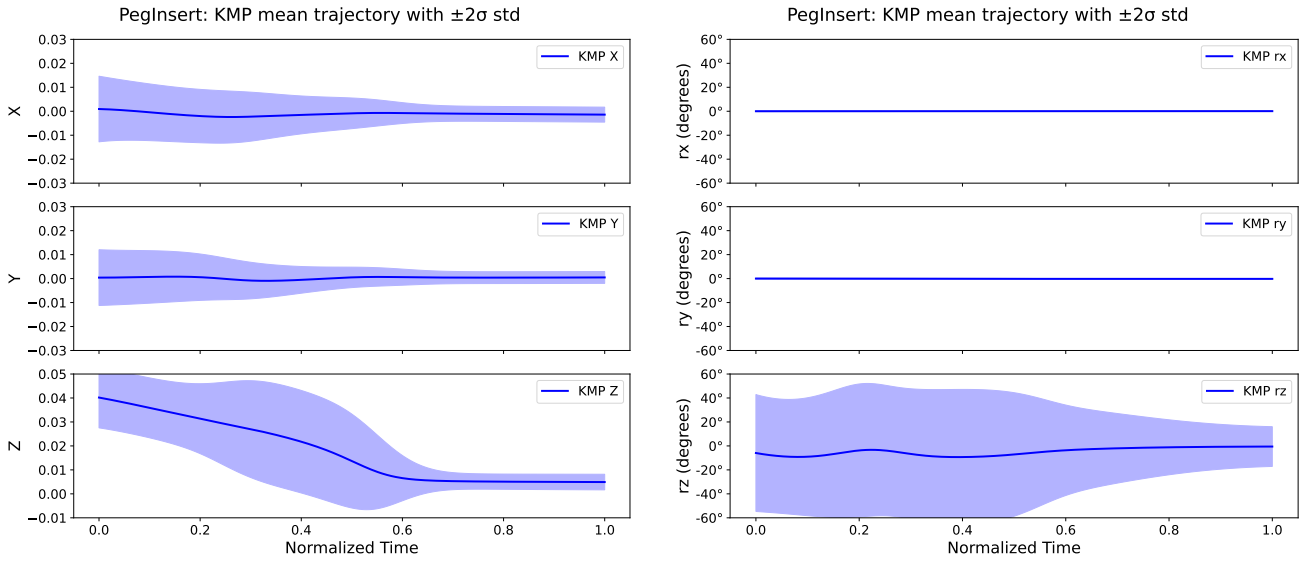
6.1. KMP Mean and Variance Visualization

To examine the learned trajectory distributions before reinforcement learning, the predicted KMP mean and covariance were visualized over normalized time (Figures 6.1–6.6). The plots show the translational and rotational parts of the predicted KMP output that were discussed in Section 5.5. The first three dimensions depict the relative position of the TCP, and the last three dimensions represent the relative rotation in axis-angle form.

The solid curves in the figures represent the predicted mean trajectory of KMP. The shaded areas show the $\pm 2\sigma$ envelope that was calculated from the diagonal entries of the predicted covariance matrix. These plots are utilized to evaluate the smoothness, physical plausibility, and consistency of the learned trajectory distribution with the demonstrated skill.

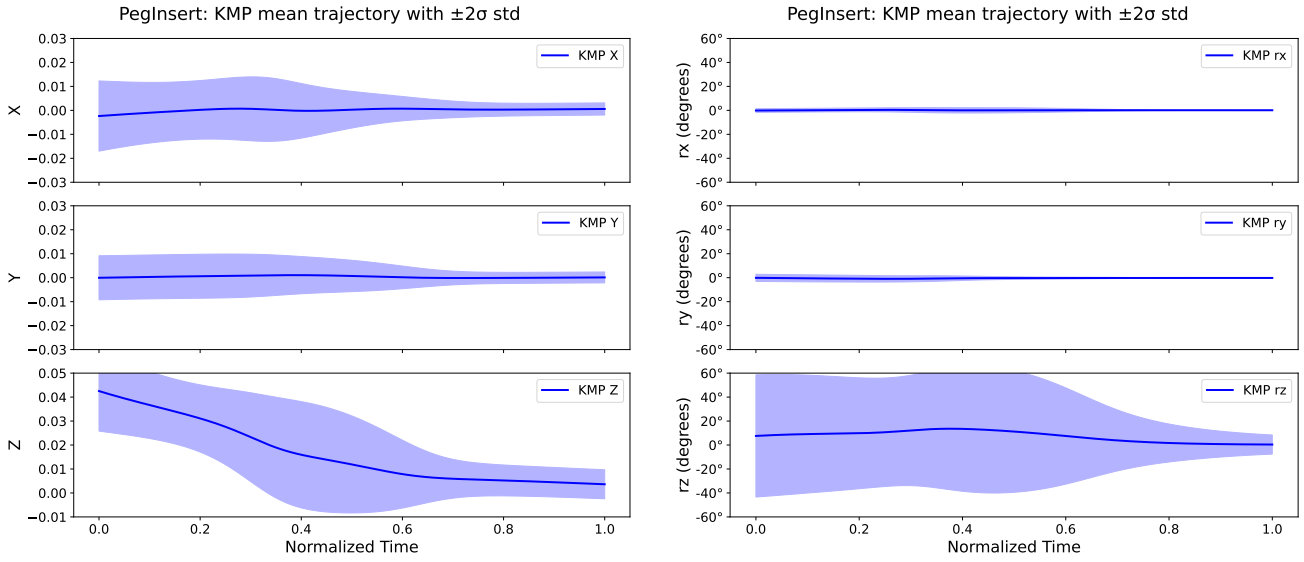
In the 4D action space configuration, the rotational part is solely influenced by the yaw component, as roll and pitch are maintained at their reference values throughout execution. Thus, the third rotational component is the predominant factor in the orientation plots. In the 6D framework, all three rotational components vary over normalized time.

For the PegInsert task, the KMP predictions for the 4D and 6D settings are shown in Figures 6.1 and 6.2.



(a) (b)

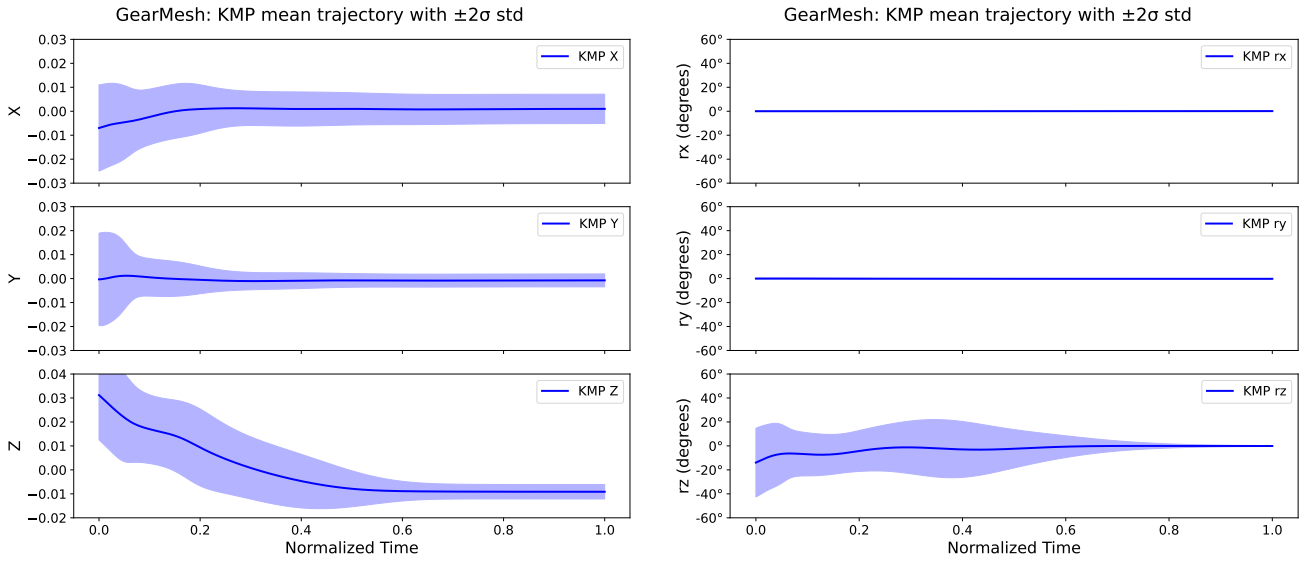
Figure 6.1.: KMP mean trajectory with $\pm 2\sigma$ envelope for PegInsert with 4D action space over normalized time. (a) Translational components, (b) Rotational components.



(a) (b)

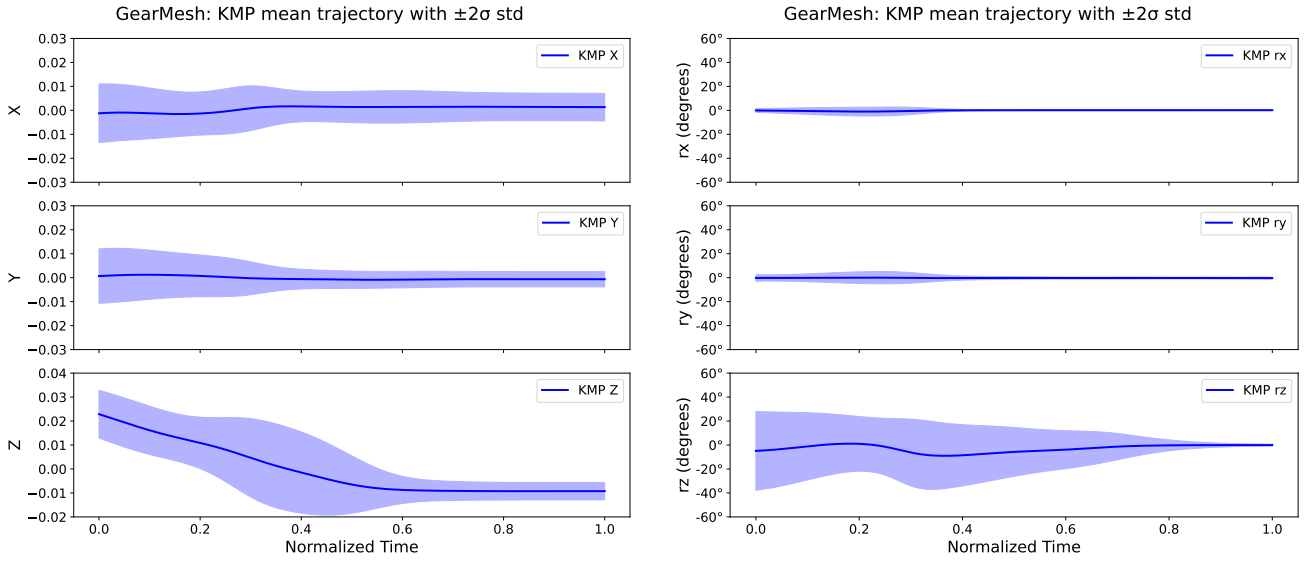
Figure 6.2.: KMP mean trajectory with $\pm 2\sigma$ envelope for PegInsert with 6D action space over normalized time. (a) Translational components, (b) Rotational components.

For the GearMesh task, the KMP predictions for the 4D and 6D settings are shown in Figures 6.3 and 6.4.



(a) (b)

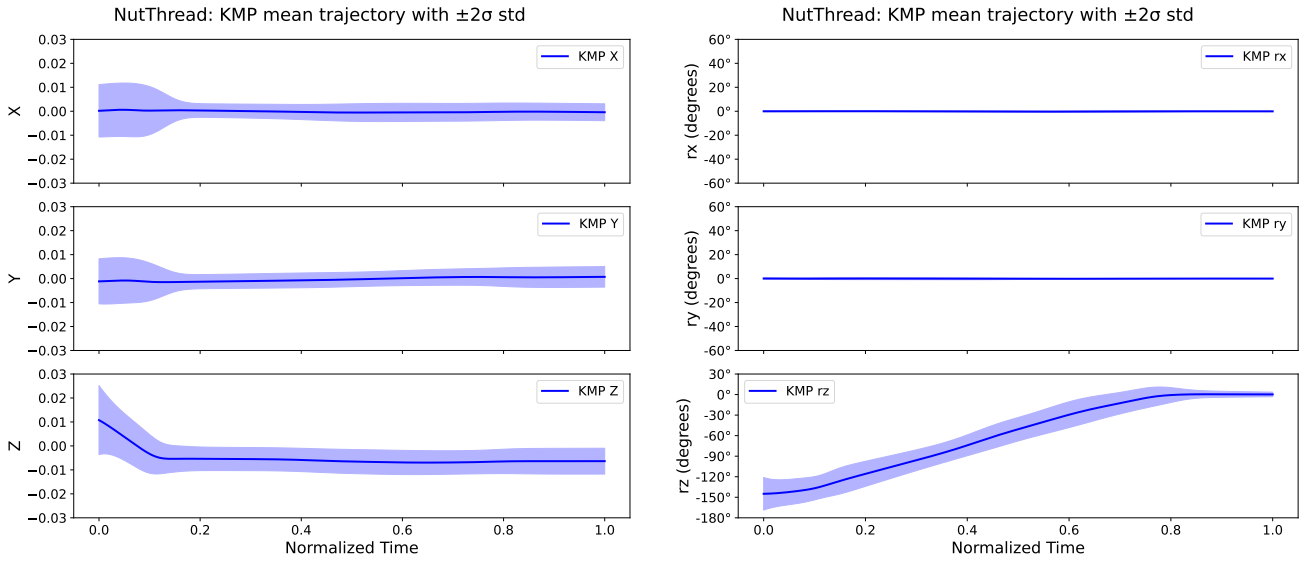
Figure 6.3.: KMP mean trajectory with $\pm 2\sigma$ envelope for GearMesh with 4D action space over normalized time. (a) Translational components, (b) Rotational components.



(a) (b)

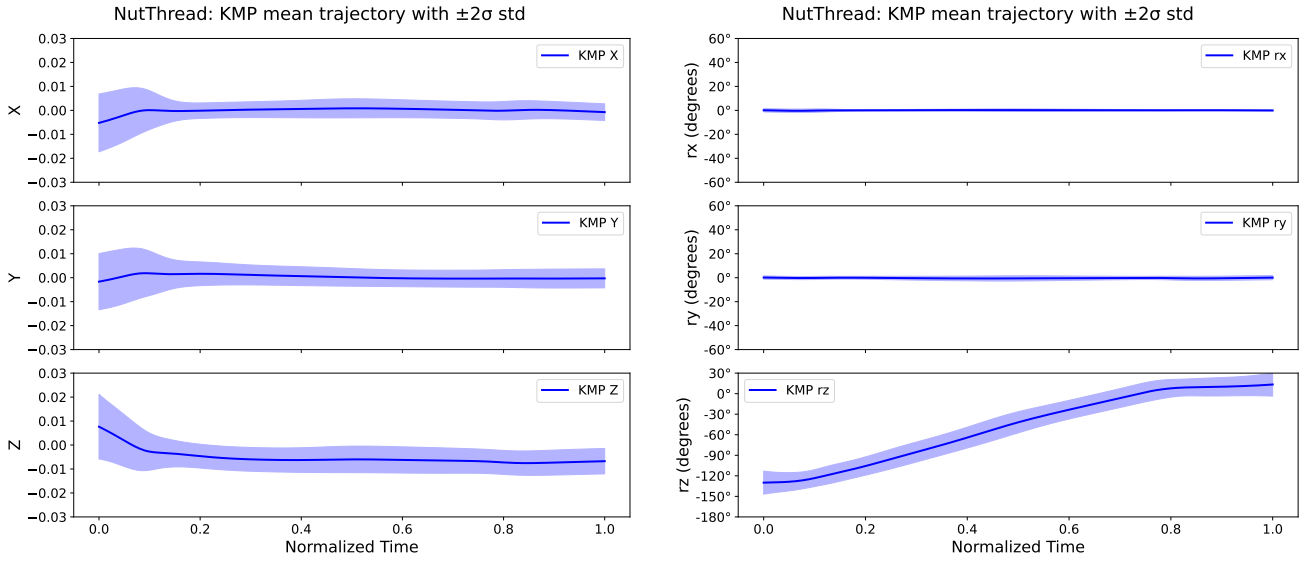
Figure 6.4.: KMP mean trajectory with $\pm 2\sigma$ envelope for GearMesh with 6D action space over normalized time. (a) Translational components, (b) Rotational components.

For the NutThread task, the KMP predictions for the 4D and 6D settings are shown in Figures 6.5 and 6.6.



(a) (b)

Figure 6.5.: KMP mean trajectory with $\pm 2\sigma$ envelope for NutThread with 4D action space over normalized time. (a) Translational components, (b) Rotational components.



(a) (b)

Figure 6.6.: KMP mean trajectory with $\pm 2\sigma$ envelope for NutThread with 6D action space over normalized time. (a) Translational components, (b) Rotational components.

The VAE training curves are presented in the following section.

6.2. VAE Training Curves

To analyze the optimization behavior of the variational autoencoder, the KL loss, reconstruction loss, and total loss were tracked throughout training for the PegInsert, GearMesh, and NutThread datasets. Figures 6.7, 6.8, and 6.9 show the KL loss, reconstruction loss, and total loss, respectively. In each figure, (a) corresponds to the training split and (b) corresponds to the validation split.

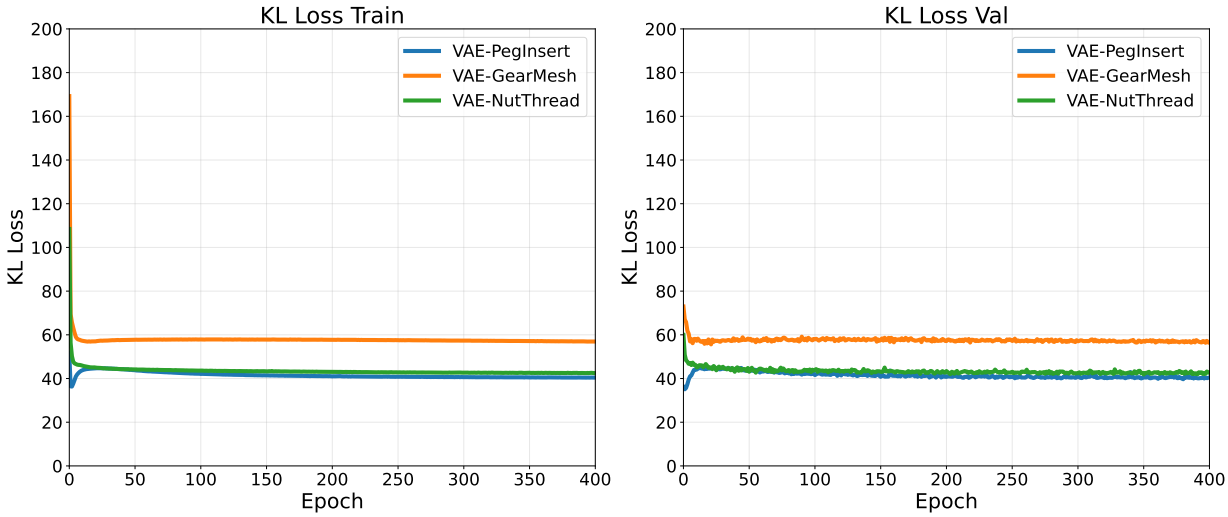


Figure 6.7.: KL loss curves of the variational autoencoder for PegInsert, GearMesh, and NutThread. (a) Training loss, (b) validation loss.

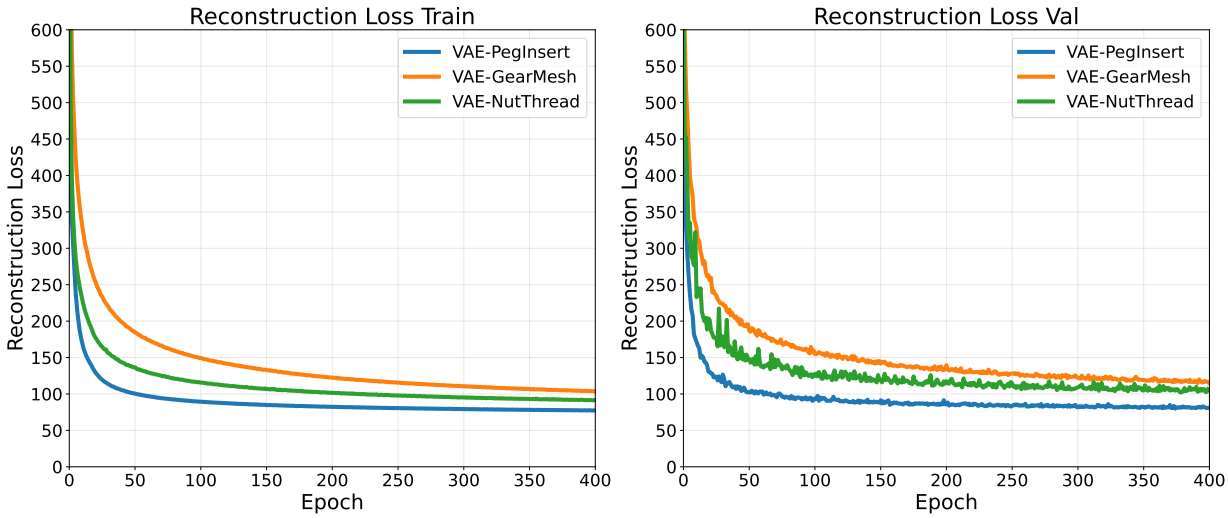


Figure 6.8.: Reconstruction loss curves of the variational autoencoder for PegInsert, GearMesh, and NutThread. (a) Training loss, (b) validation loss.

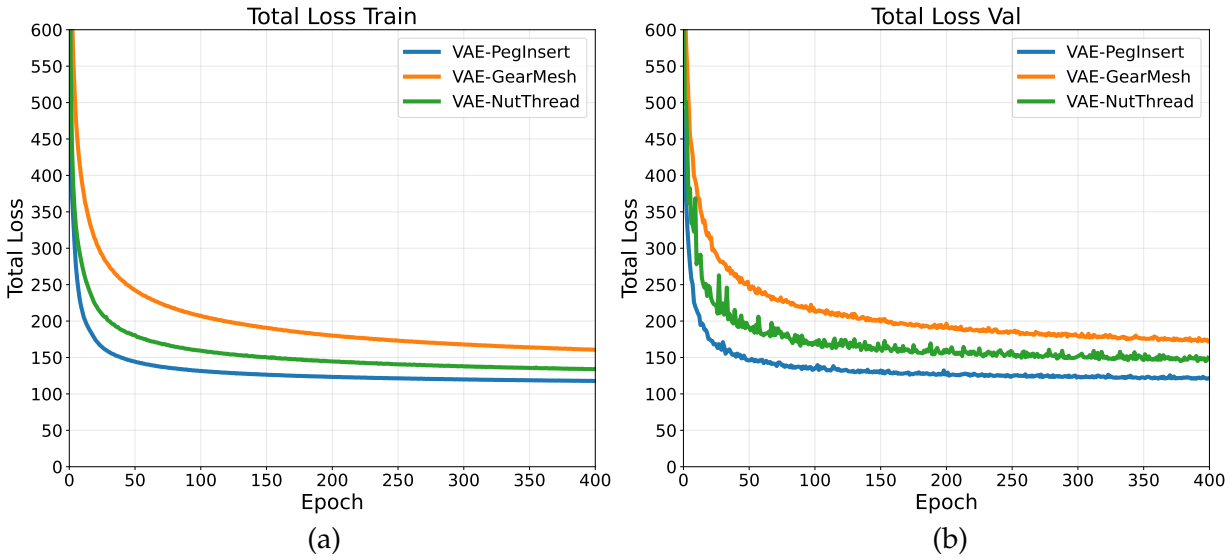


Figure 6.9.: Total loss curves of the variational autoencoder for PegInsert, GearMesh, and NutThread. (a) Training loss, (b) validation loss.

Collectively, these plots offer a comprehensive perspective on the convergence of the

VAE across the three tasks. The KL loss measures the regularization impact on the latent distribution, the reconstruction loss indicates the precision of the image reconstructions, and the total loss encompasses the overall optimization goal during training. Analyzing the training and validation curves assists in evaluating the consistency of the learned visual representation across the three assembly environments.

The learning performance of all policy variants is presented in Section 6.3.

6.3. Learning Performance

The following figures show the learning performance of the SAC and KGRL policies across the tasks in both 4D and 6D action spaces. For each task, representative frames from a successful and an unsuccessful trial are shown before the learning curves.

6.3.1. Evaluation Metrics

Performance was evaluated using the following metrics:

- **Mean reward**: The mean reward is the average reward obtained by all parallel agents (128 robots) during the evaluation episodes. This statistic provides insight into both sample efficiency and stability of the learning process. The convergence of the mean reward curves indicates how quickly the policies learn effective behaviors.
- **Success rate (%)**: The percentage of successful task completions over the evaluation episodes, reflecting the overall effectiveness of the learned policy in achieving the desired task outcomes. Success was defined according to task-specific geometric and contact criteria provided by the simulation environment.
- **Mean contact force magnitude**: This metric measures the average magnitude of the interaction forces generated by the robots during an episode. It is computed by summing the forces applied by all robots at every timestep and normalizing by the total number of timesteps in one episode:

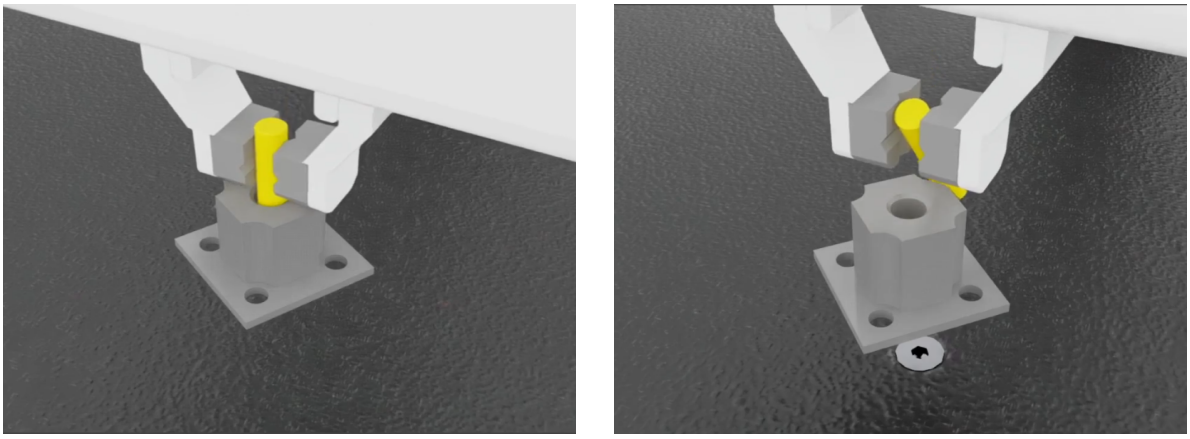
$$\bar{F} = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \|\mathbf{F}_{i,t}\|, \tag{6.1}$$

  where $N = 128$ is the number of parallel robots and T denotes the number of timesteps in an episode. This metric provides an indication of how aggressively the policy interacts with the environment during task execution.

6.3.2. PegInsert

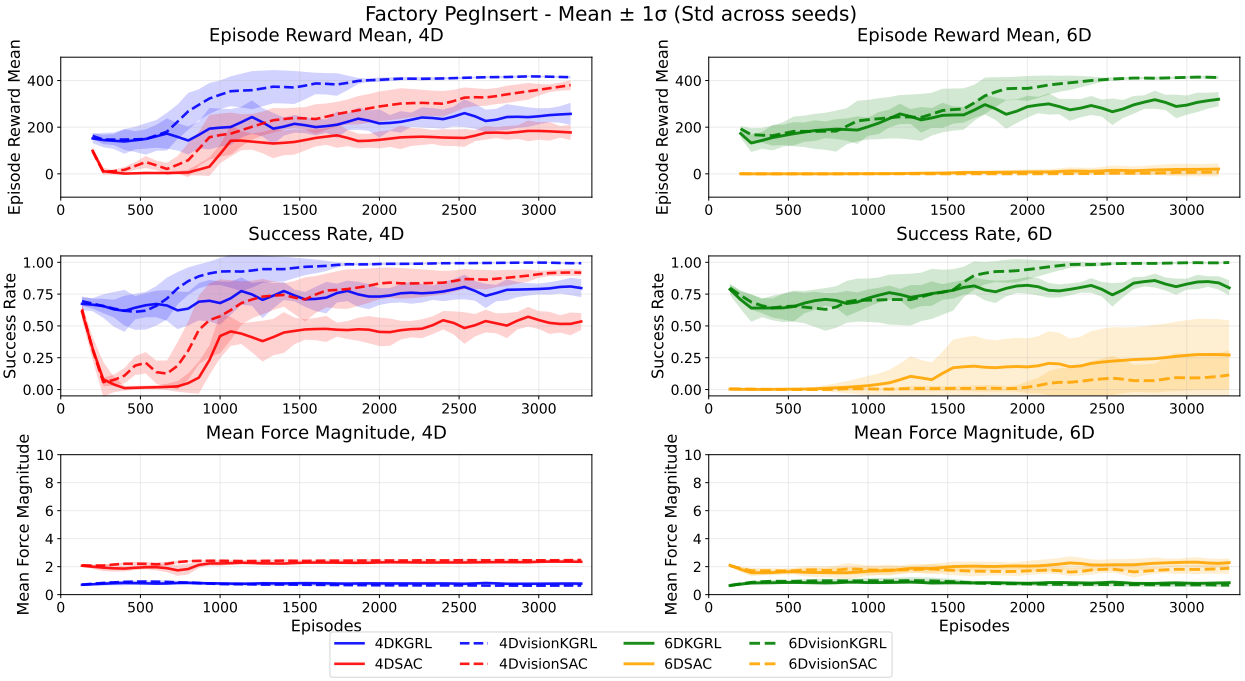
This section reports the results for the PegInsert task. Figure 6.10 shows representative frames from the task execution, where (a) depicts a successful insertion and (b) a failed attempt. Figures 6.11 and 6.12 report the learning curves for mean reward, success rate, and

mean contact force magnitude in the Factory and Forge environments, respectively, for both the 4D and 6D action space configurations.



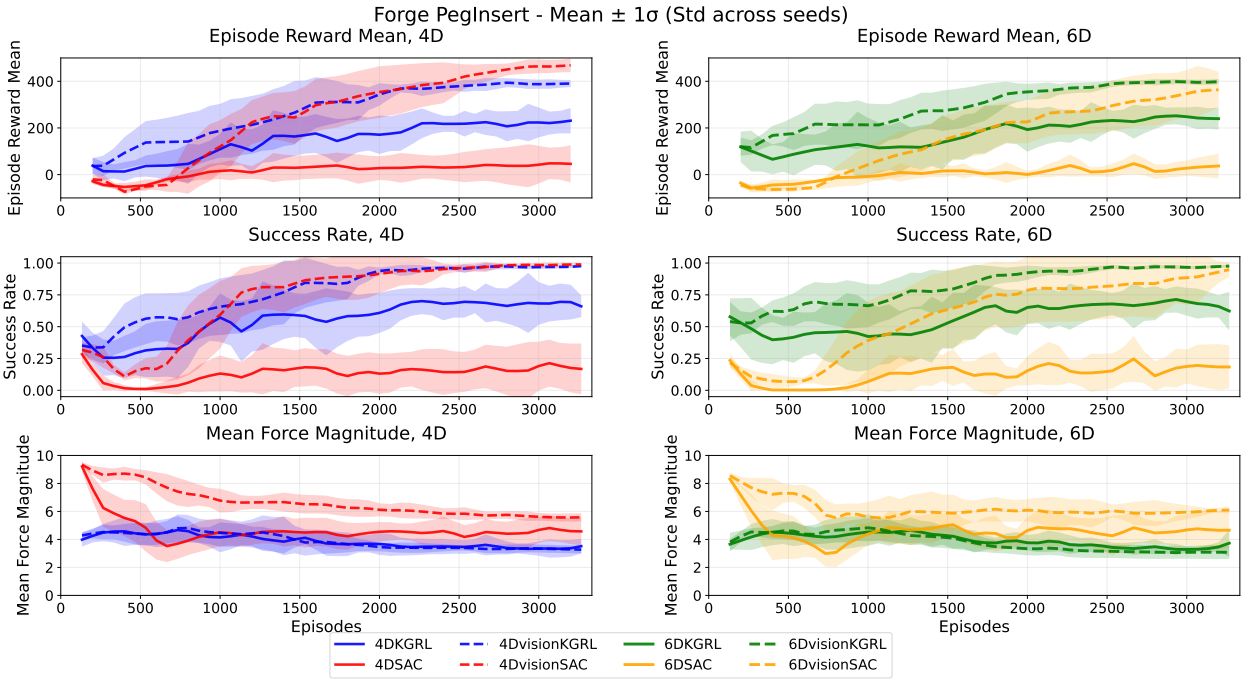
(a) (b)

Figure 6.10.: Representative trials for the PegInsert task. (a) Successful insertion, (b) failed attempt.



(a) (b)

Figure 6.11.: Performance metrics for the Factory PegInsert task. (a) 4D action space, (b) 6D action space.

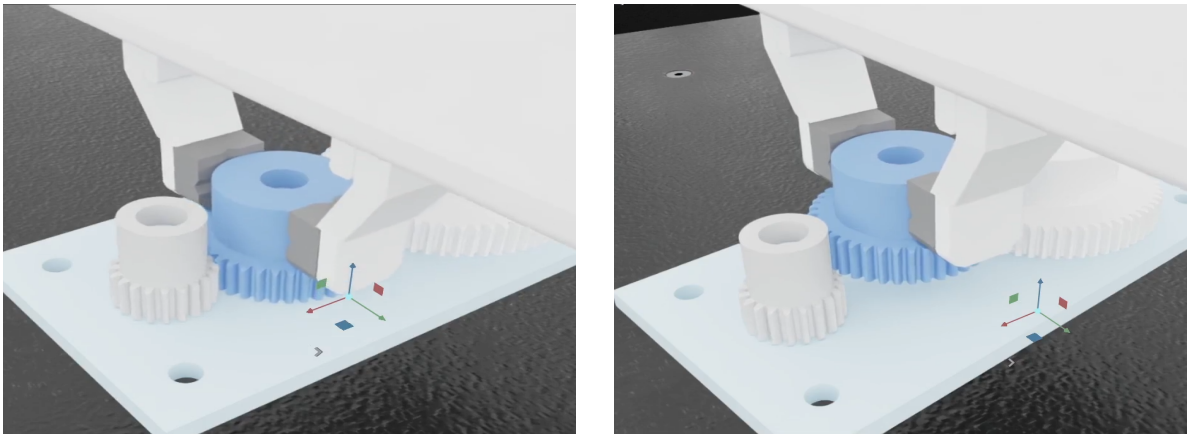


(a) (b)

Figure 6.12.: Performance metrics for the Forge PegInsert task. (a) 4D action space, (b) 6D action space.

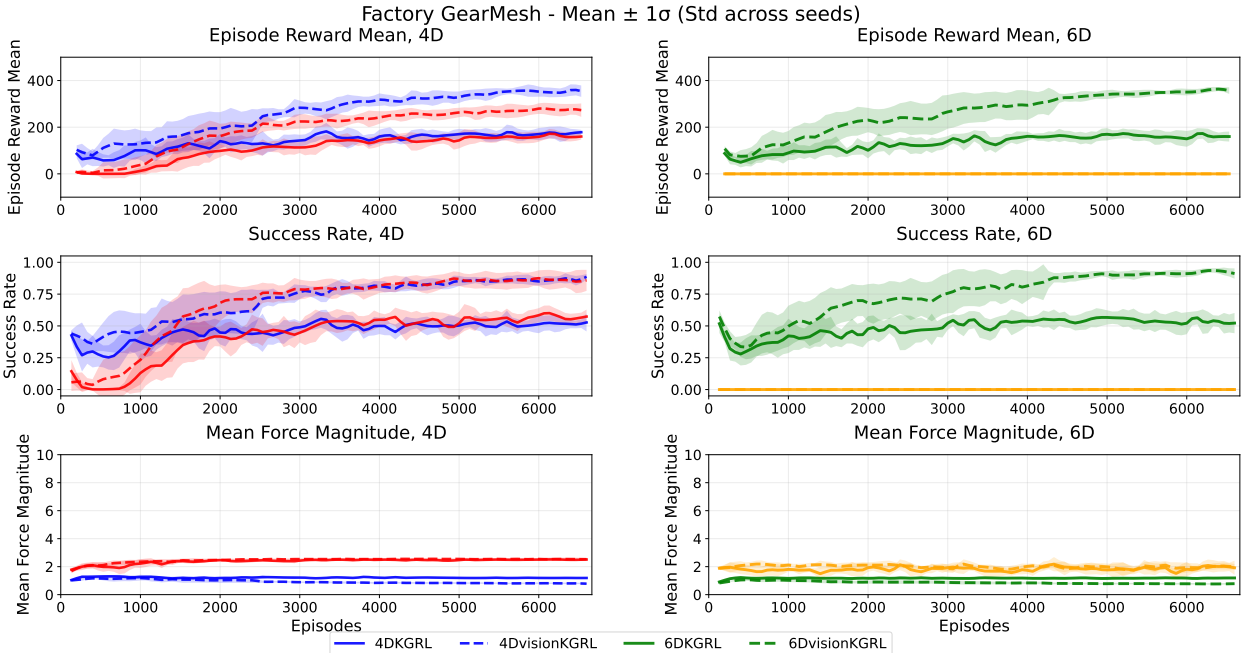
6.3.3. GearMesh

This section reports the results for the GearMesh task. Figure 6.13 shows representative frames from the task execution, where (a) depicts a successful meshing and (b) a failed attempt. Figures 6.14 and 6.15 report the learning curves for mean reward, success rate, and mean contact force magnitude in the Factory and Forge environments, respectively, for both the 4D and 6D action space configurations.



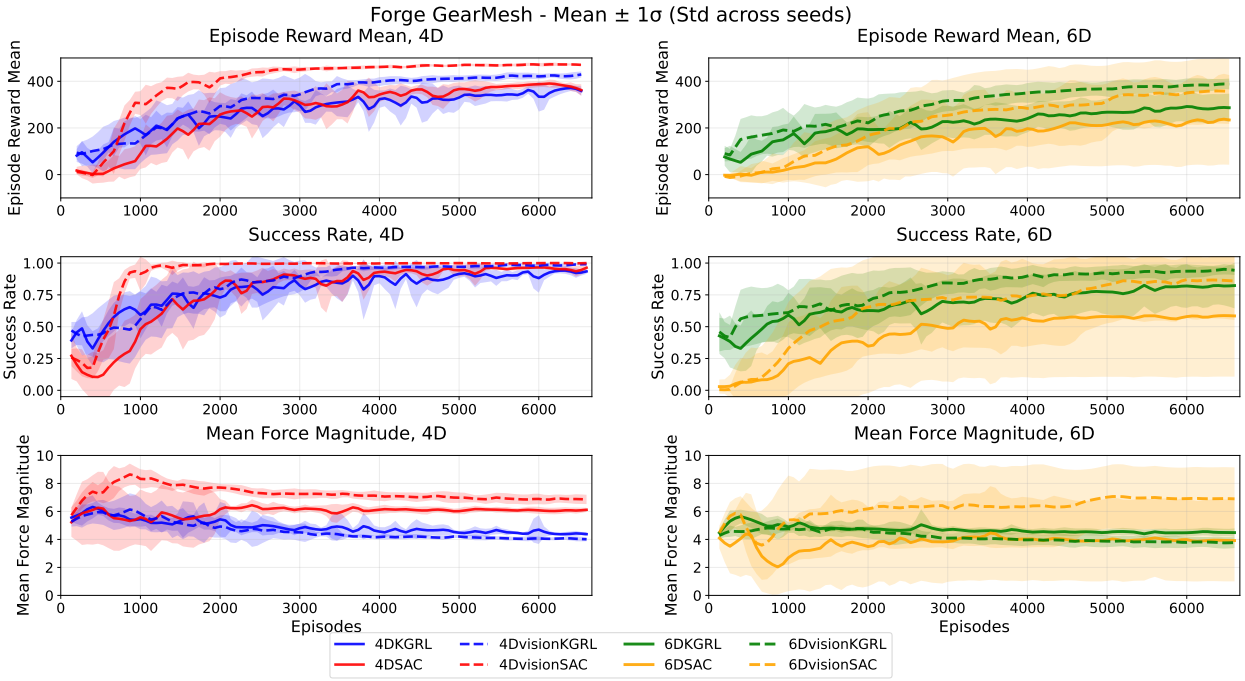
(a) (b)

Figure 6.13.: Representative trials for the GearMesh task. (a) Successful meshing, (b) failed attempt.



(a) (b)

Figure 6.14.: Performance metrics for the Factory GearMesh task. (a) 4D action space, (b) 6D action space.

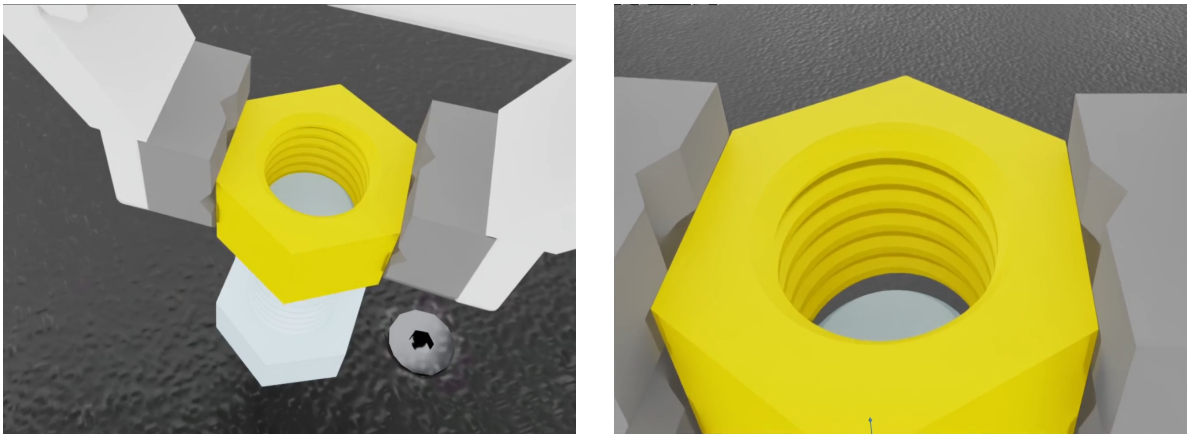


(a) (b)

Figure 6.15.: Performance metrics for the Forge GearMesh task. (a) 4D action space, (b) 6D action space.

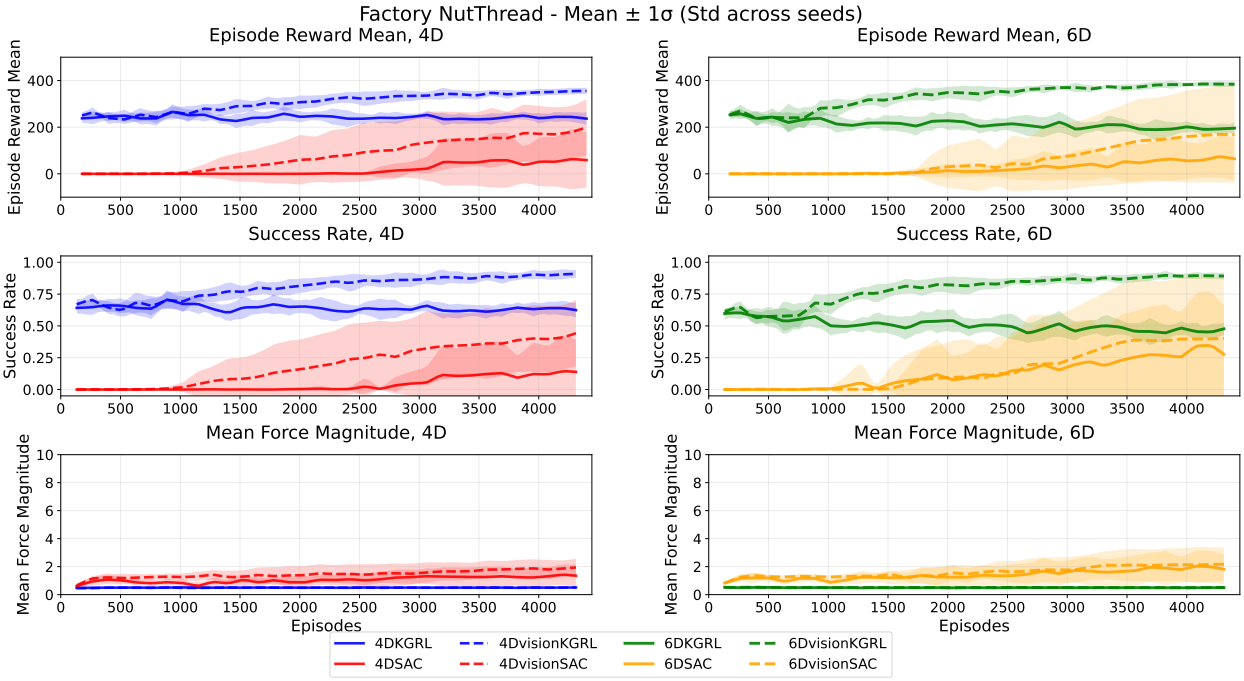
6.3.4. NutThread

This section reports the results for the NutThread task. Figure 6.16 shows representative frames from the task execution, where (a) depicts a successful threading and (b) a failed attempt. Figures 6.17 and 6.18 report the learning curves for mean reward, success rate, and mean contact force magnitude in the Factory and Forge environments, respectively, for both the 4D and 6D action space configurations.



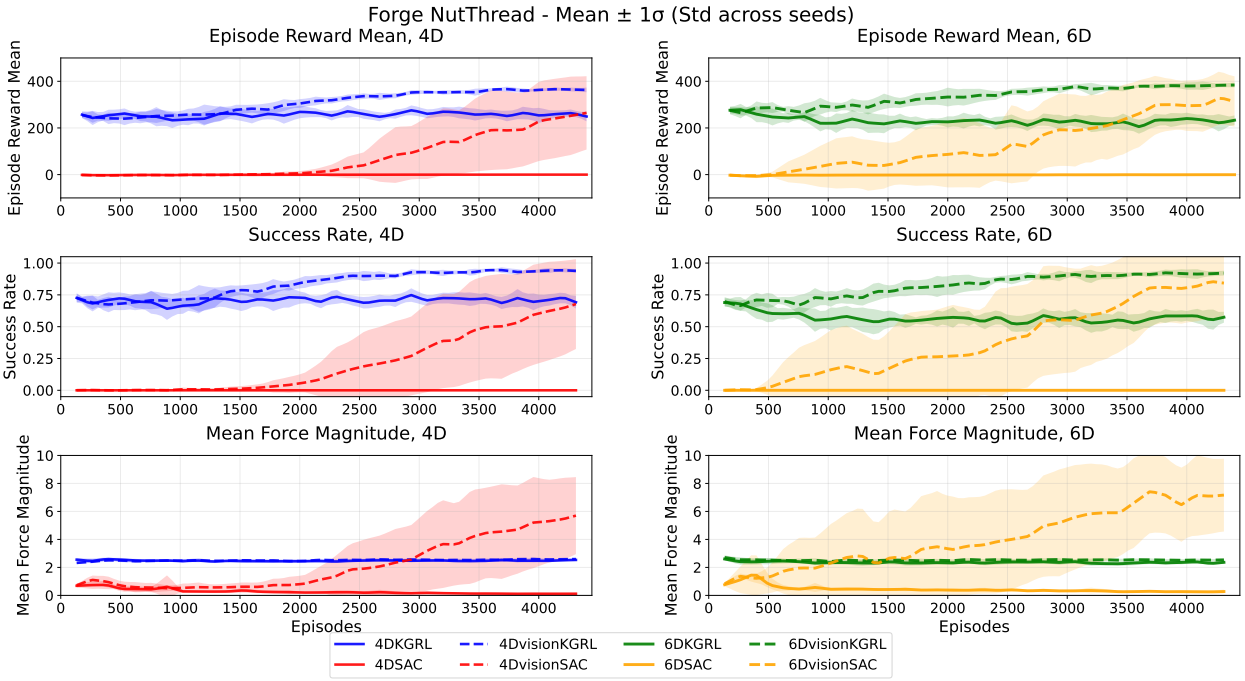
(a) (b)

Figure 6.16.: Representative trials for the NutThread task. (a) Successful threading, (b) failed attempt.



(a) (b)

Figure 6.17.: Performance metrics for the Factory NutThread task. (a) 4D action space, (b) 6D action space.



(a) (b)

Figure 6.18.: Performance metrics for the Forge NutThread task. (a) 4D action space, (b) 6D action space.

Tables 6.2, 6.1, 6.3, and 6.4 report the final mean rewards, success rates, mean contact force magnitudes, and times to first success across all tasks, environments, and action space configurations.

Table 6.1.: Final mean reward (mean ± std, across 10 seeds) across all tasks, environments, and action space configurations. Best configurations are highlighted in bold.

Task	Method	Factory		Forge	
		4D	6D	4D	6D
PegInsert	KGRL	**256.99 ± 47.48**	**319.60 ± 30.63**	**231.10 ± 54.48**	**239.70 ± 47.09**
	SAC	177.43 ± 32.50	20.48 ± 24.76	45.96 ± 79.34	36.93 ± 52.95
	VisionKGRL	**414.25 ± 5.60**	**413.07 ± 5.83**	390.53 ± 15.50	**398.15 ± 11.83**
	VisionSAC	380.29 ± 21.33	7.08 ± 18.59	**468.50 ± 29.22**	363.61 ± 76.54
GearMesh	KGRL	**178.38 ± 10.26**	**159.87 ± 21.56**	358.57 ± 19.75	**286.63 ± 60.40**
	SAC	160.01 ± 13.17	0.00 ± 0.00	**362.43 ± 24.98**	234.53 ± 192.21
	VisionKGRL	**354.44 ± 24.53**	**356.85 ± 14.22**	429.31 ± 12.83	**388.37 ± 23.11**
	VisionSAC	273.09 ± 28.76	0.00 ± 0.00	**470.34 ± 3.73**	356.93 ± 139.84
NutThread	KGRL	**236.56 ± 26.63**	**195.61 ± 24.15**	**249.06 ± 20.91**	**233.08 ± 20.60**
	SAC	58.83 ± 118.72	64.23 ± 94.54	−0.16 ± 0.13	−0.46 ± 0.24
	VisionKGRL	**355.28 ± 13.88**	**384.03 ± 12.99**	**362.92 ± 10.15**	**383.94 ± 10.28**
	VisionSAC	198.74 ± 121.05	167.82 ± 205.68	264.63 ± 157.27	313.09 ± 108.26

Table 6.2.: Final success rates (mean ± std, across 10 seeds) across all tasks, environments, and action space configurations. Best configurations are highlighted in bold.

Task	Method	Factory		Forge	
		4D	6D	4D	6D
PegInsert	KGRL	**0.797 ± 0.071**	**0.799 ± 0.060**	**0.660 ± 0.086**	**0.623 ± 0.152**
	SAC	0.535 ± 0.066	0.272 ± 0.272	0.168 ± 0.200	0.184 ± 0.170
	VisionKGRL	**0.993 ± 0.005**	**0.998 ± 0.002**	0.976 ± 0.007	**0.976 ± 0.011**
	VisionSAC	0.919 ± 0.025	0.114 ± 0.192	**0.989 ± 0.010**	0.948 ± 0.058
GearMesh	KGRL	0.527 ± 0.071	**0.522 ± 0.080**	0.934 ± 0.019	0.823 ± 0.179
	SAC	**0.574 ± 0.055**	0.000 ± 0.000	**0.963 ± 0.022**	**0.585 ± 0.478**
	VisionKGRL	**0.886 ± 0.022**	**0.912 ± 0.041**	0.991 ± 0.008	**0.943 ± 0.037**
	VisionSAC	0.858 ± 0.083	0.000 ± 0.000	**1.000 ± 0.000**	0.859 ± 0.306
NutThread	KGRL	**0.624 ± 0.057**	**0.477 ± 0.050**	**0.692 ± 0.050**	**0.574 ± 0.043**
	SAC	0.139 ± 0.255	0.278 ± 0.394	0.000 ± 0.000	0.000 ± 0.000
	VisionKGRL	**0.909 ± 0.031**	**0.892 ± 0.028**	**0.939 ± 0.016**	**0.921 ± 0.019**
	VisionSAC	0.442 ± 0.262	0.402 ± 0.481	0.678 ± 0.353	0.843 ± 0.283

Table 6.3.: Final mean contact force magnitude (mean ± std, across 10 seeds) across all tasks, environments, and action space configurations. Best configurations are highlighted in bold.

Task	Method	Factory		Forge	
		4D	6D	4D	6D
PegInsert	KGRL	**0.787 ± 0.096**	**0.853 ± 0.072**	**3.517 ± 0.522**	**3.731 ± 0.966**
	SAC	2.345 ± 0.028	2.289 ± 0.307	4.573 ± 1.027	4.659 ± 1.297
	VisionKGRL	**0.648 ± 0.017**	**0.695 ± 0.137**	**3.272 ± 0.274**	**3.080 ± 0.468**
	VisionSAC	2.472 ± 0.046	1.887 ± 0.404	5.577 ± 0.292	6.095 ± 0.248
GearMesh	KGRL	**1.191 ± 0.047**	**1.193 ± 0.064**	**4.378 ± 0.156**	**4.486 ± 0.301**
	SAC	2.510 ± 0.053	1.921 ± 0.504	6.117 ± 0.152	3.916 ± 2.914
	VisionKGRL	**0.785 ± 0.047**	**0.784 ± 0.061**	**4.011 ± 0.144**	**3.775 ± 0.395**
	VisionSAC	2.531 ± 0.097	1.982 ± 0.215	6.858 ± 0.334	6.899 ± 2.247
NutThread	KGRL	**0.510 ± 0.022**	**0.506 ± 0.016**	**2.533 ± 0.047**	**2.363 ± 0.098**
	SAC	1.351 ± 0.801	1.811 ± 1.075	0.119 ± 0.059	0.275 ± 0.133
	VisionKGRL	**0.508 ± 0.019**	**0.508 ± 0.020**	**2.596 ± 0.082**	**2.538 ± 0.084**
	VisionSAC	1.942 ± 0.623	2.172 ± 1.194	5.700 ± 2.752	7.168 ± 2.603

Table 6.4.: Time to first success in seconds (mean ± std, across 10 seeds) across all tasks, environments, and action space configurations. Best configurations are highlighted in bold. "—" indicates no successful completion was recorded.

Task	Method	Factory		Forge	
		4D	6D	4D	6D
PegInsert	KGRL	4.174 ± 1.454	**4.200 ± 1.083**	3.114 ± 0.831	3.673 ± 0.773
	SAC	**3.606 ± 2.144**	5.972 ± 1.448	**2.615 ± 1.473**	**3.879 ± 2.480**
	VisionKGRL	3.071 ± 0.602	**3.222 ± 0.499**	3.230 ± 1.015	3.080 ± 0.633
	VisionSAC	**2.513 ± 1.495**	5.117 ± 1.122	**2.811 ± 2.359**	**2.388 ± 1.304**
GearMesh	KGRL	**4.440 ± 1.757**	4.470 ± 1.084	**2.742 ± 1.027**	**3.665 ± 0.654**
	SAC	5.106 ± 1.795	—	2.889 ± 1.647	2.970 ± 1.120
	VisionKGRL	**2.664 ± 0.819**	**3.349 ± 0.489**	2.768 ± 0.827	2.959 ± 0.632
	VisionSAC	3.863 ± 1.950	—	**2.185 ± 0.745**	**2.244 ± 0.865**
NutThread	KGRL	8.595 ± 0.155	**7.935 ± 0.338**	**8.876 ± 0.405**	**8.876 ± 0.405**
	SAC	**7.050 ± 0.707**	9.816 ± 1.999	—	—
	VisionKGRL	8.497 ± 0.203	7.827 ± 0.237	**8.558 ± 0.263**	**7.980 ± 0.293**
	VisionSAC	**7.136 ± 0.964**	**7.026 ± 0.654**	7.614 ± 1.213	8.416 ± 0.657

7. Discussion

This chapter evaluates the experimental results detailed in Chapter 6 and explores their implications for learning-based robotic assembly. The discussion is structured by task to enable a thorough analysis of the observed behaviors, succeeded by an analysis of the recurring patterns that arise across all experiments.

7.1. PegInsert Task

7.1.1. Factory Environment

In the 4D Factory PegInsert configuration, as shown in Figure 6.11 (a), both KGRL variants demonstrate a faster convergence and achieve higher final success rate compared to the corresponding SAC baselines. The average contact forces produced by KGRL policies are significantly lower than those generated by SAC policies. This suggests that KMP guidance, governed by the variance of the demonstrations, enhances learning efficiency and facilitates safer interactions with the environment during the insertion process.

The distinction becomes more evident in the 6D configuration, as shown in Figure 6.11 (b). When the action space is extended to include complete end-effector pose control, both variants of SAC are unable to learn effectively. The increased dimensionality of the exploration space hinders unconstrained reinforcement learning from developing effective insertion strategies within the given training budget. Conversely, both KGRL variants attain success rates similar to those of their 4D equivalents. This illustrates that the trajectory prior provided by KMP, shaped by the demonstration variance, effectively mitigates the increased complexity of the higher-dimensional action space by restricting exploration to viable motion patterns.

7.1.2. Forge Environment

The Forge environment incorporates force observations into the state representation, thereby modifying the learning dynamics. In the 4D Forge PegInsert task, as shown in Figure 6.12 (a), SAC with vision demonstrates improved learning performance compared to the Factory setting. Nonetheless, SAC without vision still achieves a low success rate, indicating that proprioceptive and force data alone are inadequate for reliable task execution.

An important observation in this context is that VisionSAC and VisionKGRL achieve similar final success rates. Nevertheless, VisionSAC achieves a higher cumulative reward. This difference can be explained by analyzing the time to first success, as detailed in Table 6.4. VisionSAC completes the task earlier than VisionKGRL, and because the success reward is aggregated throughout the remaining period of the episode rather than being a singular

signal, earlier task completion yields a greater total reward. The delayed completion time of VisionKGRL can be attributed to its dependence on the demonstration trajectory, which provides a particular approach velocity and motion profile. This limitation hinders the ability of the policty to succeed the task as fast as unrestricted exploration might allow, but it also leads to significantly reduced contact forces during the interaction. VisionKGRL attains an equivalent success rate to VisionSAC while applying considerably reduced force to the environment, indicating a safer and more regulated insertion behavior.

In the 6D Forge PegInsert configuration, shown in Figure 6.12 (b), SAC without vision attains a low success rate, whereas SAC with vision approaches the performance of VisionKGRL more closely. Nonetheless, both KGRL variants demonstrate faster convergence, achieve higher final success rates, and apply reduced contact forces compared to the SAC baselines.

7.2. GearMesh Task

7.2.1. Factory Environment

The 4D Factory GearMesh task shows a different pattern in contrast to PegInsert, as shown in Figure 6.14 (a). In this context, the convergence rate and final success rates of SAC and KGRL are quite similar. Both VisionSAC and VisionKGRL attain comparable final success rates; nevertheless, VisionKGRL achieves higher cumulative reward. This again can be clarified by the time to first success indicated in Table 6.4. VisionKGRL accomplishes the task faster in this case, which allows it to accumulate more success reward over the episode duration.

In the 6D Factory GearMesh configuration, shown in Figure 6.14 (b), the pattern observed in PegInsert reappears. Both SAC variants are unable to learn due to the increased complexity of the action space, whereas both KGRL variants achieve success rates similar to those in the 4D setting. This further supports the claim that KMP guidance, derived from the demonstration variance, is crucial for maintaining learning efficiency as the dimensionality of the action space increases.

7.2.2. Forge Environment

In the 4D Forge GearMesh task, as shown in Figure 6.15 (a), VisionSAC exhibits a more rapid convergence than VisionKGRL. Nonetheless, VisionSAC applies significantly greater contact forces during training, both in the initial exploration phase and after convergence. An analysis of the acquired behaviors provides an explanation for this difference.

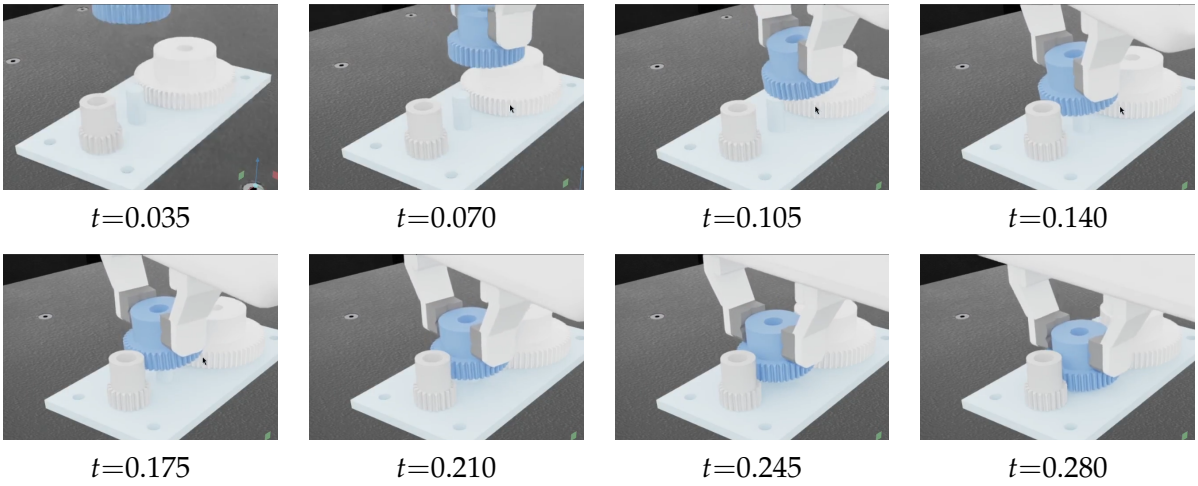


Figure 7.1.: Execution rollout of the VisionSAC policy on the 4D Forge GearMesh task, sampled at equal intervals $\Delta t = 0.035$ over the normalized episode duration $t \in [0.035, 0.280]$.

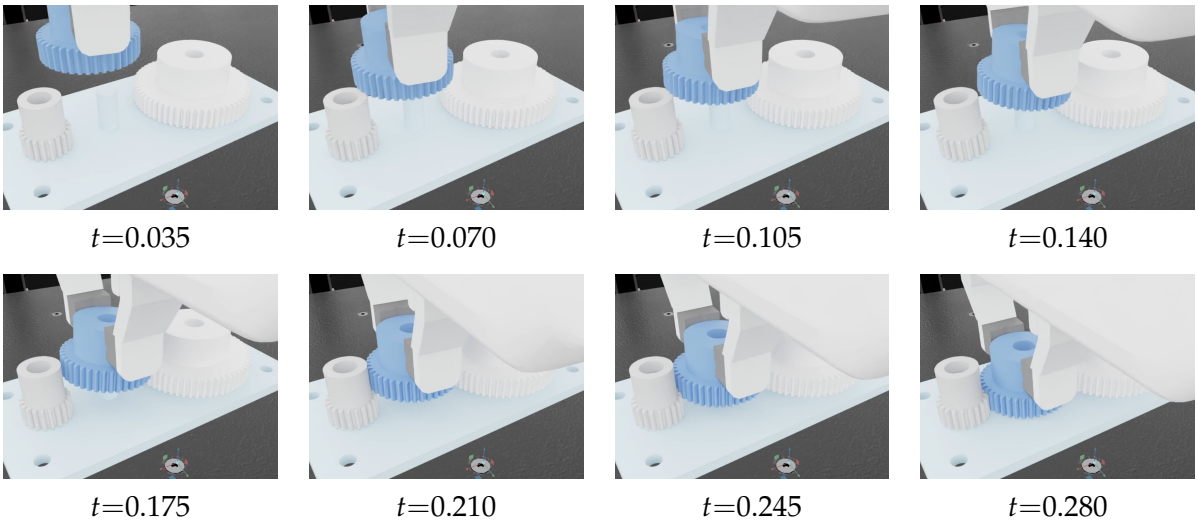


Figure 7.2.: Execution rollout of the VisionKGRL policy on the 4D Forge GearMesh task, sampled at equal intervals $\Delta t = 0.035$ over the normalized episode duration $t \in [0.035, 0.280]$.

As illustrated in Figure 7.1, the unconstrained VisionSAC policy discovers an aggressive strategy whereby the end-effector hits the first gear, then hits the second gear quickly, before positioning for insertion. This behavior might accelerate convergence but produces significant contact forces during each contact event. This exemplifies a key characteristic of unconstrained reinforcement learning: in the absence of motion priors, the policy may discover solutions that are efficient with respect to the reward signal but result in physically aggressive interactions

that would be undesirable or hazardous in practical applications. The trajectory prior in KGRL prevents such behaviors, as visible in Figure 7.2, by limiting the policy to operate within the vicinity of a demonstrated motion pattern that gradually and smoothly approaches the gear mesh configuration While final success rates of VisionSAC and VisionKGRL are similar, VisionSAC consistently achieves a higher cumulative reward. Similar to the Forge PegInsert case, Table 6.4 indicates that VisionSAC accomplishes the task more rapidly, resulting in a greater accumulated reward despite equivalent success rates.

In the 6D Forge GearMesh configuration, shown in Figure 6.15 (b), both KGRL variants achieve higher success rates and rewards compared to the SAC baselines. Moreover, the SAC policies demonstrate significant variation across various random seeds, suggesting that unconstrained learning in this high-dimensional context is extremely sensitive to initialization. The KGRL variants, in contrast, consistently demonstrate reduced variance across seeds, indicating enhanced robustness and reproducibility of the acquired behaviors. Low variance across seeds is a beneficial characteristic in practice, as it indicates that the training result is less reliant on random initialization and is more likely to yield a consistent policy in any specific training run.

7.3. NutThread Task

7.3.1. Factory Environment

The NutThread task represents the most challenging assembly scenario, necessitating sub-milimetric alignment and threading of a nut onto a bolt. In both the 4D and 6D Factory environments, as shown in Figure 6.17, both KGRL variants achieve substantially higher success rates and rewards compared to the SAC baselines. The SAC policies fail to converge within the designated training episodes, suggesting that unrestricted exploration is inadequate for discovering the specific threading behavior necessary for this task. The KGRL policies apply significantly reduced force, which is crucial in a threading task, as excessive force may result in cross-threading or mechanical damage.

7.3.2. Forge Environment

The Forge NutThread findings further highlight the complexity of this task. In both the 4D and 6D environments, as shown in Figure 6.18, SAC without visual input attains a success rate of zero, exhibiting no evidence of learning during the entire training period. The SAC with vision demonstrates some learning capability, yet converges markedly slower than the KGRL variants, displays greater variance across seeds, and applies significantly higher contact forces. The force levels generated by the SAC policies in this task are especially concerning from a safety standpoint, as they continously increase during training and might surpass acceptable limits in a physical assembly system.

A notable observation in both Factory and Forge NutThread experiments is that KGRL without vision and SAC without vision exhibit no indications of learning. This signifies that visual perception is essential for completing the NutThread task. The threading operation

necessitates sub-millimetric alignment precision, and even tiny misalignment between the nut and bolt results in task failure. Proprioceptive and force observations alone are inadequate for achieving this degree of alignment. Visual features facilitate the policy to execute precise adjustments required to solve this task, by encoding the relative spatial configuration of the nut and bolt.

7.4. Recurring Observations Across Tasks

7.4.1. Contact Force Reduction Through KMP Guidance

A constant observation across all tasks, environments, and action space dimensionalities is that KGRL policies generate lower mean contact forces than SAC policies. This pattern is consistent in Factory and Forge environments, across 4D and 6D action spaces, and applies to both vision and non-vision variants. The KMP prior restricts the policy to operate within the vicinity of the demonstrated motion, where the allowable deviation is governed by the demonstration variance, thereby inherently preventing the aggressive contact interactions that may occur from unrestricted exploration. This property is crucial for practical applications, as excessive contact forces may harm the robot, the workpieces, or the assembly fixtures.

7.4.2. Robustness and Variance Across Seeds

KGRL policies consistently demonstrate reduced variance across various random seeds in comparison to SAC policies. This phenomenon is evident across all tasks, particularly in the more challenging configurations such as Forge GearMesh 6D (Figure 6.15 (b)) and Forge NutThread (Figure 6.18). A reduced variance across seeds indicates that the training results are more predictable and less reliant on specific initializalizations. This implies that a reduced number of training runs are required to achieve an effective policy, thereby reducing the likelihood of unsuccessful training attempts. This robustness is due to the structured exploration facilitated by the trajectory prior, which decreases the sensitivity of the learning algorithm to the initial policy parameters.

7.4.3. Action Space Dimensionality

The transition from 4D to 6D action spaces significantly influences learning efficiency. In the Factory PegInsert and Factory GearMesh tasks, as shown in Figures 6.11 (b) and 6.14 (b), SAC entirely fails to learn in the 6D configuration, whereas KGRL preserves performance levels comparable to the 4D case. This illustrates that KMP guidance is especially beneficial as the dimensionality of the action space expands. The The KMP-based trajectory prior provides a structured reference motion, informed by the demonstration variance, that significantly alleviates the exploration challenge in higher-dimensional spaces. In the absence of such guidance, the reinforcement learning agent is required to simultaneously discover suitable behaviors across all six degrees of freedom, which poses a significant challange for the RL agent.

7.4.4. The Role of Visual Perception

The NutThread task provides the clearest evidence for the requirement of visual observations. In both Factory and Forge environments, all non-vision variants, whether SAC or KGRL, show no signs of learning within given training budget. This indicates that visual perception is not only beneficial but crucial for tasks necessitating sub-millimetric alignment precision. The visual encoder captures spatial relationships between manipulated objects that cannot be deduced solely from proprioceptive or force measurements.

Even in less challanging tasks such as PegInsert and GearMesh, visual observations still improve performance and robustness, and remain important for achieving high success rates.

The usage of a visual encoder with fixed weights in reinforcement learning also enhances training stability. The consistent visual representation during training enables the policy to comprehend a stable feature space. This prevents representation drift, which could otherwise introduce a non-stationarity state space into the learning process and hinder convergence.

7.4.5. Complementary Roles of Perception and KMP Guidance

The experimental findings suggest that visual perception and KMP guidance tackle different aspects of the learning problem and are most effective when they are combined. KMP guidance predominantly affects the action space by limiting exploration to viable motion patterns, thus enhancing learning stability and decreasing contact forces. Visual perception, on the other hand, enhances state representation by providing geometric and spatial information regarding the relative arrangement of the manipulated objects.

The integration of both components produces policies that are stable during training and adaptable during execution. KMP guidance guarantees that exploration is both safe and effective, while visual observations allow the policy to adapt to changes in the initial object pose and environmental conditions. The complementary relationship is most clearly illustrated in the NutThread task, where KMP guidance alone is inadequate due to the necessary alignment precision, and visual perception alone is insufficient because of the complexity of the threading motion.

7.4.6. Influence of Environment Physical Properties on Contact Forces

A significant observation concerns the absolute magnitude of contact forces in the two simulation environments. In the factory setting, despite the lack of force sensing in the observation space and the absence of a force penalty in the reward function, the average contact forces are significantly lower than those observed in the corresponding forge experiments. The difference is due to the different physical properties configured in Factory and Forge environments within IsaacLab. The Forge environment utilizes higher task-space proportional gains in its operational space controller, leading to a more rigid end-effector response, and this consequently results in higher contact forces during environmental interactions. This finding underscores that the physical parameters of the simulation environment, especially controller gains and contact dynamics, significantly affect the force profiles observed during

training and execution. The absolute force values reported in the experiments should be interpreted within the context of the respective environmental configurations rather than being directly compared between Factory and Forge. For practical implementation, careful adjustment of controller gains is crucial to guarantee that the acquired policies operate within permissible force thresholds.

7.5. Limitations

Despite the observed achievements, several limitations must be recognized. Firstly, the effectiveness of the KMP trajectory prior is dependent on the quality of the demonstrations utilized for its formulation. Unnecessary deviations or ineffective motion patterns in demonstrations may bias the learning process towards suboptimal behaviors.

Secondly, recovery from substantial deviations from the nominal trajectory can be challenging when employing KGRL. Since the trajectory progression in KGRL follows a fixed time progression, the reference motion continues to advance even if the robot temporarily deviates from the intended path. This may restrict the capability of the policy to naturally recover from significant disturbances during execution.

Thirdly, the current framework depends on visual encoders that are customized for the specific task. This may improve the quality of the learned representations for specific tasks, but it might limit the scalability of the method when handling a wider range of manipulation tasks. A more general visual representation, possibly obtained by using a more advanced Vision Transformer (ViT) model such as DINOv2, may enhance transferability across tasks.

7.6. Implications for Contact-Rich Assembly

The findings of this thesis demonstrate that the combination of visual perception and KMP-based trajectory priors represents an effective method for developing contact-rich assembly skills. Visual perception improves geometric awareness and resilience under varying initial conditions, whereas KMP guidance stabilizes exploration and facilitates safer interactions with the environment. The task-specific results indicate that the significance of each component is dependent on the requirements of the assembly task. For tasks with moderate alignment requirements, KMP guidance alone can significantly enhance performance. Visual perception is crucial for tasks necessitating high-precision alignment. The advantage of KMP guidance becomes especially evident as the dimensionality of the action space expands. In the 6D context, where the agent is required to manage all six degrees of freedom of the end-effector simultaneously, unconstrained reinforcement learning consistently fails to discover effective strategies within the training budget, whereas KGRL preserves performance levels comparable to the 4D case. This indicates that the KMP-based trajectory prior effectively accommodates the increased complexity within higher-dimensional action spaces, making it an essential element for scalable assembly learning. The integration of visual perception and KMP guidance provides the most robust and reliable performance across the evaluated tasks and

conditions, indicating a promising direction for the advancement of learning based robotic assembly systems.

8. Conclusion and Future Work

This thesis investigated the combined use of visual representation learning and probabilistic trajectory guidance for reinforcement learning in contact-rich robotic assembly tasks. The objective was to improve the stability, safety, and robustness of policy learning in manipulation contexts marked by high-dimensional control, complex contact dynamics, and small tolerance regions.

The Kernelized Guided Reinforcement Learning (KGRL) framework was enhanced to incorporate multimodal observations, including proprioception, force sensing, and compact visual latent representations obtained from task-specific Variational Autoencoders (VAEs). The resulting framework integrates trajectory priors derived from demonstrations utilizing Kernelized Movement Primitives (KMP) with reinforcement learning policies trained in the NVIDIA IsaacLab simulation environments. The primary contributions of this thesis, as presented in Chapter 1, are revisited below in the light of the experimental results.

The framework was validated through an extensive large-scale evaluation across Factory and Forge simulation environments on increasingly challenging assembly tasks: PegInsert, GearMesh, and NutThread. Standard SAC policies were evaluated against trajectory-guided KGRL policies and their vision-augmented counterparts in both 4D and 6D action space configurations. A systematic multimodal ablation study was conducted across four observational configurations: proprioceptive-only, force-augmented, vision-augmented, and combined force and vision settings, in order to isolate the individual and combined effects of each observation modality. The ablation study demonstrated that visual perception is essential for high-precision tasks such as NutThread, where all non-visual variants are incapable of learning.

Under all assessed conditions, trajectory-guided KGRL policies demonstrated decreased contact forces and reduced variance across random seeds in comparison to standard SAC baselines, thereby confirming enhanced safety and robustness. The decrease in contact forces signifies a smoother interaction with the environment and implies that trajectory guidance limits exploration to physically safer motion patterns.

The integration of visual representations demonstrated higher resilience against geometric uncertainty. Visual observations provide supplementary information about object orientation and alignment that cannot be acquired solely through proprioception or force sensing. The incorporation of visual latent features enables policies to more effectively manage varying initial conditions and object configurations, leading to a higher overall success rate.

The results obtained from the 6D action space setting are considered as the most significant contribution of this thesis. The 6D configuration, in which the agent controls all six degrees of freedom of the end-effector, is more general and more representative of real-world robotic manipulation. In this context, unconstrained SAC consistently fails to learn across various

tasks and environments, while KGRL attains success rates comparable to those achieved in the 4D scenario. This illustrates that the KMP-based trajectory prior effectively accommodates the increased exploration complexity that comes with higher-dimensional action space.

Overall, the experimental results demonstrate that the incorporation of structured trajectory priors with visual perception is an effective method for developing contact-rich manipulation skills, offering a systematic framework to improve the reliability of learning-based robotic assembly systems.

Notwithstanding these promising results, several limitations remain. The effectiveness of trajectory-guided reinforcement learning depends on the quality of the demonstrations employed to establish the trajectory prior. Furthermore, the current formulation relies on a predetermined trajectory progression along the normalized time variable, which might limit the recovery behavior in cases of substantial deviations from the reference trajectory. Additionally, the visual representation in this work utilizes task-specific Variational Autoencoders that are trained individually for each assembly scenario. While this provides compact and task-relevant latent observations, it might restrict the generalizability of the perception component.

Future research directions include multiple extensions of the proposed framework. A promising direction relates to the recovery capability of KGRL when the robot encounters undesirable states during operation. In the current formulation, KMP inference time increases monotonically, regardless of the actual state of the robot. If the robot significantly diverges from the intended trajectory, for example, as a result of an unintended collision that displaces or disorients the object being held, the prior trajectory continues to advance, and the policy attempts to execute the subsequent phases of the motion with an uncorrected pose. In contrast to unconstrained reinforcement learning, which can revisit previous configurations and reattempt strategies, KGRL is unable to regress along the trajectory. An appropriate extension would involve enhancing the KGRL policy by incorporating the selection of KMP inference time as an additional action dimension. This would enable the agent, upon identifying an undesirable condition, to choose an earlier inference time, thereby returning to an earlier phase of the nominal trajectory. The robot could subsequently adjust its position and orientation before continuing the assembly motion. This mechanism would integrate the safety and stability advantages of trajectory guidance with the ability of autonomous error recovery.

Another important future research direction relates to visual representation learning. This study employs frozen VAE encoders to obtain compact latent representations; however, recent advancements in vision transformers offer more expressive visual representations. Pretrained Vision Transformer models such as DINOv2 may provide semantically richer attributes that generalize across diverse tasks and domains. Integrating transformer-based visual representations into the KGRL framework may improve robustness to real-world perceptual noise and facilitate sim-to-real transfer.

Ultimately, expanding the framework to real world robotic systems remains an essential advancement towards practical implementation. The combination of trajectory-guided reinforcement learning with transformer-based visual representations and real-world perception

systems may facilitate the reliable acquisition of contact-rich assembly skills directly on physical robots.

These directions highlight the potential of integrating structured priors, modern visual representation learning, and reinforcement learning to improve learning based robotic manipulation in complex assembly scenarios.

A. Appendix

This appendix contains additional material that supports the main thesis but does not fit directly into the body of the document. Specifically, it provides further analysis of the performance degradation observed when normalized time is excluded from the KGRL observation and presents the corresponding training results.

A.1. Time Ablation for KGRL

An ablation study on the Isaac Forge PegInsert task utilizing the 4D action space was conducted to examine the impact of normalized time on the KGRL policy input. Four variants were compared: KGRL with normalized time, KGRL without normalized time, vision-augmented KGRL with normalized time, and vision-augmented KGRL without normalized time.

Figure A.1 demonstrates a clear and consistent decline in performance when normalized time is omitted from the policy observation. This effect exists in both standard KGRL and the vision-augmented KGRL variants. When normalized time is included, the policy achieves higher rewards, converges faster, and reaches significantly higher success rates. Conversely, the variants without time demonstrate inadequate convergence and significantly underperform compared to their counterparts.

This behavior is not coincidental, but arises directly from the structure of KGRL. The policy operates on top of a trajectory prior generated by KMP, with the prediction explicitly dependent on the normalized time s. The KMP mean and covariance are functions of s, and therefore the reference trajectory and its associated uncertainty progress over time. In particular, the covariance $\boldsymbol{\Sigma}(s)$ regulates the strength of the null space correction, which governs how strongly the policy can deviate from the reference trajectory.

Consequently, normalized time is a crucial input for KGRL. In the absence of s, the policy is unable to determine the current progression along the trajectory, therefore it cannot accurately interpret the KMP output. A single local observation may correspond to various phases of motion, each linked to different nominal mean trajectories and covariance structures. This ambiguity makes it fundamentally difficult for the policy to learn a consistent mapping from observations to null space actions.

Overall, this ablation demonstrates that normalized time is not a supplementary feature but a fundamental component of the KGRL formulation. The KMP inference and the corresponding null space correction are explicitly dependent on s; thus, its removal breaks the underlying structure of the method. Consequently, all experiments in this thesis include normalized time s as a component of the observation, including the SAC baselines.

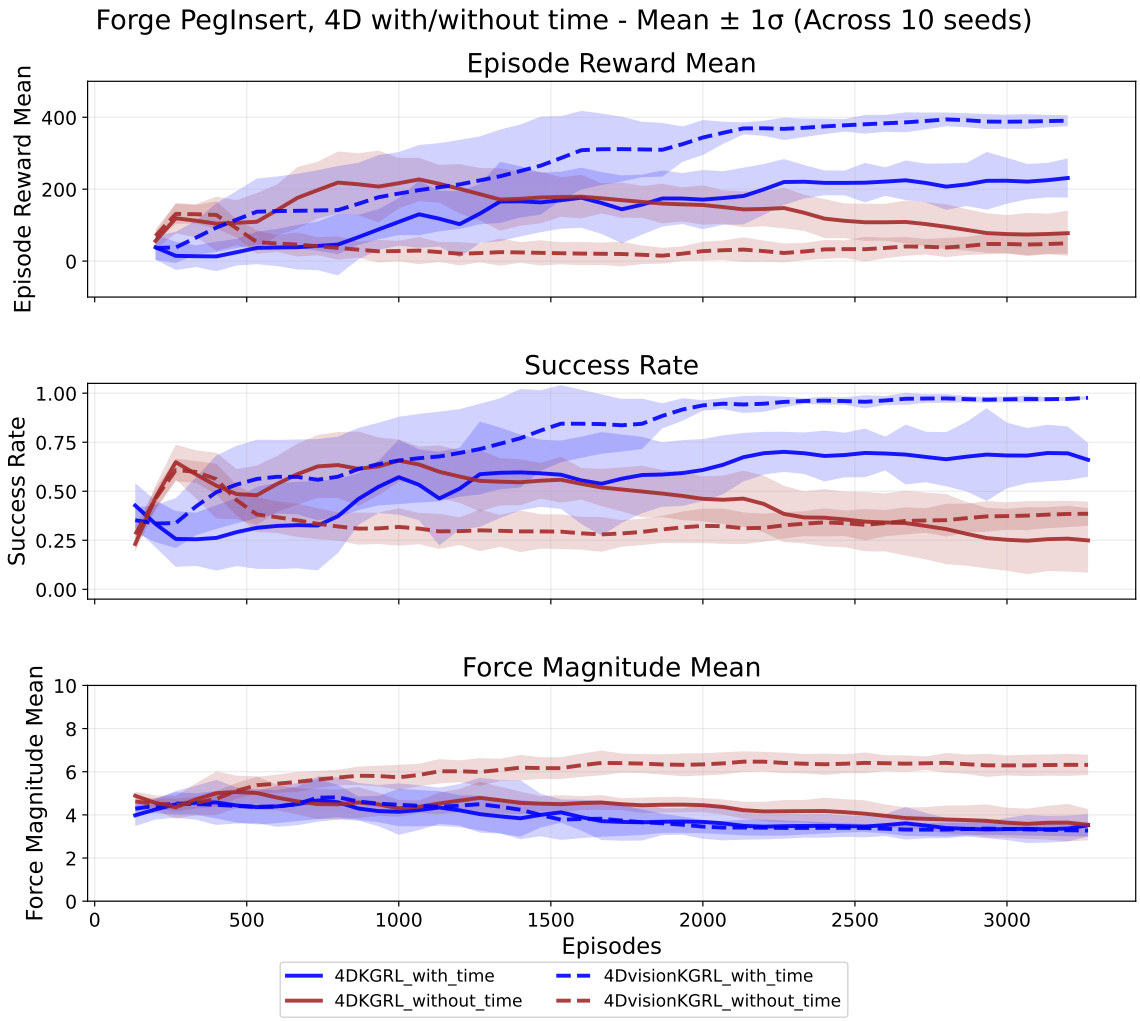


Figure A.1.: Training curves for the time ablation study on Isaac Forge PegInsert with a 4D action space. The figure compares KGRL and vision augmented KGRL, each with and without normalized time in the observation.

List of Figures

List of Tables

Bibliography

[1] N. Hogan. "Impedance Control: An Approach to Manipulation: Part III-Applications". In: *Journal of Dynamic Systems, Measurement, and Control* 107.1 (Mar. 1985), pp. 17–24. ISSN: 0022-0434. DOI: 10.1115/1.3140701. eprint: https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/107/1/17/5492321/17_1.pdf. URL: https://doi.org/10.1115/1.3140701.

[2] D. Whitney. "Historical perspective and state of the art in robot force control". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 262–268. DOI: 10.1109/ROBOT.1985.1087266.

[3] Y. Gai, J. Guo, D. Wu, and K. Chen. "Feature-Based Compliance Control for Precise Peg-in-Hole Assembly". In: *IEEE Transactions on Industrial Electronics* 69.9 (2022), pp. 9309–9319. DOI: 10.1109/TIE.2021.3112990.

[4] J. Kober, J. A. Bagnell, and J. Peters. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274. DOI: 10.1177/0278364913495721.

[5] J. Ding, C. Wang, and C. Lu. "Transferable Force-Torque Dynamics Model for Peg-in-hole Task". In: *CoRR* abs/1912.00260 (2019). arXiv: 1912.00260. URL: http://arxiv.org/abs/1912.00260.

[6] J. Xu, Z. Hou, W. Wang, B. Xu, K. Zhang, and K. Chen. "Feedback Deep Deterministic Policy Gradient With Fuzzy Reward for Robotic Multiple Peg-in-Hole Assembly Tasks". In: *IEEE Transactions on Industrial Informatics* 15.3 (2019), pp. 1658–1667. DOI: 10.1109/TII.2018.2868859.

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1861–1870. URL: https://proceedings.mlr.press/v80/haarnoja18b.html.

[8] L. Dodeja, K. Schmeckpeper, S. Vats, T. Weng, M. Jia, G. Konidaris, and S. Tellex. "Accelerating Residual Reinforcement Learning With Uncertainty Estimation". In: *IEEE Robotics and Automation Letters* 11.1 (2026), pp. 970–977. DOI: 10.1109/LRA.2025.3636808.

[9] A. Padalkar, F. Stulp, G. Neumann, and J. Silvério. "Towards Safe and Efficient Learning in the Wild: Guiding RL With Constrained Uncertainty-Aware Movement Primitives". In: *IEEE Robotics and Automation Letters* 10.7 (2025), pp. 6880–6887. DOI: 10.1109/LRA.2025.3566599.

[10] J. Silvério and Y. Huang. "A Non-parametric Skill Representation with Soft Null Space Projectors for Fast Generalization". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 2988–2994. DOI: 10.1109/ICRA48891.2023.10161065.

[11] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell. "Kernelized movement primitives". In: *The International Journal of Robotics Research* 38.7 (2019), pp. 833–852. DOI: 10.1177/0278364919846363. eprint: https://doi.org/10.1177/0278364919846363. URL: https://doi.org/10.1177/0278364919846363.

[12] Y. Guo, I. Akinola, L. Johannsmeier, H. Hadfield, A. Gupta, and Y. Narang. *SPARR: Simulation-based Policies with Asymmetric Real-world Residuals for Assembly*. 2026. arXiv: 2602.23253 [cs.RO]. URL: https://arxiv.org/abs/2602.23253.

[13] NVIDIA, : M. Mittal, et al. *Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning*. 2025. arXiv: 2511.04831 [cs.RO]. URL: https://arxiv.org/abs/2511.04831.

[14] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, A. Handa, and D. Fox. "Factory: Fast Contact for Robotic Assembly". In: *Proceedings of Robotics: Science and Systems*. New York City, NY, USA, June 2022. DOI: 10.15607/RSS.2022.XVIII.035.

[15] M. Noseworthy, B. Tang, B. Wen, A. Handa, C. Kessens, N. Roy, D. Fox, F. Ramos, Y. Narang, and I. Akinola. "FORGE: Force-Guided Exploration for Robust Contact-Rich Manipulation Under Uncertainty". In: *IEEE Robotics and Automation Letters* 10.5 (2025), pp. 4436–4443. DOI: 10.1109/LRA.2025.3551637.

[16] C. Chen, H. Wang, Y. Pan, and D. Li. "Vision-based robotic peg-in-hole research: integrating object recognition, positioning, and reinforcement learning". In: *The International Journal of Advanced Manufacturing Technology* 135 (Oct. 2024), pp. 1119–1129. DOI: 10.1007/s00170-024-14482-y.

[17] N. Pitchandi, S. Perumaal, and M. Irulappan. "Insertion Force Analysis of Compliantly Supported Peg-In-Hole Assembly". In: *Assembly Automation* 37 (July 2017), pp. 00–00. DOI: 10.1108/AA-12-2016-167.

[18] Z. Hou, H. Dong, K. Zhang, Q. Gao, K. Chen, and J. Xu. "Knowledge-Driven Deep Deterministic Policy Gradient for Robotic Multiple Peg-in-Hole Assembly Tasks". In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2018, pp. 256–261. DOI: 10.1109/ROBIO.2018.8665255.

[19] M. Nigro, M. Sileo, F. Pierri, K. Genovese, D. D. Bloisi, and F. Caccavale. "Peg-in-Hole Using 3D Workpiece Reconstruction and CNN-based Hole Detection". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 4235–4240. DOI: 10.1109/IROS45743.2020.9341068.

[20] Y. Zhao, F. Gao, Y. Zhao, and Z. Chen. "Peg-in-Hole Assembly Based on Six-Legged Robots with Visual Detecting and Force Sensing". In: *Sensors* 20.10 (2020). ISSN: 1424-8220. DOI: 10.3390/s20102861. URL: https://www.mdpi.com/1424-8220/20/10/2861.

[21] Z. Zhang, Y. Wang, Z. Zhang, L. Wang, H. Huang, and Q. Cao. "A residual reinforcement learning method for robotic assembly using visual and force information". In: *Journal of Manufacturing Systems* 72 (2024), pp. 245–262. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2023.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0278612523002352.

[22] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. "A survey of robot learning from demonstration". In: *Robotics and Autonomous Systems* 57.5 (2009), pp. 469–483. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2008.10.024. URL: https://www.sciencedirect.com/science/article/pii/S0921889008001772.

[23] S. A. Mehta, Y. U. Ciftci, B. Ramachandran, S. Bansal, and D. P. Losey. "Stable-BC: Controlling Covariate Shift With Stable Behavior Cloning". In: *IEEE Robotics and Automation Letters* 10.2 (2025), pp. 1952–1959. DOI: 10.1109/LRA.2025.3526439.

[24] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors". In: *Neural Computation* 25.2 (2013), pp. 328–373. DOI: 10.1162/NECO_a_00393.

[25] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. "Probabilistic Movement Primitives". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 2616–2624.

[26] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine. "SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 16961–16969. DOI: 10.1109/ICRA57147.2024.10610040.

[27] J. Luo, C. Xu, J. Wu, and S. Levine. "Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning". In: *Science Robotics* 10.105 (2025), eads5033. DOI: 10.1126/scirobotics.ads5033. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.ads5033. URL: https://www.science.org/doi/abs/10.1126/scirobotics.ads5033.

[28] J. Stranghöner, P. Hartmann, M. Braun, S. Wrede, and K. Neumann. "SHaRe-RL: Structured, Interactive Reinforcement Learning for Contact-Rich Industrial Assembly Tasks". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2026. arXiv: 2509.13949 [cs.RO]. URL: https://arxiv.org/abs/2509.13949.

[29] T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy. "Residual Learning From Demonstration: Adapting DMPs for Contact-Rich Manipulation". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4488–4495. DOI: 10.1109/LRA.2022.3150024.

[30] K.-H. Ahn, M. Na, and J.-B. Song. "Robotic assembly strategy via reinforcement learning based on force and visual information". In: *Robotics and Autonomous Systems* 164 (2023), p. 104399. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2023.104399. URL: https://www.sciencedirect.com/science/article/pii/S0921889023000386.

[31] P. Jin, Y. Lin, Y. Song, T. Li, and W. Yang. "Vision-force-fused curriculum learning for robotic contact-rich assembly tasks". In: *Frontiers in Neurorobotics* 17 (2023). ISSN: 1662-5218. DOI: 10.3389/fnbot.2023.1280773. URL: https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2023.1280773.

[32] J. Tang, X. Yuan, and S. Li. "Visual-Tactile Fusion and SAC-Based Learning for Robot Peg-in-Hole Assembly in Uncertain Environments". In: *Machines* 13.7 (2025). ISSN: 2075-1702. DOI: 10.3390/machines13070605. URL: https://www.mdpi.com/2075-1702/13/7/605.

[33] J. Luo, O. Sushkov, R. Pevceviciute, W. Lian, C. Su, M. Vecerik, N. Ye, S. Schaal, and J. Scholz. "Robust Multi-Modal Policies for Industrial Assembly via Reinforcement Learning and Demonstrations: A Large-Scale Study". In: *Proceedings of Robotics: Science and Systems*. Virtual, July 2021. DOI: 10.15607/RSS.2021.XVII.088.

[34] B. Tang, M. Lin, I. Akinola, A. Handa, G. Sukhatme, F. Ramos, D. Fox, and Y. Narang. "IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality". In: July 2023. DOI: 10.15607/RSS.2023.XIX.039.

[35] C. M. Bishop. *Pattern Recognition and Machine Learning*. 1st ed. Information Science and Statistics. New York, NY: Springer, 2006. ISBN: 978-0-387-31073-2.

[36] S. Calinon. "A tutorial on task-parameterized movement learning and retrieval". In: *Intelligent Service Robotics* 9.1 (2016), pp. 1–29. ISSN: 1861-2784. DOI: 10.1007/s11370-015-0187-9. URL: https://doi.org/10.1007/s11370-015-0187-9.

[37] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005. ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001. eprint: https://direct.mit.edu/book-pdf/2514321/book_9780262256834.pdf. URL: https://doi.org/10.7551/mitpress/3206.001.0001.

[38] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018. ISBN: 978-0-262-03924-6.

[39] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2014. URL: http://arxiv.org/abs/1312.6114.