Bridging the Sim-to-Real Gap for Surgical Robotic Manipulation Tasks Using Diffusion Models

by

Ethan Rajah

Supervisor: Lueder Kahrs
April 2025

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

Bridging the gap between simulated and real-world environments remains a central challenge in developing robust vision-based reinforcement learning (RL) policies for surgical robotics. This study presents a novel RL training pipeline that integrates fine-tuned diffusion models to generate realistic image observations from Unity-based simulations. By leveraging DreamBooth to fine-tune Stable Diffusion models on real surgical task data, style transfer is achieved, allowing for the generation of high-fidelity images that closely resemble the real-world task environment. The diffusion-enhanced images are used during policy training, demonstrating that RL agents can learn meaningful action distributions despite the stochasticity of generated observations. Experimental results show that diffusion-based policies follow learning trends comparable to baseline PPO policies without diffusion inference, although training speed is limited by the slow diffusion denoising process. The variability introduced by diffusion-generated textures aids policy robustness by exposing agents to diverse visual inputs that mimic real-world inconsistencies. While sim-to-real transfer was not conducted due to extended training times, the modular pipeline developed here provides a strong foundation for future experimentation in both simulated and real environments. These findings highlight the potential of diffusion models for enhancing domain randomization and improving policy generalization in surgical robotics tasks.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Lueder Kahrs for providing me with the opportunity to work at his lab for my undergraduate thesis. His guidance and support throughout this project have been invaluable, and I am grateful for the knowledge and experience I have gained throughout my time at the MEDCVR lab.

I would also like to thank my mentor, Amey Pore, for his continuous support and encouragement during my time at the lab. His expertise and insights have greatly contributed to my understanding of the field, and I am thankful for the time he dedicated to helping me with my project. I would also like to extend my appreciation to all the members of the MEDCVR lab for their support and collaboration throughout this project. Their expertise and advice have been instrumental in shaping my research and enhancing my learning experience. I am grateful for the collaborative environment that the lab fosters, which has allowed me to grow as a researcher.

Finally, I would like to thank my family and friends for their unwavering support and encouragement throughout my academic journey. Their belief in me has always motivated me to strive for excellence and never give up on my goals. I am truly grateful for their love and support.

Contents

List of Tables

List of Figures

List of Acronyms

Acronym	Definition
ALDM	Adversarial Supervision Layout-to-Image Diffusion Model
CLIP	Contrastive Language-Image Pre-training
CUT	Contrastive Unpaired Translation
CycleGAN	Cycle Generative Adversarial Network
DCL	Deep Contrastive Learning
dVRK	da Vinci Research Kit
dVSS	da Vinci Surgical System
FID	Fréchet Inception Distance
GAN	Generative Adversarial Network
KID	Kernel Inception Distance
LC	Laparoscopic Cholecystectomy
LDM	Latent Diffusion Model
LoRA	Low-Rank Adaptation
MDP	Markov Decision Process
ODE	Ordinary Differential Equation
PPO	Proximal Policy Optimization
PSM	Patient-Side Manipulator
RL	Reinforcement Learning
ROS	Robot Operating System
SB3	Stable Baselines 3
SAC	Soft Actor-Critic
VAE	Variational Autoencoder
ViT	Vision Transformer

Chapter 1

Introduction

1.1 Motivation

The increasing autonomy in robot-assisted surgery is an emerging field of research aimed at reducing surgeon workload and enhancing overall procedural outcomes [1]. This area of research addresses challenges in maintaining precise manipulation control of surgical tools within constrained environments, while only having limited resolution images for visual feedback and navigation. Reinforcement Learning (RL) has been the most promising approach to solving these challenges within surgical environments, particularly in simulation where many image samples can be collected at a low cost [2]. Sim-to-real strategies have been developed as a framework to transfer learned simulation policies to real robot control, using techniques such as domain randomization and adaptation to generate more realistic simulations for image-based training [3–5]. These techniques are crucial for aiding efforts in bridging the sim-to-real gap, which refers to the performance disparity between RL policies trained in simulation and their performance in real-world environments. This gap is often influenced by differences in visual distributions between simulation and the real world, such as variations in lighting, occlusions, and noise, which can challenge the generalization of policies trained in simulation when applied to real-world scenarios. Recent investigations in bridging the sim-to-real gap have utilized synthetic image architectures such as Cycle Generative Adversarial Networks (CycleGANs) to generate realistic renderings of simulation images to use as input for image-based RL agents [6]. However, GANs require large amounts of training data, are often required to be retrained from scratch to suit new tasks, tend to experience mode collapse, and struggle with noisy pixel inputs. Unpaired image-to-image translation alternatives such as Contrastive Unsupervised Translation (CUT) and Deep Contrastive Learning (DCL) have been applied to the sim-to-real surgical scene to consolidate the training and generalization issues observed with CycleGANs, but the synthetic image

results highlight a loss in image information necessary to compensate for the dynamics of the physical robot [7]. Diffusion models have become an emergent method of solving the limitations of CycleGAN architectures within the generative machine learning research field because of its ability to achieve state-of-the-art synthesis results on image data through a sequential application of denoising autoencoders [8, 9]. However, they have yet to be applied to RL policy training for surgical tasks, despite their potential for generating generalizable, high-fidelity images required to bridge the sim-to-real gap.

This novel study aims to investigate the sim-to-real performance of RL agent policies trained on diffusion-generated images over various surgical tasks, such as tissue manipulation and thread insertion. The findings from this study provide insights into future research directions for enhancing the sim-to-real transfer of vision-based RL policies in surgical robotics, where environments and imaging capabilities are complex and dynamic. With the aim of better understanding diffusion image learning for surgical tasks, this work introduces a framework for fine-tuning and deploying diffusion models into common RL training algorithms, providing future opportunities for focused experimentation and analysis on the effects of various surgical data and task styles on policy performance. Due to the significant amount of time required for researchers to tune their experimental setup to match simulation for successful sim-to-real transfer, investigating potential improvements to policy generalization and transferability through diffusion can reduce the need for extensive domain randomization techniques during training, and therefore accelerate the development of autonomous surgical systems.

1.2 Objectives

The objective of this study is to establish a foundation for selecting an appropriate approach to domain randomization during policy training for complex surgical tasks, offering insights into how effectively policies can transfer high-fidelity image learning to real-world surgical scenarios. Building on previous sim-to-real research from the Medical Computer Vision and Robotics (MEDCVR) lab, this work aims to evaluate the effectiveness of diffusion models in bridging the policy reality gap, in comparison to existing methods in literature. The main objectives of this study are as follows:

- **Define a fine-tuning methodology for high fidelity diffusion inference**, enabling the use of collected real-world image data of various surgical tasks to learn realistic representations specific to surgical scenes.
- **Develop an RL training pipeline that incorporates fine-tuned diffusion models** for generating realistic images from simulation input observations, providing a

method for learning generalizable policy action distributions from high-fidelity images.

- **Transfer trained diffusion-based policies to a real dVRK robot**, and conduct experiments using the Sim2Real control pipeline described in Section 2.1 to refine and evaluate the policy performance.
- **Compare the proposed diffusion-based method to traditional domain randomization techniques** in vision-based RL policy training to assess the capabilities of diffusion in closing the sim-to-real gap.

Overall, these objectives aim to provide insights into how diffusion models can be used to bridge the performance gap between simulated and real-world surgical environments, with a specific focus on analyzing improvements and limitations when using diffusion inference for vision-based RL policy training.

Chapter 2

Related Work

Due to the complexity of surgical tasks, automation research for surgical robotics has focused on solving various sub-tasks such as tissue manipulation, cutting and suturing using vision-based approaches. Researchers aim to use the limitations and challenges discovered when solving simpler surgical sub-tasks to build a framework within the field for solving more complex tasks using RL.

2.1 Sim-to-Real Transfer & Domain Randomization

Many surgical robotics researchers use the da Vinci Research Kit (dVRK), a robotics research platform based on the da Vinci Surgical System (dVSS), for solving the aforementioned sub-tasks because of its frequent use as a surgical robot for minimally invasive procedures [4, 10]. Thananjeyan et al. [11] and Haiderbhai et al. [4] use vision-based RL to learn image-based manipulation of the dVRK, with the former relying on manually collected and annotated real-world images for training, rather than training in simulation and transferring to the real robot. This approach is limited by the work required to collect real-world data, which is often expensive and time-consuming, as well as by the reduced flexibility of agent exploration for optimal policy training compared to learning in simulated environments. The latter approach uses the Unity3D game engine as a simulator due to its sophisticated underlying physics engine, Nvidia PhysX, and the Unity ML-Agents package, which facilitates the training of RL agents between Unity and PyTorch. Furthermore, this work simulates the kinematics of the dVSS Patient Side Manipulator (PSM) so that it can be controlled by the agent through inverse kinematics. Using a Proximal Policy Optimization (PPO) [12] agent from ML-Agents, the PSM is trained to push a cube from a starting position to a desired goal position, effectively demonstrating the simplified motions associated with tissue manipulation. The work builds a Sim2Real pipeline to transfer the learned policy

from ML-Agents to the real dVRK using the Robotics Operating System (ROS) and a series of Cartesian position increments and joint state conversions based on the input image passed into the model. This effectively removes the reliance on the Unity simulation when executing the policy on the real robot. Moreover, experiments with the trained policy and the Sim2Real pipeline demonstrate the significant improvements in task success when using domain randomization techniques such as randomizing lighting, textures, camera angles, and cube physics during simulation training.

Using the same framework, Gondokaryono et al. [13] use vision-based RL to learn how to roll a miniature block with a surgical robot tool tip. The results again demonstrate the challenges of sim-to-real transfer and the need for more sophisticated domain randomization techniques during policy training to bridge the sim-to-real gap. Further research has explored training simulation policies for more complex surgical tasks, such as cutting. However, these studies highlight a significant limitation: it is not easy to identify sim-to-real gap related failures of trained policies when executing on a real robot. To address this, broader domain randomization ranges within simulations are often required as a form of compensation, although being a costly method, especially if the ranges are large or the task becomes more complex [3].

2.2 Bridging the Sim-to-Real Gap using GANs

Aside from the aforementioned domain randomization techniques that can be applied when training in simulation, there has been significant research in the use of generative models for generating realistic images for RL policy training, particularly using GANs [14]. This process is described in literature as pixel-level domain adaptation (or style transfer), where models are conditioned on the pixels of a source image and then re-styled to generate a representation that closely resembles the style of the target domain [6, 15]. GANs consist of two neural networks, a generator and a discriminator, where the generator learns to generate images that are indistinguishable from the target domain, and the discriminator learns to distinguish between real and generated images. This architecture provides a feedback loop that allows the generator to improve its image generation capabilities over time [16]. Rao et al. [6] use CycleGANs to map images from simulation to a realistic domain and jointly train an RL agent to complete a grasping task. The CycleGAN architecture consists of two GANs, one to learn and perform the domain adaptation from the source to the target and the other to do the opposite translation. The RL agent is trained jointly with the sim-to-real CycleGAN so that additional image consistency losses introduced in their work can be used to learn task geometry semantics that must be preserved during the image translation process. However, these approaches only address the visual domain gap between

simulation and reality, not any physics based differences, thus emphasizing the continued need for domain randomization when training in simulation, albeit ideally to a lesser extent than what is traditionally necessary for consistent task success. More importantly, this work is constrained by the deterministic nature of the GAN architecture, making it unsuitable for surgical vision-based RL tasks, as imaging systems like endoscopes are inherently noisy and have limited resolution.

Contrastive GAN architectures, such as CUT and DCL, have proven effective in narrowing the sim-to-real gap for vision-based RL policies in surgical deformable object manipulation tasks. Scheikl et al. [7] find that CUT and DCL perform better for unpaired image-to-image translation than CycleGAN and require less data for training. CUT and DCL maximize mutual information between the source and target image through the use of a contrastive loss approach. This aids with stabilizing training and ensures that these methods can retain key visual features during the image translation process, which is crucial for maintaining the task semantics of the surgical scene. However, the experimental results demonstrate that the contrastive GAN approaches struggle with preserving key image features required for effective policy training, thus resulting in poor agent performance when transferred to the real robot.

2.3 Diffusion Models for High-Fidelity Image Generation

GAN architectures are generally known to struggle with capturing the full data distribution of the images it is trained on and are often difficult to optimize due to instability. While recent studies have worked on advancing the GAN architecture, literature has shown that the architecture still struggles with mode collapse, where the generator produces a limited set of images repeatedly, thus reducing the generalization capabilities of the model [16]. Diffusion models have become popular in the generative machine learning research field due to their ability to consolidate the training and generalization issues observed with GANs [17]. The diffusion model architecture consists of a series of denoising autoencoders that are applied sequentially to the input image, which is first augmented with Gaussian noise. The sequential denoising process forms a Markov chain, where each step in the denoising chain is conditioned on the previous step. This architecture is probabilistic as it is designed to learn a data distribution by gradually denoising a normally distributed variable, essentially learning the reverse process of adding noise to the data [8, 16].

2.3.1 LDMs & Applications of Diffusion Models

Among the many diffusion models deployed by researchers, the Stable Diffusion model from Stability AI has become popular for generating state-of-the-art synthesis results on image

data because of its relatively lightweight architecture, allowing it to be run on consumer GPUs, and its flexibility in conditioning the denoising process on various modalities through the use of cross-attention [18]. Stable Diffusion uses a latent diffusion model (LDM) [8] comprising of a variational autoencoder (VAE) to convert images into a lower-dimensional latent space and apply Gaussian noise, a U-Net decoder to denoise the latent vectors, and another VAE to convert the denoised latent vectors back into images. The Stable Diffusion architecture also makes use of a pre-trained Contrastive Language-Image Pre-training (CLIP) Vision Transformer (ViT) text encoder which maps text and images to a common latent space, allowing the model to generate images from text prompts [19, 20].

Robotics research has been conducted on the application of diffusion models for bridging the sim-to-real gap, but not in the context of vision-based RL policy training. For instance, Li et al. [5] introduce an Adversarial Supervision Layout-to-Image Diffusion Model (ALDM) to generate realistic-style images from a Gazebo simulation environment for use as training data for a YOLOv8 object detection model. This work employs a discriminator to the diffusion model to provide pixel-level feedback on the similarity between the generated image and the input segmentation layout. The paper highlights the demand for precise pose consistency of objects between the real and generated images, particularly for robot grasping tasks. Although demonstrating high fidelity image generation and object detection results, the training process for the diffusion model involves using two large datasets, ADE20K and Cityscapes, which together compose of about 50,000 images. Datasets of this size are not feasible for the surgical domain because of the limited availability of real-world surgical images, particularly due to the high cost of collecting and annotating the data [18].

Diffusion models have become the standard for generating high-fidelity images, especially for artistic and creative applications [9], and have resulted in many optimizations and control enhancements for general public use [21]. Chung et al. [15] introduce a training free style transfer method for diffusion models that allows users to apply a style transfer to an image without the need for retraining the model. This work manipulates the latent features of the transformer self-attention layers during the reverse diffusion process to inject style information into the image generation process. Specifically, the method substitutes the self-attention key and value vectors of the source image with those of the style image, therefore acting as a cross attention mechanism that is able to maintain source contents while modifying its style to match the target image. While the training free approach saves time and computational resources, the experimental results show that this approach would not be appropriate for vision-based RL policy training, as it lacks the ability to control the style transfer for various objects within the surgical scene, such as tissue and surgical tools.

2.3.2 Fine-Tuning & Conditioning Diffusion Models

To make conditional control over diffusion model outputs more accessible to general users, Zhang et al. [22] introduce ControlNet, a neural network architecture that adds spatial conditioning control to the model. This is done by locking the parameters of the pre-trained diffusion model and creating a trainable copy of the encoder layers with an external conditioning vector as input. The reused parameters of the trainable copy provide robustness for handling diverse input conditions and is connected to the locked model with zero convolution layers. This work introduces the 1×1 convolutional layer as a zero convolution because the weight and bias parameters are initialized to zero, preventing random noise from influencing the hidden states of the neural network during training. This method significantly reduces the need for manual prompt and image refinement when generating images using a diffusion model. Currently, there are many ControlNet variations to support various types of control inputs (i.e., edge maps, human pose skeletons, segmentation masks, and depth maps). This enables users to generate high fidelity images with specific pose constraints, shapes, boundaries, and styles without the need for text prompts. More importantly, by restricting the number or rank of trainable parameters in the large pre-trained model, ControlNet can alleviate the effects of overfitting and catastrophic forgetting, which are common when fine-tuning with limited data [22, 23].

Kaleta et al. [18] make use of ControlNet and Stable Diffusion to generate realistic images from simulation for laparoscopic cholecystectomy (LC), a minimally invasive surgical procedure for gallblader removal. This work serves as a basis for the investigation done in this study as it demonstrates the potential of diffusion models for generating realistic surgical data from simulation for vision-based RL policy training. Furthermore, this work addresses the issue of limited data availability in clinical environments for fine-tuning diffusion models by using Dreambooth, a diffusion-based fine tuning method for subject-driven image generation. Dreambooth allows researchers to fine-tune diffusion models on small datasets (typically ranging from 5 to 200 images per style), enabling the model to associate a unique text identifier with the specific subject or features represented in the dataset [24]. Kaleta et al. use Dreambooth to fine-tune the text encoder and U-Net weights of a Stable Diffusion model on an LC dataset of 100 images, therefore providing a unique identifier to be used for generating realistic LC-style images from simulation images during inference. ControlNet's SoftEdge network is used to generate consistent shapes and boundaries of objects when translating the source image to the target style, while the Tile network is used to introduce fine details to background tissue while preserving accurate color information. This work demonstrates state-of-the-art sim-to-real image translation results, but is limited in its application to vision-based RL due to the lack of temporal coherence in the simulation data [18].

2.4 Insights from Prior Studies

The novel approach presented in this study builds on the limitations and challenges identified in previous research, aiming to develop more robust vision-based RL policies for surgical applications. Real-world surgical images are often scarce and suffer from noise due to the constraints of endoscopic tools. However, diffusion-based architectures and training frameworks such as Dreambooth offer a compelling alternative, generating high-fidelity surgical images that outperform traditional GAN architectures. The stochastic nature of diffusion enables these models to capture the full data distribution of a given style through sequential denoising, making them well-suited for surgical image generation. Additionally, Dreambooth facilitates fine-tuning with small datasets, further enhancing its applicability to surgical robotics research. Diffusion models ensure consistency between simulation and synthetic images through ControlNet, a crucial requirement for vision-based RL policy training, where PSM actions in simulation must be preserved in realistic renderings.
Compared to architectures like CycleGAN, diffusion-based methods address key challenges in robot physics consistency and image generalization, making them a promising candidate for advancing vision-based RL policy training in surgical robotics and bridging the sim-to-real gap.

Chapter 3

Methodology

This study makes use of the Sim2Real control framework [4] for training and deploying vision-based RL policies for surgical tasks. A thread placement task is designed in Unity to demonstrate the challenges of performing successful sim-to-real transfer of simulation trained RL policies, particularly on complex tasks. This task is used in conjunction with a baseline tissue manipulation task already implemented by Haiderbhai et al. [4] to evaluate the performance of the diffusion-based RL training pipeline proposed in this study. The following sections provide an overview of the methods used to formulate this investigation.

3.1 Thread Placement Task Design

3.1.1 Problem Formulation

Suturing is a medical technique used to hold tissues together for healing. This is done by medical practitioners using a needle and thread to sew the tissues together. Similar to previous works by Haiderbhai et al. [3] and Gondokaryono et al. [13], the task is simplified to focus on thread placement, one of the core sets of motions required to perform suturing successfully. The placement task remains challenging as it requires the RL agent to learn the 3D control necessary to place a rope segment into a cylinder's hole, solely based on visual observations. The problem is formulated as a visual servoing task to mimic surgeon teleoperation of the dVSS, where the PSMs are operated based on the first-person visual feedback that the surgeon receives from the endoscopic camera [3].

3.1.2 Simulation Setup

The thread placement simulation consists of a single PSM of the dVRK, a rope object based on the Obi Rope asset for Unity [3], and a cylindrical object with a hole in its center.

The rope is designed with three particles to provide an accurate representation of the real rope physics, with surface colliders to interact with the cylindrical object representing the placement target. Furthermore, the rope elasticity and bend constraints are visually tuned to mimic the ropes available in the lab for real experiments, giving values of 0.9 and 0.001 respectively within the Obi Rope configuration tool. One particle of the rope is attached to the PSM's end effector tool tip, while the other two particles are free to move based upon the motion of the manipulator. The cylindrical object is designed in the Fusion360 CAD environment and imported into Unity. The size of the cylinder, its hole diameter, and its position are randomized during RL training to ensure the agent learns to generalize to different hole sizes. Similarly, the rope physics and physical attributes are randomized and tuned through experimentation to aid in compensating for the sim-to-real gap. The simulation environment and the associated domain randomization parameter ranges are shown in Figure 3.1 and Table 3.1 respectively.

(a) (b) (c)

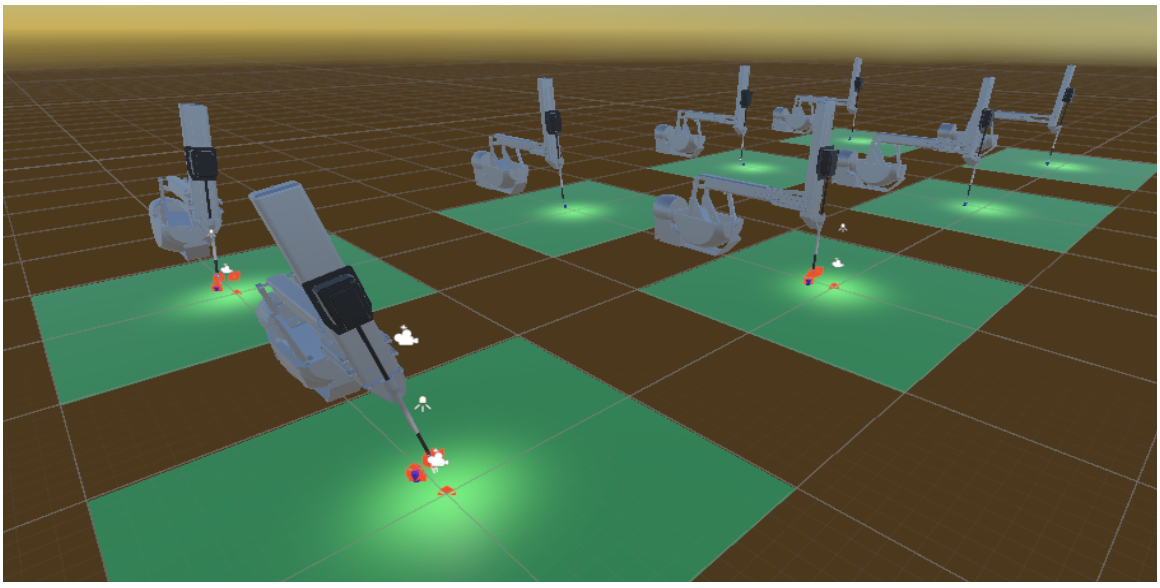
Figure 3.1: Thread placement simulation environment setup. (a) Initial state of the simulation with the rope attached to the PSM end effector. (b) Intermediate state of the simulation with the rope being guided towards the cylinder's hole. (c) Final state of the simulation with the rope successfully placed inside the cylinder's hole.

3.1.3 Policy Reward Function Design

The Unity ML-Agents framework is used to train a PPO-based RL agent to perform the thread placement task, similar to previous works which provide more depth on the framework formulation [3, 13]. The task is formulated as a Markov Decision Process (MDP) with $MDP = (S, A, P, r, \gamma)$, where the objective is to optimize a policy defining probabilistic transitions (P) from states ($s \in S$) to actions ($a \in A$) such that the agent maximizes the cumulative reward defined by r. In the thread placement task, the goal of the policy is to move the PSM end effector into a position that aligns the rope segment with the cylinder's hole, and then move the rope downwards into the hole. The reward function is designed

Table 3.1: Domain randomization parameters for the thread placement simulation.

Domain Parameter	Parameter Range
Light Intensity (lux)	[5, 50]
Light Position (m)	$x \in [-1.0, 1.0]$ $y \in [4.0, 6.0]$ $z \in [-2.0, 2.0]$
Shadow Probability	[0.0, 1.0]
Floor Static Friction (N)	[0.0, 0.3]
Floor Dynamic Friction (N)	[0.0, 0.3]
Camera Position (m)	$x \in [-0.1, 0.1]$ $y \in [1.7, 2.5]$ $z \in [-1.0, 1.0]$
Camera Orientation (°)	$x \in [45, 55]$ $y \in [-15, 15]$ $z = 0$
Cylinder Size Factor ([0.3 0.3 1] Default Scale)	$x, y \in [0.10, 0.12]$ $z \in [0.03, 0.05]$
Rope R Color	[0.5, 1.0]
Ground R Color	[0.0, 0.4]
Ground G Color	[0.6, 0.8]
Cylinder B Color	[0.25, 0.75]
Rope Thickness (m)	[0.02, 0.03]
Rope Stretch Factor	[0.97, 1.05]
Rope Length Variation	[0.0, 0.3]



(a)

Figure 3.2: 8 PSMs learning the thread placement task in parallel within the Unity simulation environment.

based on Unity's coordinate system, where the X-Z plane is parallel to the floor plane and the Y axis represents height from the floor origin. A secondary policy is also trained on a 2D version of the task for greater simplicity, constraining the action space and randomized cylinder position to only the Y-Z plane. The reward at each time step is calculated by Equation 1.

$$r(s) = \begin{cases} r_{\text{gf}}, & \text{if within close proximity above cylinder's hole} \\ r_{\text{gf}} + r_{\text{rt}} + r_{\text{ns}} + r_{\text{nf}} + r_{\text{en}}, & \text{otherwise} \end{cases} \tag{1}$$

Components:

$$r_{\text{gf}} = \begin{cases} 0.01, & \text{if } y_{\text{d}}(t) < y_{\text{d}}(t-1) \text{ and in proximity} \\ -0.01, & \text{if } y_{\text{d}}(t) > y_{\text{d}}(t-1) \text{ and in proximity} \\ -1, & \text{if only in proximity at t-1 timestep} \\ -0.0001, & \text{otherwise} \end{cases} \tag{2}$$

$$y_{\text{d}}(t) = |y_{\text{r}}(t) - y_{\text{gf}}(t)| \tag{3}$$

$$r_{\text{rt}} = f_{\text{exp}}(y_{\text{r}}, y_{\text{gb}}) + f_{\text{exp}}(x_{\text{r}}, x_{\text{gf}}) + f_{\text{exp}}(z_{\text{r}}, z_{\text{gf}}) \tag{4}$$

$$r_{\text{ns}} = -\frac{\text{Current Step Count}}{\text{Max Step Count for Episode}} \tag{5}$$

$$r_{\text{nf}} = \begin{cases} -1, & \text{if } y_{\text{tooltip}} = y_{\text{floor}} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$r_{\text{en}} = \begin{cases} -3.5, & \text{if } |x_{\text{r}} - x_{\text{gf}}| > 1 \text{ or } |z_{\text{r}} - z_{\text{gf}}| > 1 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$f_{\text{exp}}(a, b) = \exp\left(-\alpha \left|a - b\right|\right) \tag{8}$$

The reward function consists of sparse and dense components depending on the agent's

proximity to the cylinder hole. r_{gf} is the goal flag reward, which, when in proximity of the goal, provides a small reward for moving the rope downwards towards the completion flag/marker within the hole, and a penalty for moving the rope upwards. For this problem, close proximity is defined by a 0.05 Unity unit X-Z plane radius around the cylinder object, and a 0.2 Unity unit Y axis distance from the top cylinder surface. The proximity activation of this reward is determined experimentally within simulation to ensure the agent receives a continuous reward signal when attempting to place the rope above the hole. r_{gf} is automatically set to a reward of -1 if the rope's free end is outside of the hole proximity and was inside it at the previous time step. This logic is represented by Equation 2 and Equation 3. r_{rt} is the rope tool reward, which provides an exponential reward (Equation 8) based on the distance between the rope and the 3D coordinates of the cylindrical object. Specifically, the 3D position of the rope free end is compared to the 3D center position of the permeable goal flag disk placed within the hole to indicate that the rope is inside the hole and the task is complete. One caveat in this reward is that the Y coordinate of the cylindrical object surface (y_{gb}) is compared to the Y coordinate of the rope free end rather than the goal disk Y coordinate to provide the intermediate reward signal for first reaching a position above the hole. r_{ns} is the time penalty, which provides a negative reward based on Equation 5 to encourage the agent to complete the task as fast as possible. Furthermore, a large negative reward (Equation 6) is given if the Y coordinate of the PSM end effector (tool tip) is touching the floor to prevent the agent from spending many training steps exploring the floor bounds. Finally, r_{en} is an exploration penalty, which provides a large negative reward if the agent moves the rope too far away from the hole in the X or Z directions, thus preventing the agent from spending many training steps exploring areas far from the cylindrical object. The aforementioned rewards are summed together to complete the function case in Equation 1 where the rope is not within close proximity to the cylinder hole. The episode ends when the rope passes through the goal flag within the hole, giving the agent a reward of 2, or when the episode reaches the maximum step count of 10000.

Figure 3.2 shows the training environment consisting of 8 PSMs learning the thread placement task in parallel through ML-Agents. The policy is trained on an i7-9700K, Nvidia 3070, and 32GB of RAM for 30 million steps with 128×128 image observations and a time scale of 1 to ensure that the rope physics are being solved accurately at each step of the simulation. The rest of the PPO hyperparameters are set to the default values provided by the ML-Agents documentation [25]. Moreover, the trained policy is transferred to the real PSM for evaluation, making use of the Sim2Real control framework described in depth in previous works [3, 4, 13].

3.2 Diffusion Image Generation & RL Integration

3.2.1 Problem Formulation

This study aims to investigate the performance enhancements of vision-based RL policies that are trained using realistic simulation images generated from diffusion inference, as mentioned in Sections 1.1 and 1.2. This requires fine-tuning a diffusion model on real images collected from the dVRK and developing an RL training pipeline to integrate outputs from the diffusion model into the agent's observation space. Images generated by the fine-tuned diffusion model are evaluated to determine the effectiveness of the model in preserving simulated dynamics while providing a realistic visual representation of the task. Policies are trained for the mature tissue manipulation task presented by Haiderbhai et al. [4], with the objective of validating the diffusion-based RL training pipeline on a simpler task before applying it to the more complex thread placement task, which demands finer control of the PSM end effector. The results from training and evaluating the policies will provide researchers developing vision-based RL policies for surgical tasks with a new method to bridge the sim-to-real gap, in addition to the current domain randomization techniques.

3.2.2 Diffusion Model Training

As discussed in Section 2.3, the work presented by Kaleta et al. [18] is used as a reference for fine-tuning a Stable Diffusion model because of its success in generating realistic surgical images from simulation image inputs. Three separate models based on Stable Diffusion v1.5 are trained: two using images from real policy demonstrations of the tissue manipulation (block pushing) task, and another using images from real policy demonstrations of the thread placement task. The difference between the two tissue manipulation models is that one is trained on demonstrations with a solid color background [4], while the other is trained on demonstrations with a tissue textured background for evaluating the complexity capabilities of the diffusion model approach.

The LDM loss that is optimized during training is presented in Equation 3.1, where $\epsilon_\theta(z_t, t, \tau_\theta(y))$ is the sequence of denoising autoencoders over $t = 1...T$ and z_t is the latent representation ($\mathcal{E}$) of the input, x during the denoising process. $\tau_\theta(y)$ is introduced as a domain specific encoder that projects a conditioning input, y to an intermediate representation, therefore acting as a pre-processor for handling various condition modalities. ϵ_θ and τ_θ are jointly optimized within the squared L_2 norm, as computed with respect to the Gaussian noise ϵ, thus maintaining the training objective of ϵ_θ in predicting a denoised variant of z_t [8].

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right] \tag{3.1}$$

Training images are manually selected to cover the various procedural stages of the task, including when the PSM tool tip is not present in the image, and are preprocessed to a resolution of 512×512 before being used for training with Dreambooth. The Dreambooth training pipeline available on Hugging Face [24] is used to fine-tune both the text encoder and U-Net of the Stable Diffusion model, as recommended for producing the best quality images of each task based on the learned association with the text identifier [26]. Moreover, the tissue manipulation task models are bound to a "pushblock" unique text identifier during Dreambooth training, whereas the thread placement task model is bound to a text identifier named "ropehole".

As Dreambooth has the tendency to overfit and therefore hallucinate aspects of the training data, the model hyperparameters and number of training images are optimized experimentally through an iterative process of visual inspection. The sensitivity to the number of training images is experimentally observed during this process, as the dataset consists of similar images that primarily differ based on the PSM tool tip's changing position relative to the objects being manipulated in the scene. A single A100 GPU instance with 20GB of VRAM, 6 AMD EPYC 7413 CPU cores and 62GB of RAM is used to train each model with the experimentally tuned hyperparameters specified in Table 3.2. The use of Dreambooth and the available compute hardware result in training times of approximately 5 minutes, thus enabling rapid experimentation with different hyperparameters and dataset sizes. An overview of the described training methodology is shown in Figure 3.3.

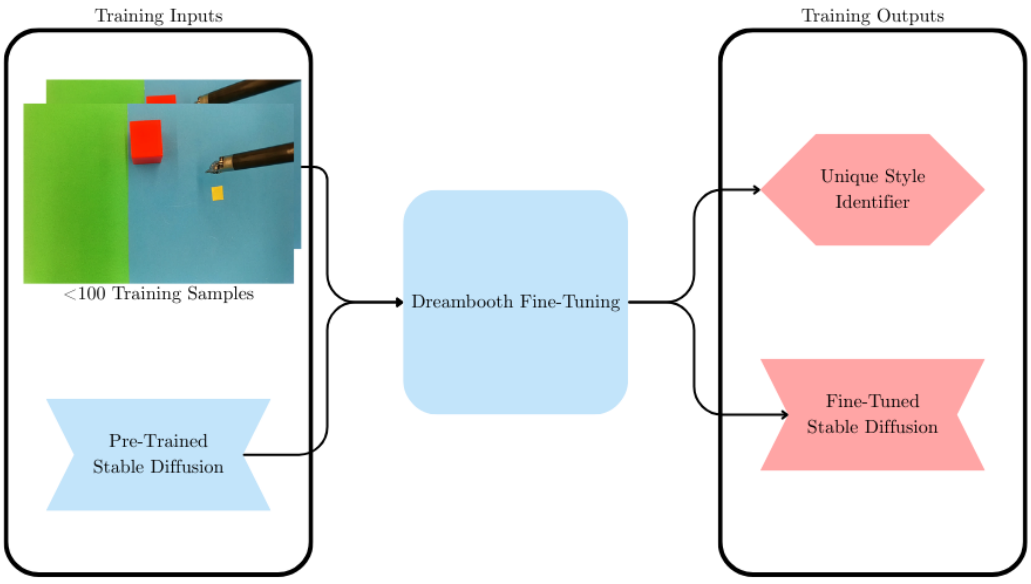


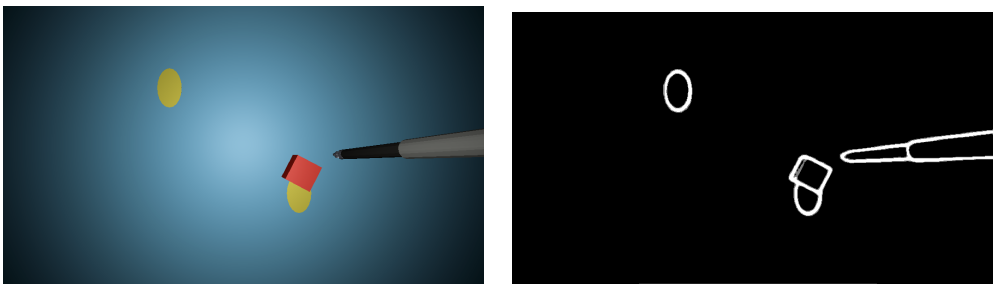
Figure 3.3: Fine-tuning diffusion framework for style transfer of various surgical tasks.

Table 3.2: Diffusion training hyperparameters and dataset size for each surgical task's model

Task Model	Dataset Size	Training Steps	Batch Size	Learning Rate
Pushblock (solid)	45	2000	4	1×10^{-5}
Pushblock (tissue)	30	500	4	5×10^{-6}
Ropehole	30	500	8	3×10^{-6}

3.2.3 Diffusion Inference Optimization

In the inference stage, the optimized models are applied in conjunction with ControlNet's SoftEdge v1.1 and Tile v1.1 models described in Section 2.3, to translate Unity simulation images into the realistic style associated with each task's text identifier. The SoftEdge control model uses edges detected with Pidinet [27] as conditioning for preserving the original edges seen in simulation. The Tile control model uses a downsampled version of the input simulation image as conditioning and internally splits the image into small tiles during processing to guide the diffusion model in generating realistic textures and details. An example of the condition images generated for ControlNet from a single tissue manipulation simulation image is shown in Figure 3.4.



(a) (b)

Figure 3.4: ControlNet condition images. (a) Original simulation image of the tissue manipulation task used for internal tiling operations within ControlNet. (b) SoftEdge control model condition image.

Similar to the hyperparameter optimization process for training the diffusion models, the ControlNet conditioning strengths and classifier-free guidance scale (CFG) are experimentally tuned individually for each task as it is observed that the variation in complexity of the model's training images result in significant differences in the denoising capabilities across the models [18]. Optimizing these parameters is crucial because the conditioning strengths determine how much weighting to apply to the condition images when generating the final output, while the CFG scale determines how strongly the diffusion model adheres to the text prompt versus its own learned biases and priors [22]. From visual inspection, a stronger

SoftEdge control used in combination with an equivalent or weaker Tile control is optimal for ensuring that PSM and object edges are preserved in the final output, and for preventing excessive texture details from being hallucinated. A summary of the selected inference parameter values for each model with 20 denoising steps is shown in Table 3.3.

Table 3.3: Optimal diffusion inference parameters for each surgical task's model

Task Model	CFG	SoftEdge	Tile
Pushblock (solid-background)	4.5	1.5	1.2
Pushblock (tissue-background)	5.0	0.6	0.55
Ropehole	5.5	1.35	0.35

3.2.4 Diffusion Inference Time Considerations

To integrate diffusion into the RL training pipeline, inference times are assessed to mitigate bottlenecks during observation collection by the RL agent. Inference, when not conducting RL training, is performed on an i7-9700K CPU, an Nvidia 3070 GPU, and 32GB of RAM, using 10 denoising steps. While higher denoising steps result in better quality images, the sequential denoising process is the primary bottleneck in achieving near-instantaneous inference times, and therefore, the number of denoising steps is reduced from 20 to 10 for use in the RL training pipeline.

Another factor influencing inference time is the choice of input and output image resolution. Stable Diffusion is trained on 512×512 images [8], making this resolution optimal for generating high quality outputs. Rombach et al. [8] note that generating outputs below 256×256 introduces significant artifacts and reduces quality. However, their work highlights that upscaling—where the input resolution is lower than the output resolution—still produces images of decent quality. This is because when upscaling, the model has already learned features at a higher resolution, allowing it to predict missing details more accurately than when forced to generate from an unseen, lower-resolution latent space. Fine-tuning Stable Diffusion on a dataset of images smaller than 512×512 is possible but requires extensive data and compute, which is beyond the scope of this study. Fortunately, a Stable Diffusion model fine-tuned on a large 256×256 dataset is available on Hugging Face, enabling comparison with the default 512×512 model when both are fine-tuned using the framework presented in Figure 3.3. With these insights, a timing analysis is conducted on 50 simulation images from the solid-background tissue manipulation task to determine the optimal input and output resolutions for the diffusion model, balancing quality and speed. Additionally, the analysis evaluates the feasibility of generating diffusion outputs at resolutions commonly used for efficient RL policy training, such as 64×64 or 128×128.

To further improve diffusion computational efficiency, pipeline optimizations such as the UniPCMultistep noise scheduler are employed to efficiently guide how noise is added and removed during the forward and reverse diffusion processes [28]. Furthermore, xFormers, a memory-efficient attention implementation is used to reduce memory consumption and increase speed during inference [29].

3.2.5 Diffusion Evaluation Metrics

The quality and realism of the diffusion model outputs are evaluated on established generative metrics such as the Frechet Inception Distance (FID) and Kernel Inception Distance (KID) [18]. FID and KID assess the similarity between real and generated image distributions through a comparison of their feature representations which are computed from a pre-trained inception network. While these metrics both evaluate image quality, FID assumes Gaussian distributions to measure similarity, whereas KID uses a kernel-based approach which is often more robust for smaller datasets [30]. For each task, 500 simulation images of the PSM performing teleoperated manipulation are collected and used to generate a synthetic dataset of diffusion model outputs. This is utilized to calculate the FID and KID scores with respect to evaluation datasets, each containing 500 images of unseen real task experiments gathered from the dVRK.

3.2.6 Vision-Based RL Pipeline

The final aspect of this study is to integrate the diffusion model capabilities into a custom RL pipeline to provide the agent with realistic images for training. The pipeline is designed using Gymnasium [31] and Stable Baselines 3 (SB3) [32], which are both widely used libraries for developing and training RL agents. This provides a flexible and efficient framework for training RL policies, allowing for easy integration of the diffusion model outputs into the agent's observation space and the use of various RL algorithms. The ML-Agents package provides a Gym-compatible wrapper for Unity environments, enabling the use of Gymnasium's API to interact with the Unity simulation through Python.

The pipeline is visualized in Figure 3.5. The modules in red represent the user defined components, such as the Dreambooth style prompt, the diffusion model, and a domain randomization configuration file. The pipeline is fully customizable and modular, allowing for easy experimentation with different diffusion models, hyperparameters, and training configurations. This flexibility is crucial for evaluating the impact of diffusion model outputs on RL agent training while maintaining the same functionality as the default ML-Agents pipeline for RL policy training. The modules encapsulated in blue represent the Gymnasium environment components, which include the Unity simulation environment of the task to

train the policy on, the Unity CameraSensor object within the simulation build for capturing observation images, and the diffusion observation wrapper, which is used to augment the observation space by performing diffusion inference on the simulation images. Due to the quality limitations for inference with input resolutions less than 512×512, as mentioned in Section 3.2.4, the pipeline highlights the use of 512×512 visual observations from the CameraSensor object for diffusion inference to ensure that the agent receives high quality images. This approach provides more opportunity to assess the effectiveness of the method in bridging the sim-to-real gap, as the agent is trained on the highest quality images possible.

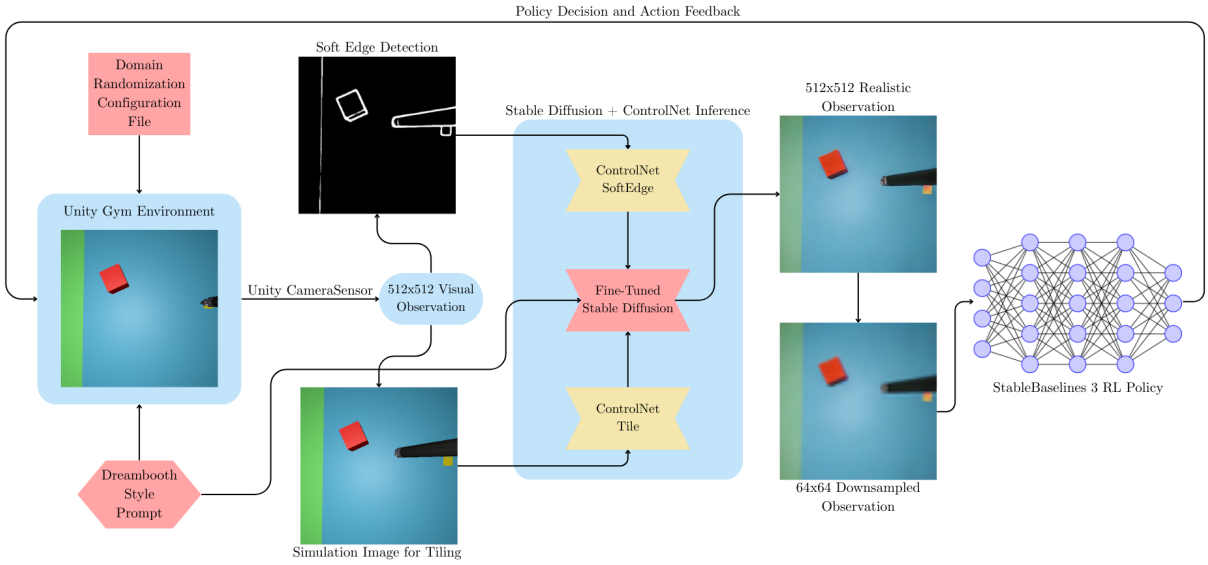


Figure 3.5: Diffusion inference RL pipeline for training vision-based policies of Unity surgical tasks using Gymnasium and SB3.

During training, the Unity simulation environment extracts a visual observation which becomes available within the pipeline as a Gymnasium observation object. The observation wrapper class acts as the diffusion processing module, using the visual observation to generate the edge mask and tile condition images required for ControlNet, and passing them into the diffusion model for inference along with the task style prompt and inference parameters. The realistic output image is downsampled to 64×64 resolution and returned from the observation wrapper class, where it is then passed into the RL agent's policy network as the observation input. The policy network uses the realistic image to predict the next action for the PSM to take in the simulation environment, creating a feedback loop of observation and action until the episode ends.

3.2.7 RL Policy Training

Due to the inference time limitations discussed in Section 3.2.4, agents are only trained on the solid-background tissue manipulation task. The tissue-background version and the thread

placement task are excluded from training, as their higher training step requirements would lead to excessive inference times. SB3 implementations of PPO and Soft Actor-Critic (SAC) are used to train the RL policies, as these algorithms are well-suited for continuous action spaces and have been shown to perform well on a variety of tasks [3, 4, 13]. The benefit that SAC provides over PPO is that it is an off-policy algorithm, meaning that it is more sample efficient and can learn from past experiences rather than requiring the agent to explore the environment in real-time. This often results in faster training times, therefore requiring less diffusion inferences as less observations are needed to be collected from the simulation environment. While this aids in addressing the denoising bottleneck from Section 3.2.4, SAC is more sensitive to hyperparameters, requiring more tuning to achieve optimal performance than with PPO [32].

The diffusion-based RL agents are trained using the recommended hyperparameters for SB3 algorithms [32], as shown in Table 3.4, using A100 GPU instances with 20GB of VRAM, 4 AMD EPYC 7413 CPU cores and 24GB of RAM. As the diffusion-based method only addresses the visual sim-to-real gap, policies are trained with varying ranges of physics, camera poses, and lighting domain randomization to ensure that the agent learns to generalize across different environments. The physics domain randomization is done by varying the static and dynamic friction of the floor, and the mass of the cube. The camera pose domain randomization is done by varying the camera position and orientation, while the lighting domain randomization is done by varying the light intensity and position. Comparing the real-world performance of diffusion-based policies trained with and without lighting domain randomization is important for assessing the magnitude of the sim-to-real gap that the diffusion model is able to bridge on its own. The ranges of the domain randomization parameters used for training the solid-background tissue manipulation task are shown in Table 3.5 and Table 3.6. Furthermore, baseline policies, with diffusion inference disabled, are trained through the pipeline to aid in understanding the effects of diffusion-augmented observations on policy learning behaviors and performance.

Table 3.4: Key hyperparameters used for SB3 PPO and SAC diffusion inference policy training for the solid-background tissue manipulation (pushblock) task.

Hyperparameter	PPO	SAC
Learning Rate	3.0×10^{-4}	3.0×10^{-4}
Batch Size	64	256
Buffer Size	2048	1.0×10^{6}
Epochs	10	N/A

Table 3.5: Domain randomization parameters for physics and camera variation (PhysCamDR) policy training on the solid-background tissue manipulation (pushblock) task.

Domain Parameter	Parameter Range
Floor Static Friction (N)	[0.0, 0.3]
Floor Dynamic Friction (N)	[0.0, 0.3]
Cube Mass (kg)	[0.01, 0.3]
Cube Mass Scale	[0.3, 0.5]
Camera Position (m)	$x \in [-1.0, 1.0]$ $y \in [2.9, 3.1]$ $z \in [0.0, 1.0]$
Camera Orientation (°)	$x \in [80, 100]$ $y \in [-3.0, 3.0]$ $z \in [80, 100]$

Table 3.6: Domain randomization parameters for full variation (FullDR) policy training on the solid-background tissue manipulation (pushblock) task.

Domain Parameter	Parameter Range
Light Intensity (lux)	[5, 50]
Light Position (m)	$x \in [-4.0, 4.0]$ $y \in [4.0, 7.0]$ $z \in [-2.0, 1.5]$
Floor Static Friction (N)	[0.0, 0.3]
Floor Dynamic Friction (N)	[0.0, 0.3]
Cube Mass (kg)	[0.01, 0.3]
Cube Mass Scale	[0.3, 0.5]
Camera Position (m)	$x \in [-1.0, 1.0]$ $y \in [2.9, 3.1]$ $z \in [0.0, 1.0]$
Camera Orientation (°)	$x \in [80, 100]$ $y \in [-3.0, 3.0]$ $z \in [80, 100]$

The decision frequency of the RL agent is a key consideration when training diffusion-based policies as it must be sufficiently frequent to ensure implicit temporal consistency of the image observations passed into the agent network. At the same time, the frequency must be kept low to mitigate bottlenecking the agent's decision-making process through

excessive diffusion inferences. By default, ML-Agents' decision requester period is set to 5, meaning that the agent will request a decision every 5 simulation steps. Using Unity's default fixed timestep of 0.02 seconds, this results in a decision being requested every 0.1 seconds (10Hz) when training with a time scale of 1. For most of the policies trained in this study, the decision requester period is set to 20, resulting in a decision being requested every 0.4 seconds (2.5Hz), which is sufficient to ensure that the agent's actions are temporally consistent with the images being passed into the policy network, while also reducing the number of diffusion inferences required to be performed during training. The policies trained for the solid-background tissue manipulation task are listed as follows:

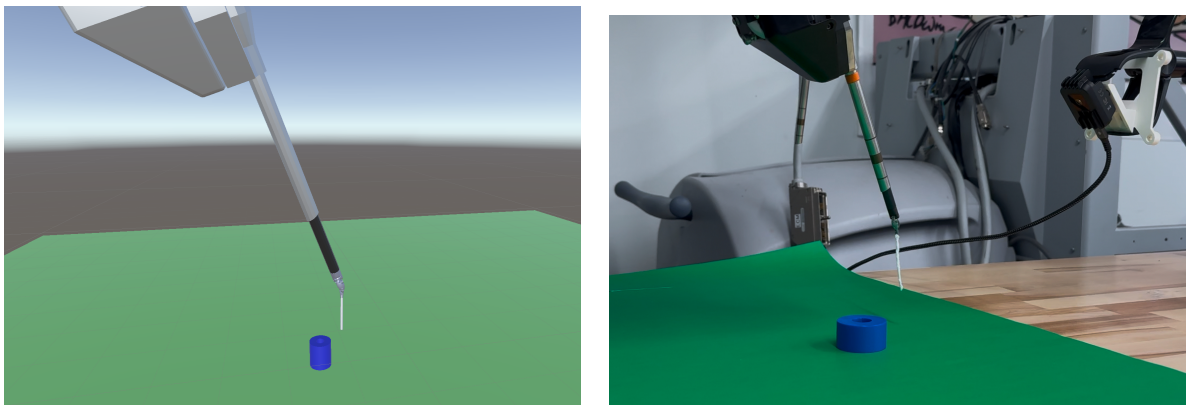
- **PPO-PhysCamDR**: diffusion-based PPO policy trained with physics and camera domain randomization, and a decision requester period of 20.
- **PPO-FullDR**: diffusion-based PPO policy trained with full domain randomization, and a decision requester period of 5.
- **PPO-NoDR**: diffusion-based PPO policy trained with no domain randomization, and a decision requester period of 20.
- **SAC-NoDR**: diffusion-based SAC policy trained with no domain randomization, and a decision requester period of 20.
- **Baseline-OrigParams**: PPO policy (no diffusion inference) trained with no domain randomization, and a decision requester period of 5.

Chapter 4

Results

4.1 Sim-to-Real Transfer for the Thread Placement Task

The experimental setup for the thread placement task consists of a scale-accurate 3D printed model of the dark blue cylinder object seen in the simulation environment, white yarn of various lengths and thicknesses to represent the rope segment, an Oak-1 camera and the Sim2Real control framework [4]. A comparison of the the simulation and experimental setup is shown in Figure 4.1, where the camera is carefully positioned to match the simulation camera pose.



(a) (b)

Figure 4.1: Comparison of the simulation and experimental setup for the thread placement task. Variations in lighting and object size are compensated for through domain randomization during training. (a) Simulation setup. (b) Experimental setup.

Figure 4.2: Cumulative reward over 30 million training steps for the 2D and 3D versions of the thread placement task. Training of the 2D version stops at 28 million steps due to observed convergence in policy performance.

Figure 4.3: Episode length over 30 million training steps for the 2D and 3D versions of the thread placement task. Training of the 2D version stops at 28 million steps due to observed convergence in policy performance.

In both simulation and experimental trials, the cylinder location is randomized in position and size to assess the generalization capabilities of the policy. The reward function defined in Equation 1 results in a simulation success rate of 75% for the 3D version of the task and 100% for the 2D version over 20 trials. Figure 4.2 shows the cumulative reward over 30

million training steps for both versions of the task, where the 3D version is able to achieve a maximum reward of 3.8, while the 2D version achieves a maximum reward of 1.5. The rewards are larger for the 3D version of the task due to the additional exponential term in Equation 4 for the alignment of the rope segment along the X-axis. Figure 4.3 shows the episode length over 30 million training steps for both versions of the task, where the 3D version is able to achieve a final episode length of about 850 steps, while the 2D version achieves a final episode length of about 1200 steps. Transferring these policies to the experimental setup emphasize the significant learning gap between the 2D and 3D policies as the 3D policy fails to complete the task in all 20 of the trials, while the 2D policy achieves a 25% success rate.

4.2 Evaluation of Diffusion-Based Image Generation

The fine-tuned diffusion model outputs demonstrate promising results for bridging the sim-to-real gap during RL training. The sim-to-real generations from the best model for each task are compared to both Unity simulation data and real experiment data for visual and quantitative evaluations. This is presented at a resolution of 896×512 due to the resolution constraints utilized by Unity Editor for saving simulation images without the RL CameraSensor component.

Table 4.1: FID and KID metrics between real and default simulation images for each surgical task. Lower scores indicate higher similarity.

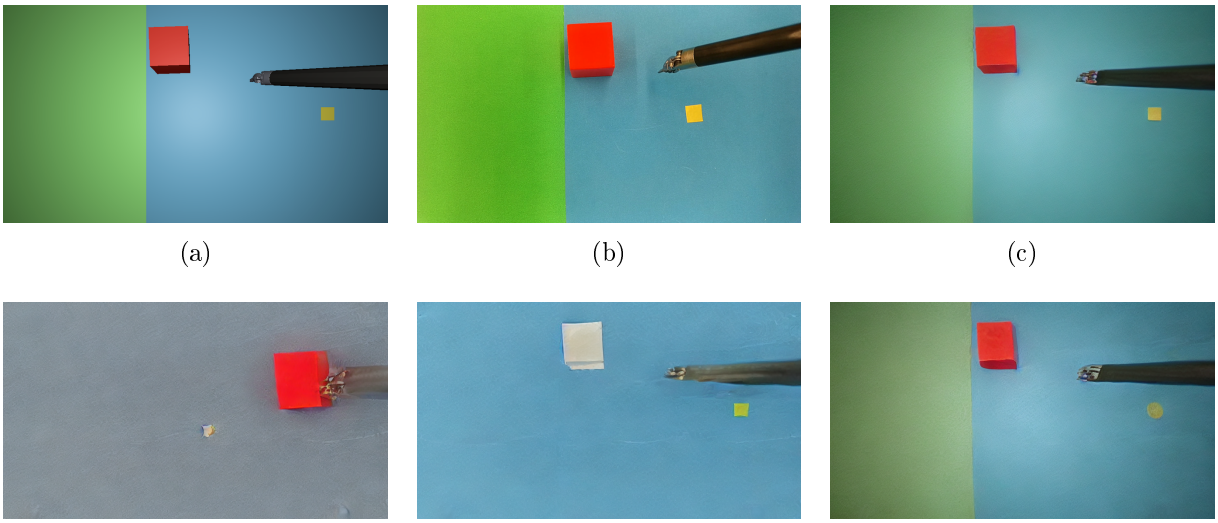
Task	FID (↓)	KID (↓)
Pushblock (solid-background)	256.46	0.272 ± 0.141
Pushblock (tissue-background)	305.58	0.466 ± 0.012
Ropehole	217.34	0.298 ± 0.007

Table 4.2: FID and KID metrics between real and diffusion generated images for each surgical task. Lower scores indicate higher similarity.

Task	FID (↓)	KID (↓)
Pushblock (solid-background)	121.03	0.121 ± 0.011
Pushblock (tissue-background)	192.73	0.207 ± 0.016
Ropehole	166.14	0.205 ± 0.008

The diffusion outputs of the solid-background tissue manipulation (pushblock) task show substantial improvements in FID and KID scores, with a reduction of approximately 53% and 55% respectively, compared to the baseline simulation as evidenced by Tables 4.1 and 4.2.

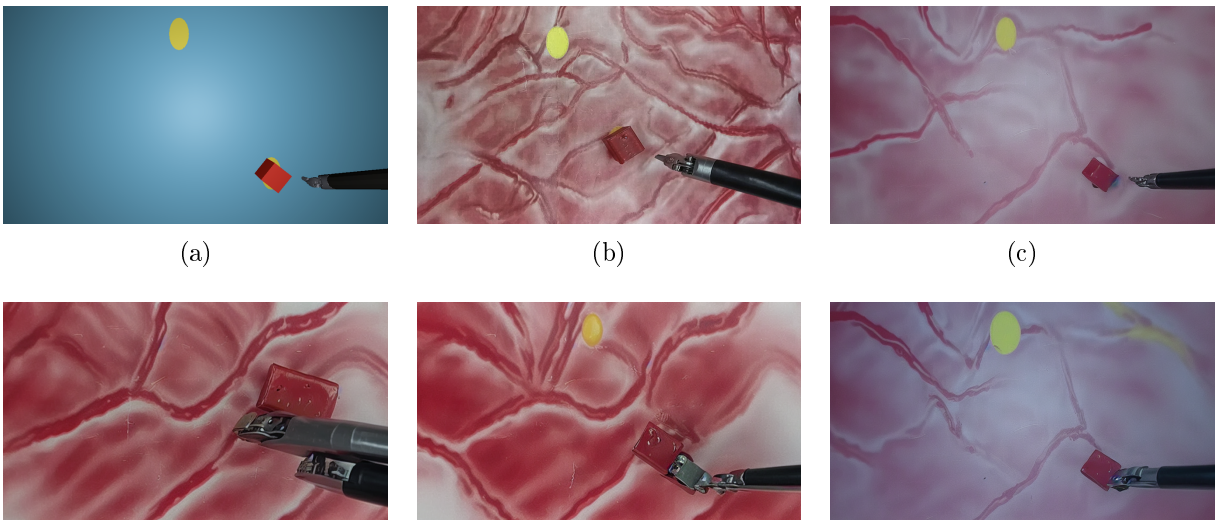
This result is visually supported by the comparison in Figure 4.4, where the best diffusion output, consisting of both SoftEdge and Tile control (Figure 4.4c), resembles the real image (Figure 4.4b) much more closely than the simulation image (Figure 4.4a).



(a) (b) (c)

(d) (e) (f)

Figure 4.4: Visual comparison between simulation, real, and diffusion model outputs for the solid-background "pushblock" task. (a) Simulation image. (b) Real image. (c) SoftEdge and Tile control. (d) No control. (e) SoftEdge only control. (f) Tile only control.



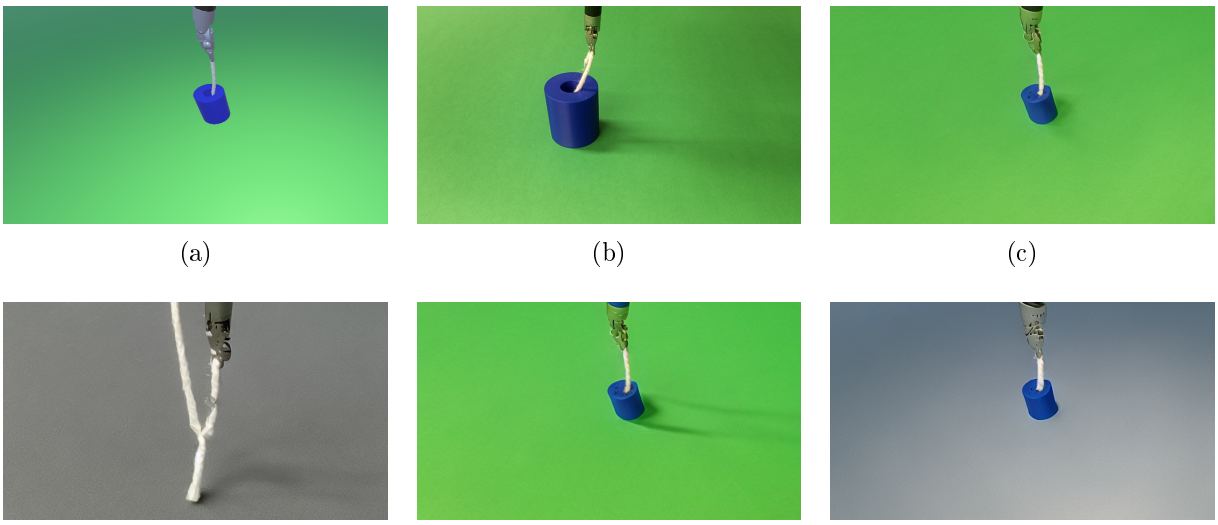
(a) (b) (c)

(d) (e) (f)

Figure 4.5: Visual comparison between simulation, real, and diffusion model outputs for the tissue-background "pushblock" task. (a) Simulation image. (b) Real image. (c) SoftEdge and Tile control. (d) No control. (e) SoftEdge only control. (f) Tile only control.

Regarding the more complex tissue-background tissue manipulation (pushblock) task, Tables 4.1 and 4.2 show a 37% reduction in FID and a 55% reduction in KID, indicating a

significant enhancement in the realism of the generated images compared to the simulation counterparts. The results are visually represented in Figure 4.5, where the diffusion outputs are able to capture the semantics of the tissue background present in the real image and apply them to the solid-background simulation image. Similar to the solid-background task, the combination of SoftEdge and Tile control (Figure 4.5c) produce the most realistic output.



(a) (b) (c)

(d) (e) (f)

Figure 4.6: Visual comparison between simulation, real, and diffusion model outputs for the "ropehole" task. (a) Simulation image. (b) Real image. (c) SoftEdge and Tile control. (d) No control. (e) SoftEdge only control. (f) Tile only control.

The thread placement (ropehole) task only shows a 24% reduction in FID and a 31% reduction in KID compared to the simulation images, as shown in Tables 4.1 and 4.2. The visual comparison in Figure 4.6 shows that the diffusion model is able to learn the rope texture and bright green background associated with the real image, but struggles to capture the hole in the cylindrical object. Diffusion generations for this task require careful fine-tuning of the ControlNet strength parameters, as the model is prone to generating washed out backgrounds such as the grey dominance seen in Figure 4.6d and Figure 4.6f.

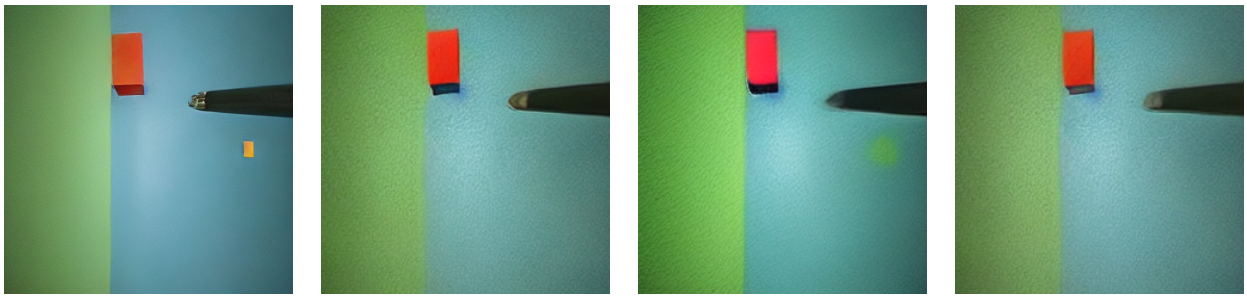
4.3 Timing Analysis of Diffusion Inference

Diffusion inference times are measured for various input and output resolutions, averaging over 50 generations with 10 denoising steps. The results are shown in Table 4.3, where the average time is calculated for each combination of input and output resolution. The results show that inference time is reduced by about one second when using a 256×256 output resolution compared to the baseline 512×512 output resolution, regardless of the input resolution. However, output resolutions lower than 256×256 are not shown in the table

as they produce outputs of complete noise, even when using the lower resolution 256×256 pre-trained diffusion model for inference. Although the baseline 512×512 model produces images with input and output resolutions of 512×512 at an average time of 3.32 seconds, the highest among the tested combinations, it is important to note that the output quality is significantly better than the other combinations. This is evident in Figure 4.7, where the 512×512 model is able to generate a high-fidelity image with clear details, while the other combinations struggle to produce similar quality outputs and begin to be dominated by noise. Figure 4.7 visualizes a subset of resolution combinations from Table 4.3 to provide a range of observed diffusion outputs, demonstrating the trade-off between inference time and image quality and highlighting the potential effects of resolution choice on diffusion-based policy training.

Table 4.3: Diffusion model inference time comparison for common RL input and output resolutions with 10 denoising steps and averaging over 50 generations. Pre-trained resolution represents what resolution the model was originally trained to denoise based on training data. 64×64 and 128×128 output resolutions are omitted as their outputs are purely noise.

Pre-Trained Res.	Input Res.	Output Res.	Average Time (s)
512×512	512×512	512×512	3.32
	256×256	256×256	2.55
	128×128	256×256	2.48
	64×64	256×256	2.45
256×256	256×256	256×256	2.47
	128×128	256×256	2.44
	64×64	256×256	2.41



(a) (b) (c) (d)

Figure 4.7: Visual comparison between diffusion generations for various input and output resolutions. (a) 512×512 model, 512×512 input/output resolution. (b) 256×256 model, 256×256 input/output resolution. (c) 512×512 model, 64×64 input, 256×256 output resolution. (d) 256×256 model, 64×64 input, 256×256 output resolution.

4.4 Training Performance of Diffusion-Based RL Policies

The diffusion-based policy training results are shown in Figures 4.8 to 4.12, where the mean reward, explained variance, entropy loss, policy gradient loss and value loss are plotted over one million training steps for the solid-background tissue manipulation task. In addition to policies defined in Section 3.2.7, the results also include a baseline PPO policy trained with no diffusion and adjusted hyperparameters from the recommended SB3 hyperparameters defined in Table 3.4. This is done due to the observed lack of convergence of the baseline policy with the original SB3 hyperparameters, indicating that the hyperparameters are not suitable for the solid-background task. More specifically, the batch size is increased from 64 to 256, the buffer size is increased from 2048 to 10240, and the number of epochs are decreased from 10 to 3. The mean reward curve in Figure 4.8 shows that these hyperparameter changes result in a significant increase in the mean reward, with the optimal hyperparameter baseline policy (shown in orange) achieving a maximum reward of 2.3, while the policies with the original hyperparameters achieve maximum rewards less than zero over the one million training steps. The SAC policy performs the worst out of all the diffusion-based policies, achieving the maximum reward of -1, likely due to the need for more careful hyperparameter tuning. Due to its instability and the lack of similar loss metrics to PPO as an actor-critic algorithm [32], SAC is not included in the explained variance and loss plots.

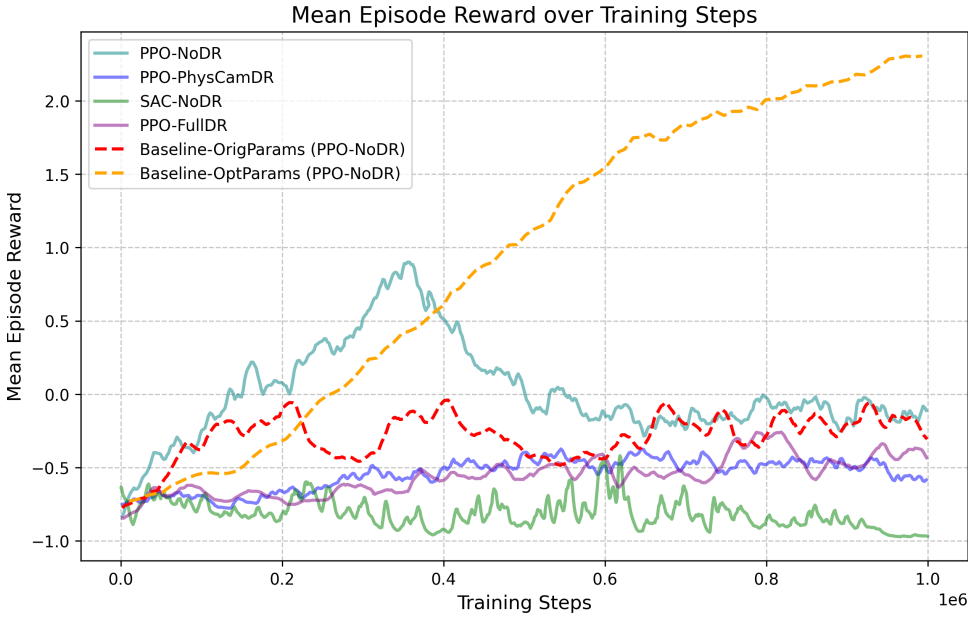


Figure 4.8: Mean reward over one million training steps for solid-background "pushblock" PPO and SAC policies. Baseline policies are trained with no diffusion inference, using both the original SB3 hyperparameters and the optimal hyperparameters.

The explained variance plots in Figure 4.9 visually depict how well of a predictor each policy is. Values closer to 1 indicate that the policy value function is a good predictor

of the cumulative reward returns, whereas values closer to 0 or negative indicate that the policy is not learning well. Most of the policies trend towards 1 with increasing steps, with the exception of the full domain randomization policy (shown in purple) which oscillates downwards towards -0.2.

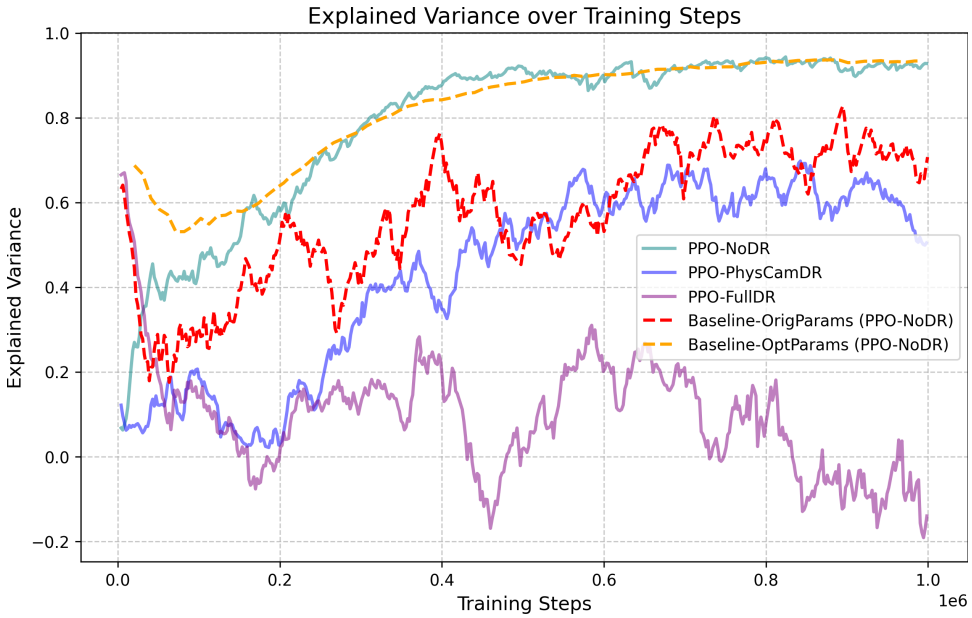


Figure 4.9: Explained variance over one million training steps for solid-background "pushblock" PPO policies. Baseline policies are trained with no diffusion inference, using both the original SB3 hyperparameters and the optimal hyperparameters.

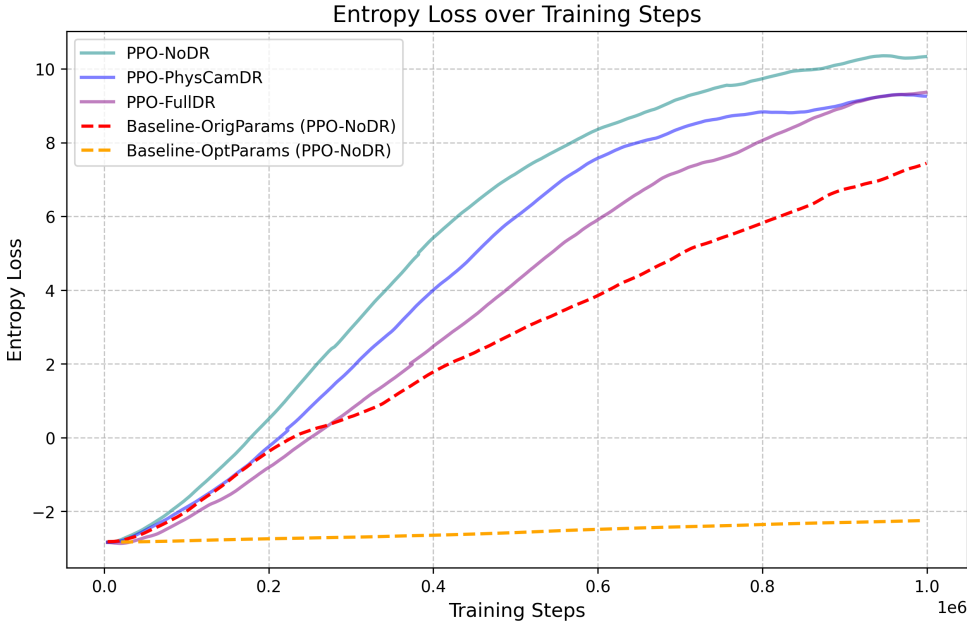


Figure 4.10: Entropy loss over one million training steps for solid-background "pushblock" PPO policies. Baseline policies are trained with no diffusion inference, using both the original SB3 hyperparameters and the optimal hyperparameters.

The entropy loss plots in Figure 4.10 visually depict how much exploration is being done by each policy. A gradual decrease in entropy loss is expected as the policy learns to exploit the environment and converge towards a maximum reward. The policies trained with the original hyperparameters demonstrate a large increase in entropy loss from values of -2.5 to 10 over one million training steps, whereas the optimal hyperparameter baseline policy only increases from -2.5 to -2.0 over the same training steps.

The policy gradient loss plots in Figure 4.11 and the value loss plots in Figure 4.12 aid in diagnosing stability and convergence issues during training. The policy gradient loss for all diffusion-based policies oscillate above zero as the number of training steps increase, with the optimal hyperparameter baseline policy showing the most stability and convergence in the shortest amount of time. This loss should decrease in earlier stages of training to indicate policy improvement, and then trend towards zero as the policy converges due to the gradients becoming smaller. The value loss is the mean squared error between the predicted state value and the true reward return for each state, meaning that as the policy becomes a better predictor of the returns, the value loss should trend towards zero. While lower value losses indicate better approximation of the reward function, values very close to zero often indicate overfitting or a lack of exploration. The trained diffusion-based policies demonstrate variance in value loss, with no clear trend towards a converged loss. The optimal hyperparameter baseline policy demonstrates some variance in value loss throughout the training process, but the original hyperparameter baseline policy quickly converges towards a value loss of zero with almost no variance.

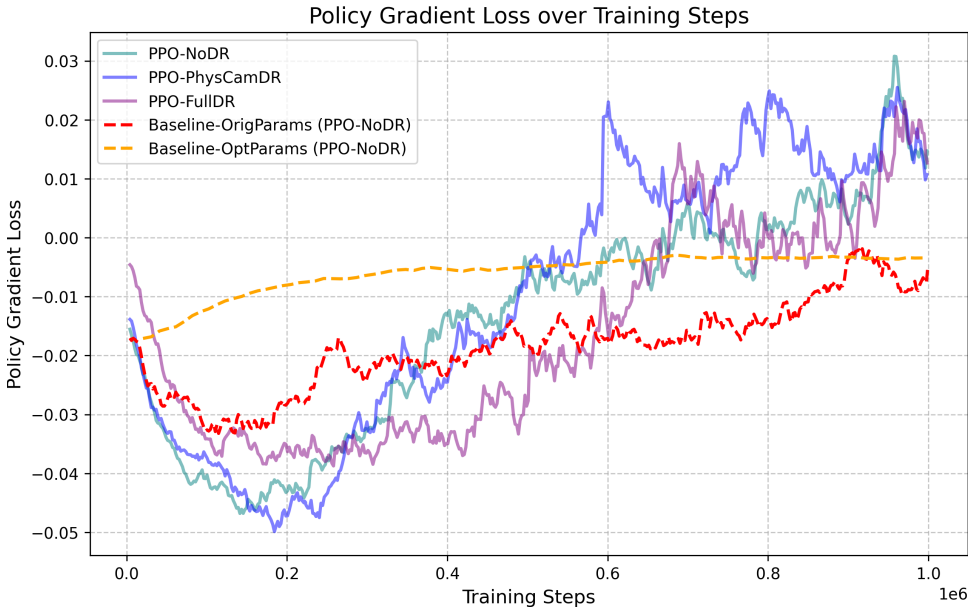


Figure 4.11: Policy gradient loss over one million training steps for solid-background "pushblock" PPO policies. Baseline policies are trained with no diffusion inference, using both the original SB3 hyperparameters and the optimal hyperparameters.

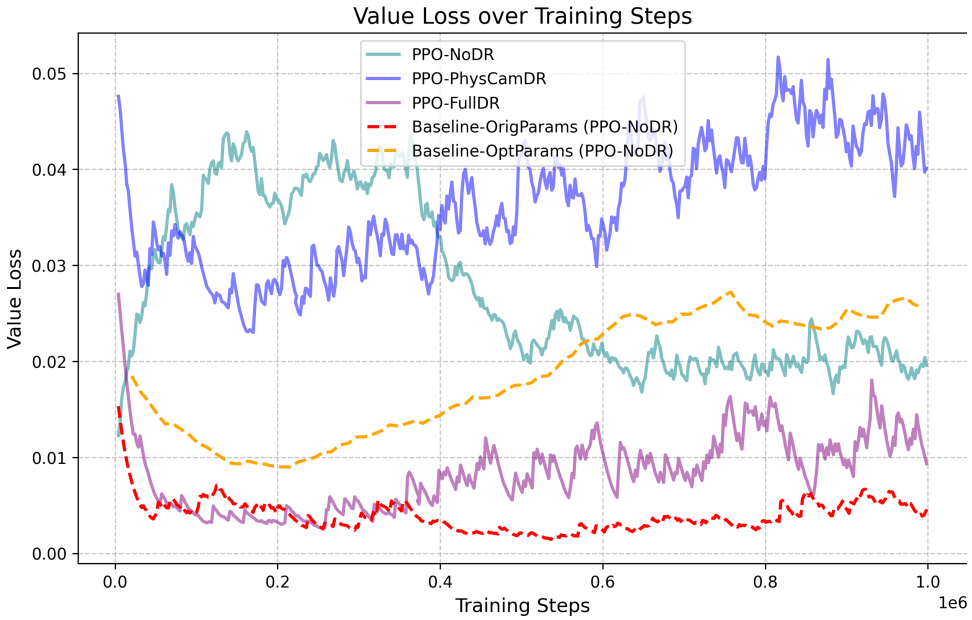
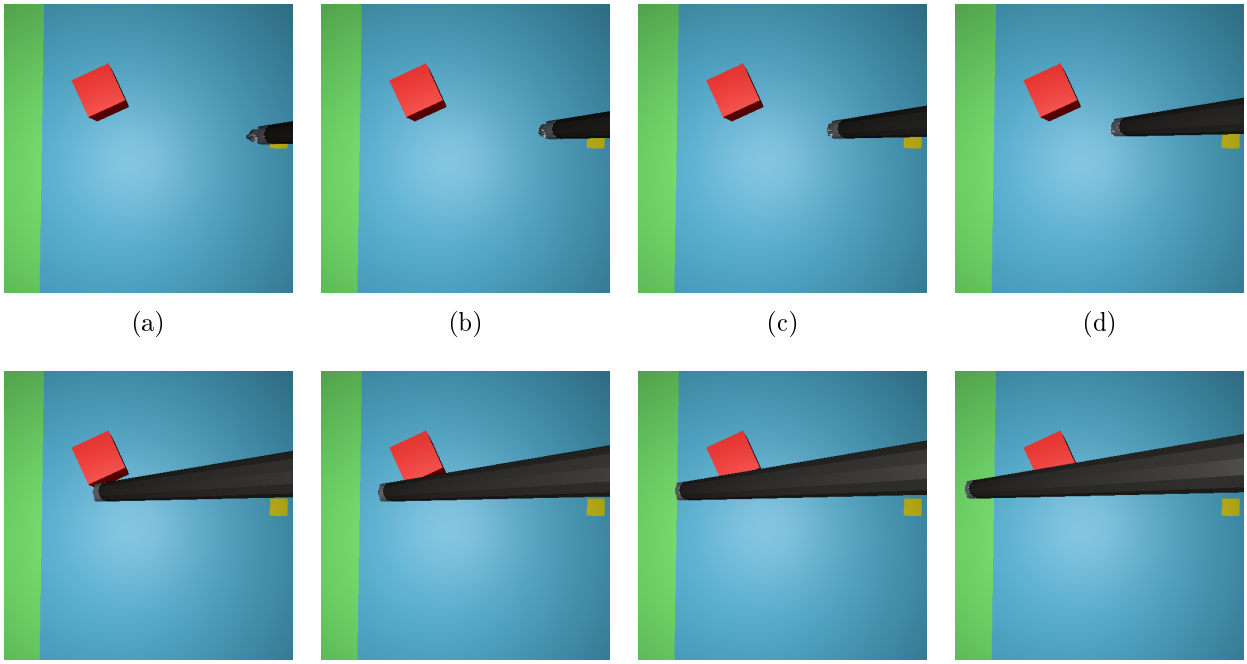


Figure 4.12: Value loss over one million training steps for solid-background "pushblock" PPO policies. Baseline policies are trained with no diffusion inference, using both the original SB3 hyperparameters and the optimal hyperparameters.

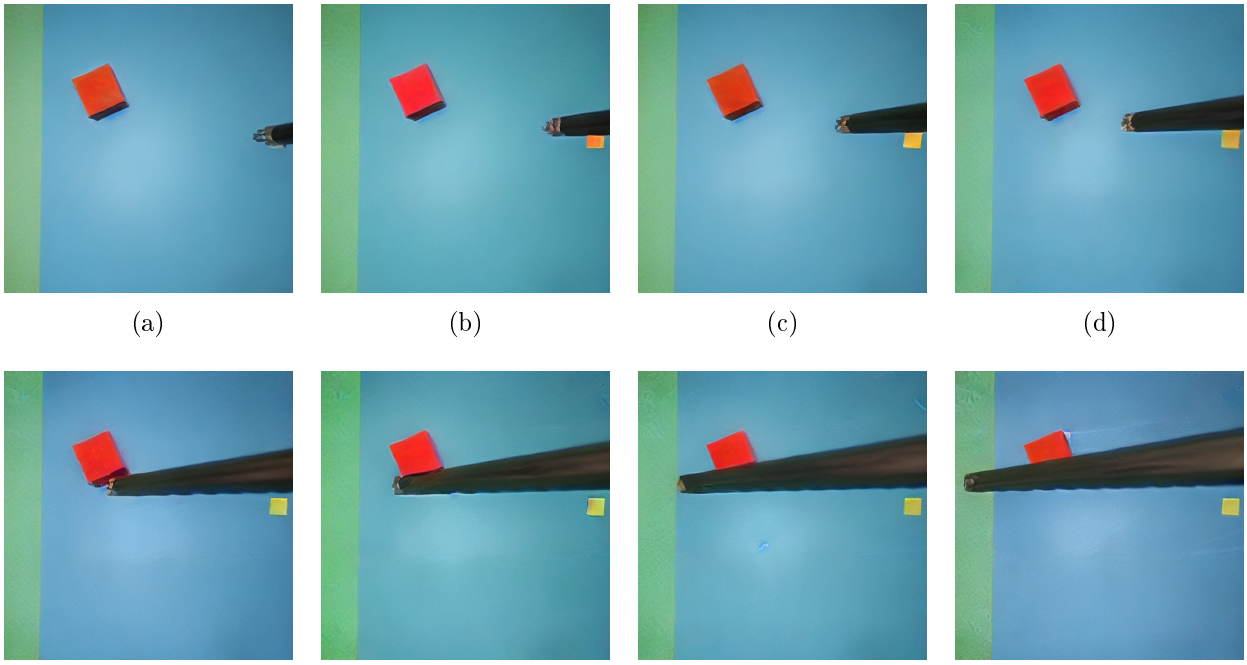
In addition to visualizing the training curves for the policies trained using the pipeline described in Figure 3.5, a sequence of Unity simulation observations and their corresponding diffusion generations used for RL policy training are shown in Figure 4.13 and Figure 4.14 for the solid-background pushblock task. The observations are collected when running inference on the PPO-NoDR policy, demonstrating that at the current state of training, the policy has only learned to move the PSM tool tip towards the green goal area, but not to push the block towards the goal area for further reward. Although the tissue-background tissue manipulation task is not used for policy training, a sequence of observations and corresponding diffusion generations for the task are shown in Figure 4.15 and Figure 4.16, confirming that this pipeline can be extended to more complex surgical task environments and conditions. The observations for this task are collected within the pipeline by teleoperating the PSM tool tip towards the red block. For both tasks, the time series of observations are depicted with increasing time from left to right.



(a) (b) (c) (d)

(e) (f) (g) (h)

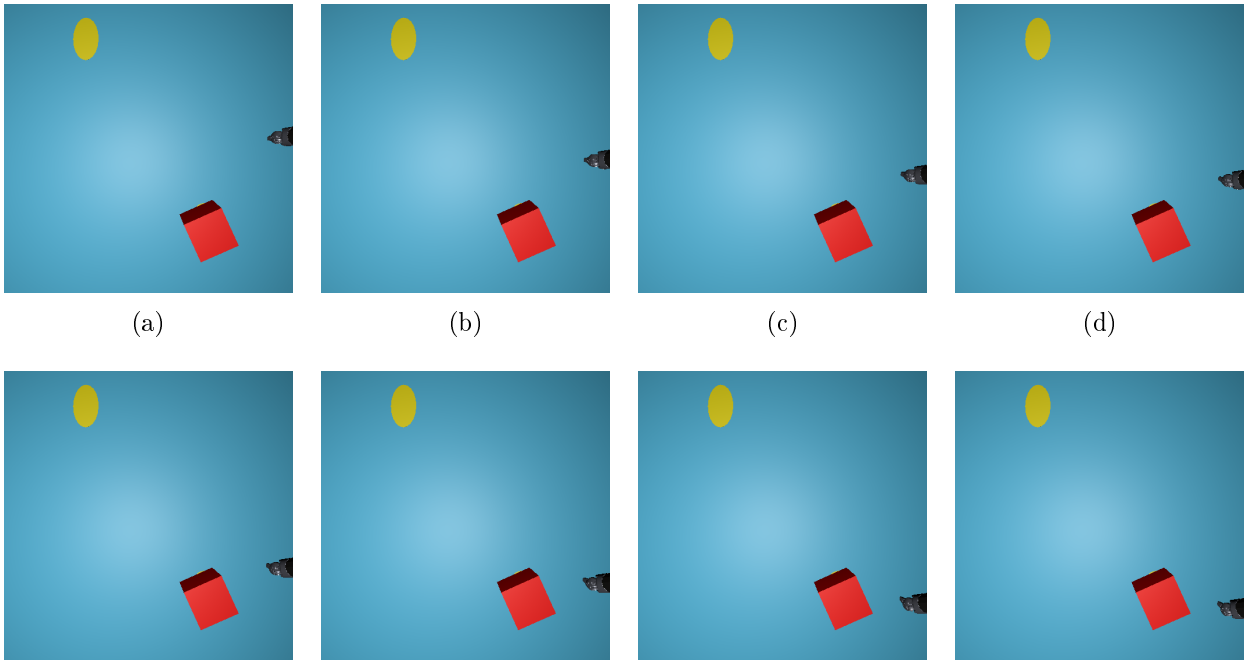
Figure 4.13: Time series of unaugmented simulation observations for policy training of the solid-background "pushblock" task.



(a) (b) (c) (d)

(e) (f) (g) (h)

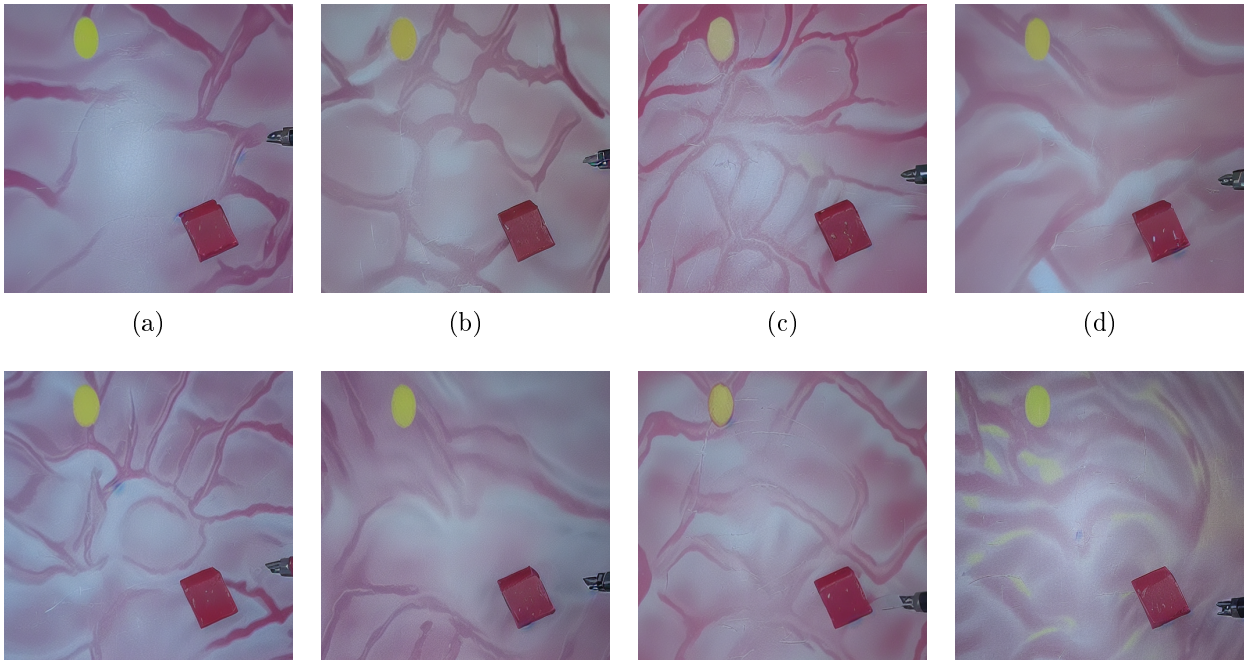
Figure 4.14: Time series of diffusion generated observations for policy training of the solid-background "pushblock" task.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.15: Time series of unaugmented simulation observations for the tissue-background "pushblock" task.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.16: Time series of diffusion generated observations for the tissue-background "pushblock" task.

Chapter 5

Discussion

5.1 Analysis of Thread Placement Task Performance

The cumulative reward and episode length curves in Figure 4.2 and Figure 4.3 provide insight into the learning progress of the 2D and 3D variants of the thread placement task. While the 2D version demonstrates a clear convergence trend, the 3D version does not exhibit similar behavior. Instead, it appears to remain in an early learning phase, as indicated by the gradual increase in rewards and the sharp decline in episode length with no plateau. This contrast suggests that the added complexity of interpreting depth information from a single camera view in the 3D task demands significantly more training for convergence. These observations emphasize the need for extended training or alternative strategies to accelerate learning in tasks requiring spatial reasoning from monocular inputs. The curves are also very noisy, indicating that a larger batch size may be required to stabilize the training process, potentially resulting in more consistent learning trends and improved convergence rates.

When transferring the policies to the real setup, significant variations in PSM control performance are observed depending on the noise produced from various lighting effects and the camera pose used, thus demonstrating the large domain gap between the simulation and real world environment for the task. For instance, when attempting to match the simulation camera pose with the experimental pose, it is observed that some camera placements result in the PSM moving randomly, while others result in the PSM moving towards the cylinder but with no correction in its trajectory for inserting the rope into the hole.

The task would benefit from wider domain randomization ranges compared to the values displayed in Table 3.1 to further decrease the domain gap, particularly with respect to the camera pose which consumed a significant amount of time to adjust for in the real setup. Furthermore, due to the depth complexity associated with the 3D version of the task, the

policy may benefit from a more sophisticated training approach such as curriculum learning to gradually increase the task complexity over time. With this, refinements to the default camera pose within simulation may be necessary for the 3D task, particularly by changing the camera pitch to orient further downward such that more depth features associated with the cylinder's hole can be observed by the policy. The discussed results provide a strong argument for the necessity of better domain adaptation techniques for effective deployment of vision-based RL policies, and can be used to assess the range of improvement associated with the proposed diffusion-based approach.

5.2 Impact of Diffusion on Simulation Realism

The degree of realism achieved from diffusion generated images is a crucial factor in determining the effectiveness of the diffusion models for sim-to-real transfer. The solid-background tissue manipulation task is able to achieve the largest FID and KID reduction between simulation and diffusion outputs, demonstrating the potential of this technique for RL training. This is reinforced through the visual comparison in Figure 4.4, where the best diffusion model output, Figure 4.4c, more closely resembles the real image compared to the simulation image through the color and texture of the red block and the green goal background. Figures 4.4c, 4.4d, 4.4e and 4.4f demonstrate the effectiveness of using ControlNet SoftEdge and Tile during inference, with Tile control guiding texture generation and SoftEdge control enhancing the edges of the generated image. This is particularly important for generating temporally consistent tool tip poses, as Figure 4.4d demonstrates the loss in pose information when no control is used. However, the diffusion and ControlNet combined output in Figure 4.4c still exhibits some artifacts in the generated image, particularly along the tool tip as the model attempts to represent the shine discoloration present in the real image, and at the edges of the red block where shadows tend to be defined. Furthermore, the best results are unable to completely capture the brightness of the red block and green goal background, likely due to the high Tile control weighting used in the inference process to achieve realistic textures. This limitation is consolidated during RL training, where standard domain randomization techniques introduce intensity variation to improve policy generalization. The use of Tile control also introduces artifacts of the simulated circular spotlight into the diffusion outputs, as its consistent presence in the simulation image causes the conditioning control to overemphasize its features when guiding local detail generation during the denoising process. Future experiments with similar task setups should consider removing the spotlight from the Unity simulation environment to prevent this effect from appearing in the realistic generated images. This is because the diffusion model implicitly learns the lighting conditions present in the training data distribution, therefore preventing the need for a spotlight to simulate the real-world conditions.

The tissue-background tissue manipulation task is significantly more challenging for the diffusion model to learn compared to the solid-background version due to the complexity of the tissue texture and colors. This is evident in the visual comparison in Figure 4.5, where even the best output (Figure 4.5c) struggles to fully capture the texture intensity features of the real style. The image generation results also reveal that the model can mistakenly identify the yellow markers and PSM end effector as tissue segments due to the high variance in background shape features, resulting in red color artifacts appearing on these elements. However, the best model is able to accurately depict various tissue patterns and introduces reflections on the PSM end effector, which are not present in the simulation images. Similar to the solid-background task, the combination of SoftEdge and Tile control during inference is crucial for balancing texture and structure of the diffusion outputs, while also maintaining the pose information of the PSM end effector and minimizing hallucinations. This is further illustrated in Figures 4.5d, 4.5e, and 4.5f, where the lack of ControlNet leads to the generation of hallucinated objects, SoftEdge control alone overlooks fine details in the simulation image, and Tile control alone results in a loss of segmentation between the block and the tool tip. It is also observed that the use of Tile Control leads to a noticeable reduction in red color intensity in the tissue background of the real image. This effect is likely caused by the dominance of white and pink tones in the local tissue texture, which are further emphasized by Tile Control's spatial attention mechanism. As this mechanism focuses on generating fine-grained local details through tiling, it may inadvertently suppress global color features such as the deep red from the tissue vessels [22].

The model performs the worst when generating realistic outputs for the thread placement task. This is visually depicted in Figure 4.6, where the best diffusion output, Figure 4.6c, struggles to capture the hole in the cylindrical object. This is likely due to the lack of a strong intensity gradient between the cylinder surface and the hole, which results in the hole being undetected in the SoftEdge conditioning image. Thus, future experiments should include a stronger gradient between the hole and the cylinder surface in simulation to improve inference quality. Nonetheless, the model is able to generate realistic rope textures while also capturing the green hues that often reflect off of the tool tip in the real images. Among the control variations, SoftEdge control alone (Figure 4.6e) captures the real style's background color and shadows more accurately than Figure 4.6c, but it fails to generate fine details such as PSM end effector colors and simulation consistent rope-tool tip interactions without Tile control. For this task, Figures 4.6c and 4.6e exhibit similar behavior due to the significant difference in ControlNet weightings between SoftEdge and Tile for optimal inference, as previously outlined in Table 3.3.

Comparing image generations throughout the fine-tuning process demonstrate the ease of overfitting when using Dreambooth, particularly for the datasets used in this study because

of their low variance in style features. While higher training steps improve tool tip quality, the background textures are often forgotten, leading to washed out backgrounds in the generated images, and therefore requiring stronger ControlNet weightings to recover image semantics through the condition images. Overall, the results indicate that the framework presented in Figure 3.3 is effective for fine-tuning diffusion models for sim-to-real transfer on surgical tasks, but the model's performance is highly dependent on the feature variance of the training data and careful selection of ControlNet parameters during inference.

5.3 Implications of Diffusion Inference Latency

The inference time of the diffusion model is a critical factor in determining its feasibility for real-time applications in training vision-based RL policies. The timing results presented in Table 4.3, paired with the visual observations in Figure 4.7, highlight that although inference time decreases by approximately one second when using lower resolutions compared to the best-performing 512×512 input and output configuration, the speed improvement comes at the cost of a significant drop in image quality. In particular, diffusion outputs generated at lower resolutions exhibit a notable loss of spatial coherence along the robot tool tip. This degradation introduces visual artifacts and blurring that obscure the tool's precise location and orientation, which are critical cues for policy learning. In the context of RL, such noisy or imprecise visual observations can mislead the agent, making it difficult to infer the correct action at each timestep. As a result, despite the faster inference, lower-resolution settings are not viable for policy training, where accurate perception of task-relevant features like the tool tip is essential for successful learning and sim-to-real transfer.

Inference improvements were also explored by converting the diffusion model into ONNX format, which is designed to perform graph optimizations and reduce model size for faster inference. However, GPU profiling during ONNX-based diffusion inference revealed frequent memory transfers between the CPU and GPU. These transfers significantly increased inference time, even when using IO binding—a technique that pre-allocates GPU memory to reduce data movement overhead.

Due to the limitation of the diffusion denoising process being the most computationally expensive step, methods of performing few-step denoising have been proposed to reduce the time complexity of sequential image generations. One approach called InstaFlow by Liu et al. make use of the curved denoising trajectory created through the use of a Gaussian noise distribution in the diffusion process, to learn an ordinary differential equation (ODE) that can be used to generate images in a single step [33]. More specifically, given a noisy latent vector in the source distribution, Z_0, the learned ODE generates a single step linear approximation of the denoising trajectory, mapping Z_0 to the denoised latent vector in the

target distribution, Z_1. The ODE is learned using a pre-trained Stable Diffusion model, effectively straightening the denoising trajectory while preserving the marginal distributions of the learned denoising behavior. The paper requires this process, termed Reflow, to be trained on a pre-trained diffusion model for 199 A100 days to achieve the one step denoising process. This represents a significant limitation of the proposed framework, as the diffusion model must be trained from pre-trained weights to learn the ODE denoising trajectory. Consequently, the publicly available Reflow model cannot be directly used with the models trained in this study, since the weights have been modified and would therefore require Reflow training. However, the work notes that Low-Rank Adaptation (LoRA) can be applied to the public Reflow model for single-step denoising. Since LoRA adds a low-rank matrix of learned weights to the pre-trained weights during inference to apply a style without modifying the base model, this approach is worth exploring in future work to reduce training cost while maintaining quality [23].

5.4 Learning Behaviors of Diffusion-Based RL Policies

The RL training pipeline presented in Figure 3.5 is able to successfully train vision-based policies using diffusion-generated images as input observations. However, sim-to-real experimentation is significantly restricted by the fact that training diffusion-based policies require about 20 days to complete one million training steps, which is much longer than the baseline PPO policy training time of about a day. As discussed in Section 5.3, diffusion inference is slow, and therefore the training time is significantly increased when using diffusion-generated images as input observations. This makes it impractical to tune hyperparameters and evaluate the performance of various versions of the policies within a reasonable time frame, thus resulting in tuning only occurring for the baseline policies. For this reason, the performance of diffusion-based policies are not evaluated with respect to the current domain randomization approaches for sim-to-real transfer in this study, but are instead only assessed in simulation and compared to non-diffusion baselines to understand learning behaviors with this approach.

Amidst training the diffusion-based policies, it was observed that the trained baseline PPO policy with no diffusion inference and hyperparameters from Table 3.4 was unable to converge within the one million training steps, although expected to based on previous work with the task [4]. From visualizing the reward curves in Figure 4.8, it is evident that the policy hyperparameters are not optimal for solving the task within the one million training steps, as the reward curves are noisy and do not exhibit strong trends towards maximizing the reward. This behavior is found to be attributed to the low batch size of 64, which is not sufficient for reducing the noise of the gradient updates during training for this task, and

therefore the reward signals are not stable enough for the policy to learn effectively. Another contributing factor to the slow and noisy learning process is the use of a small buffer size of 2048 and 10 epochs of backpropagation for policy gradient updates with the buffer samples. The reward curves demonstrate that the combination of a small buffer size and large number of epochs leads to overfitting of the policy to the current buffer of observations, which then causes more drastic changes in reward signals when new observations are added to the buffer and the policy acts on them. Consequently, the batch size was increased to 256, the buffer size was increased to 10240 and the number of epochs was reduced to 3 for the PPO policies. The increase in buffer size allows for a more diverse set of observations to be collected prior to backpropagation, as more simulation steps are taken and therefore more sets of visual observations, actions and associated reward signals are collected. By decreasing the number of epochs, the policy is able to mitigate overfitting to the current observation set, providing more stable gradient updates over the training process. A baseline PPO policy was trained with these hyperparameters (OptParams) and plotted against the previous policies in Figure 4.8. The optimal hyperparameter policy is able to converge within the one million training steps as it exhibits much less noise in the reward signal and a clear upward trend towards maximizing the reward. A similar analysis can be performed for the SAC policy, but it is omitted here due to the more sensitive tuning process, which is not as relevant to the discussion of the diffusion-based policies.

While the diffusion-based policies are not trained with the optimal hyperparameters, the results in Figure 4.8 demonstrate that they follow the learning trends of the baseline policies, emphasizing that the diffusion model is able to generate images that are semantically similar to the simulation images. This is seen when comparing the reward curves over the entire training process, where the diffusion-based policies converge towards following similar reward signal trends as the original hyperparameter baseline policy. The variance in reward signals between policies is likely due to the implicit domain randomization that occurs when using the diffusion model for generating images, as well as the additional configured randomization parameters set for the particular policy through the pipeline. This observation is significant because it indicates that results from tuning hyperparameters for the baseline policies can be directly applied to the diffusion-based policies, as the similar reward trends suggest that the policies are learning similar features from the generated images. Moreover, this provides a high likelihood that the diffusion-based policies for this task will be able to converge towards a maximum reward when trained with the optimal hyperparameters used for the baseline, thus saving computational resources and time when attempting to tune diffusion-based policy hyperparameters for future experiments.

In addition to confirming the functionality of the diffusion-based RL pipeline through the reward curves, the learning behavior of the PPO policies over the one million training steps is

analyzed through Figures 4.9 to 4.12. The explained variance curves in Figure 4.9 illustrate the effectiveness of the optimal hyperparameters in enabling the policy to learn the task efficiently, as the explained variance steadily converges toward 1 within one million training steps, without exhibiting significant noise. The same stability is not observed for the policies trained with the original hyperparameters, as the curves experience many fluctuations while slowly trending towards 1. One outlier is the no domain randomization (NoDR) diffusion-based policy, which closely follows the optimal hyperparameter policy in explained variance. As this policy is unable to learn to maximize the reward within the given number of training steps, it is likely that the policy value function is overfitting to a particular set of observation cumulative reward returns, causing the explained variance to be high but not applicable to the entire reward signal distribution. This follows the behavior seen in Figure 4.14, where the NoDR diffusion-based policy only moves the PSM tool tip straight towards the goal region, with little to no exploration of the environment during inference. However, the effects of domain randomization are noticed through the full domain randomization (FullDR) diffusion-based policy, which does not trend towards 1 over the training process. This is expected as the high variance in simulation parameters such as camera pose, physics and lighting conditions leads to a more complex cumulative reward return distribution, which takes longer for the policy value function to learn. Overall, the explained variance curves demonstrate that the diffusion-based policies are learning the task observation and action spaces very slowly due to the noise introduced by non-optimal hyperparameters and would likely learn a value function representative of the entire reward signal distribution faster if trained with the optimal hyperparameters, which provide more stable gradient updates.

The stochasticity of diffusion-augmented observations is evident in Figure 4.10, where the entropy loss of the diffusion-based policies is much higher than that of the baseline policy with the same hyperparameters. This indicates that the diffusion-based policies are exploring a wider range of actions during training due to the variance present between sequential observations, which is beneficial for generalization when transferring to the real world. However, it is clear that the non-optimal hyperparameters are causing the policies to overexplore the action space, as the entropy loss is significantly higher than the optimal hyperparameter baseline policy, which is resulting in slow convergence. The difference in entropy loss values between the original hyperparameter policies and the optimal hyperparameter baseline policy is likely due to the change in stability of the gradient updates, where large gradient changes result in the policy adjusting for more exploration to learn a better action distribution with less noise. The figure also indicates that even with the optimal hyperparameters, trained policies would benefit from more than a million training steps as there is no visible decreasing trend in the entropy loss. By increasing the number of training steps in future experiments, the diffusion-based policies—trained with the optimal

hyperparameters—may converge toward a more confident action distribution, potentially resulting in improved real-world performance.

The discussed stability and convergence issues observed when using the non-optimal hyperparameters for policy training are directly visualized through Figure 4.11 and Figure 4.12, where only the optimal hyperparameter baseline exhibits a smooth curve with converging behavior. The benefit of visualizing the policy gradient loss curves is that it demonstrates the impact of noisy realistic observations on the speed at which the diffusion-based policies are able to achieve more confident action predictions. The differences in loss between the diffusion-based policies and the baseline policy with the same hyperparameters show that the stochasticity introduced through diffusion and domain randomization result in a significant change in the optimization landscape of the task, providing opportunities for the policy to learn more complex features of the task that may be more applicable to real-world scenarios. These differences illustrate the sim-to-real distribution gap present between vision-based policies trained solely on simulation images and those trained with more realistic conditions—a contrast further emphasized by the disparities observed in the value loss curves in Figure 4.12. The increased value loss observed in the diffusion-based policies compared to the baseline highlights the challenge of learning from realistic observation representations. However, this also suggests that, when trained for longer periods with optimal hyperparameters, these policies have the potential to become more robust and better suited for noisy, real-world applications. Nevertheless, the diffusion-based policies maintain the general trend of loss convergence toward zero, albeit with increased noise, as they attempt to learn how to predict rewards of task states and actions from stochastic observations. This behavior underscores the potential of diffusion-based methods in bridging key visual aspects of the sim-to-real gap.

The comparison of time series observations of the solid-background tissue manipulation task in Figure 4.13 and Figure 4.14 demonstrate the working functionality of the pipeline presented in Figure 3.5. The diffusion-augmented observations are able to capture the key temporal pose features of the PSM tool tip, while also maintaining the details of the simulation image, but in a realistic style. This demonstrates the validity of using the pipeline for training vision-based RL policies, as the key semantics of the task required for effective policy learning are preserved in the generated images. The diffusion observations in Figure 4.14 demonstrate the stochastic nature of the diffusion model, as the generated images are not identical to one another in features, but rather exhibit variations in color, brightness and texture. This stochasticity is beneficial for RL training, as it introduces diversity in the training data, which can help the policy generalize better to unseen scenarios. For instance, the variance of the background colors and textures in sequential observations can help the policy learn to adapt to various levels of noise in the image, which is often present

in real-world surgical cameras such as endoscopes.

Similar observations are made when comparing Figure 4.15 and Figure 4.16, where the generated tissue background at each timestep consists of a different texture pattern and color intensity. The stochasticity of the background texture and colors provide the policy with a more realistic representation of the surgical environment, which can help it learn to adapt to the noise and variability present in real-world scenarios. However, Figure 4.16 also highlights a limitation of the diffusion model in achieving complete pose consistency for the PSM tool tip. The time series reveals that fine pose details, such as the downward orientation of the tool tip, are not entirely preserved in the generated images and are instead replaced by a more upright orientation. The recurrence of this observation throughout the time series indicates that this problem can be solved by including more variation of tool tip poses within the training data, as well as increasing the SoftEdge and Tile control strength during inference for effective RL training. Another hallucination is observed at the final timestep of the time series, where yellow shading can be seen along the tissue background. This is likely due to the presence of sections of the yellow oval in many of the training images. While this particular problem should not have a significant effect on the policy learning process, minimizing the amount of hallucinations in the generated images is important for ensuring that the policy learns to adapt to the correct features of the task. This indicates the potential need for a larger dataset with more variance in tissue textures to guide the diffusion model towards developing a better understanding of the real task style distribution.

The insights gained from analyzing the diffusion-based policy training behaviors provide a strong foundation for future work using the pipeline presented in this study. With faster inference times, experiments can immediately be conducted on a real dVRK setup to assess the performance of the diffusion-based policies in real-world surgical environments, particularly comparing its performance to the traditional domain randomization approaches. The experiments will provide the opportunity to understand the effectiveness of diffusion for bridging the sim-to-real gap, and aid in quantifying the amount of additional domain randomization needed to achieve similar or better task performance than with current approaches in literature. This study provides a novel foundation for answering these questions within future work over a variety of surgical tasks.

Chapter 6

Conclusion

One of the primary challenges associated with developing vision-based RL policies in simulation for surgical robotics is the large gap between the simulated and real-world environments. Generative models have been used in robotics literature to bridge this gap by generating realistic images from simulated environments, however, diffusion models have not been used in the context of RL, although having shown promising results in image generation tasks. This study fine-tuned three Stable Diffusion models on real data from different surgical tasks using Dreambooth, enabling the assignment of text prompts to represent specific task styles. Unity simulation images were used for inference with the diffusion models, providing high-fidelity realistic images similar to the real-world images of the tasks. While the sim-to-real gap is not completely closed, the results demonstrate the potential in using diffusion architectures for enhancing simulation environments, particularly for surgical robotics where tissue textures are complex.

The study also introduced an RL training pipeline that integrates fine-tuned diffusion models to generate realistic images from Unity simulation input observations. The results of the diffusion-based RL training experiments demonstrate that this pipeline can effectively facilitate the learning of policy action distributions from high-fidelity images, although the training speed is limited due to the time required for the diffusion denoising process. Training times for diffusion-based policies took up to 20 days to complete one million training steps, compared to just one day for PPO policies without diffusion inference. Due to this limitation, hyperparameter tuning was confined to baseline policies, and sim-to-real transfer experiments were not conducted with diffusion-based policies. However, the training results show that diffusion-based policies follow similar learning trends to baseline policies, indicating that they learn comparable task features despite the stochasticity of diffusion-generated images. These findings suggest that diffusion-based policies could converge to optimal performance with extended training time and proper hyperparameter tuning. Furthermore, the stochastic

nature of diffusion-generated images introduces variability that provides the policy with a more realistic representation of the surgical environment, aiding its ability to adapt to the noise and inconsistencies typical of real-world scenarios. This effect is reflected in the policy training losses and time series results, where policies trained with diffusion inference learn more nuanced task representations as a result of the image observation variability. Such variability may enhance generalization to real-world settings by encouraging the policy to develop more robust feature representations.

This study highlights the potential of diffusion models in bridging the sim-to-real gap, while also emphasizing the need for faster inference times to optimize this approach for real-world surgical tasks. With optimized inference times achieved through the use of works such as InstaFlow, experiments can be conducted on a real dVRK setup to assess the performance of the diffusion-based policies in real-world surgical environments. More generally, diffusion-based policies can be trained and evaluated directly using the modular pipeline presented in this study, enabling immediate experimentation across different tasks, hyperparameter settings, and domain randomization techniques in both simulated and real-world environments. Overall, the results of this study provide a novel foundation for future research in sim-to-real transfer using diffusion models, with the potential to significantly enhance the development of autonomous surgical systems. By leveraging the strengths of diffusion models, researchers can explore new avenues for improving policy generalization and transferability, ultimately leading to more effective and efficient robotic surgical systems.

Bibliography

[1] Attanasio, A., Scaglioni, B., De Momi, E., Fiorini, P., and Valdastri, P., "Autonomy in surgical robotics," *Annu. Rev. Control Robot. Auton. Syst.*, Vol. 4, No. 1, May 2021, pp. 651–679.

[2] Liu, N., Cai, Y., Lu, T., Wang, R., and Wang, S., "Real–Sim–real transfer for real-world robot control policy learning with deep reinforcement learning," *Appl. Sci. (Basel)*, Vol. 10, No. 5, Feb. 2020, pp. 1555.

[3] Haiderbhai, M., Gondokaryono, R., Wu, A., and Kahrs, L. A., "Sim2Real rope cutting with a surgical robot using vision-based reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, 2024, pp. 1–12.

[4] Haiderbhai, M., Gondokaryono, R., Looi, T., Drake, J. M., and Kahrs, L. A., "Robust Sim2Real transfer with the da Vinci research kit: A study on camera, lighting, and physics domain randomization," *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2022.

[5] Li, Y., Wu, Z., Zhao, H., Yang, T., Liu, Z., Shu, P., Sun, J., Parasuraman, R., and Liu, T., "ALDM-grasping: Diffusion-aided zero-shot Sim-to-Real transfer for robot grasping," March 2024.

[6] Rao, K., Harris, C., Irpan, A., Levine, S., Ibarz, J., and Khansari, M., "RL-CycleGAN: Reinforcement learning aware simulation-to-real," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2020.

[7] Scheikl, P. M., Tagliabue, E., Gyenes, B., Wagner, M., Dall'Alba, D., Fiorini, P., and Mathis-Ullrich, F., "Sim-to-real transfer for visual reinforcement learning of deformable object manipulation for robot-assisted surgery," *IEEE Robot. Autom. Lett.*, Vol. 8, No. 2, Feb. 2023, pp. 560–567.

[8] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2022.

[9] Valevski, D., Leviathan, Y., Arar, M., and Fruchter, S., "Diffusion models are real-time game engines," Aug. 2024.

[10] D'Ettorre, C., Mariani, A., Stilli, A., Rodriguez y Baena, F., Valdastri, P., Deguet, A., Kazanzides, P., Taylor, R. H., Fischer, G. S., DiMaio, S. P., Menciassi, A., and Stoyanov, D., "Accelerating Surgical Robotics Research: A Review of 10 Years With the da Vinci Research Kit," *IEEE Robotics & Automation Magazine*, Vol. 28, No. 4, 2021, pp. 56–78.

[11] Thananjeyan, B., Balakrishna, A., Nair, S., Luo, M., Srinivasan, K., Hwang, M., Gonzalez, J. E., Ibarz, J., Finn, C., and Goldberg, K., "Recovery RL: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, Vol. 6, No. 3, Jul 2021, pp. 4915–4922.

[12] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," *CoRR*, Vol. abs/1707.06347, 2017.

[13] Gondokaryono, R., Haiderbhai, M., Suryadevara, S. A., and Kahrs, L. A., "Learning Nonprehensile Dynamic Manipulation: Sim2real Vision-Based Policy With a Surgical Robot," *IEEE Robotics and Automation Letters*, Vol. 8, No. 10, 2023, pp. 6763–6770.

[14] Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., Levine, S., and Vanhoucke, V., "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE Press, 2018, p. 4243–4250.

[15] Chung, J., Hyun, S., and Heo, J.-P., "Style Injection in Diffusion: A Training-Free Approach for Adapting Large-Scale Diffusion Models for Style Transfer," *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 8795–8805.

[16] Kumar, A., Soni, S., Chauhan, S., Kaur, S., Sharma, R., Kalsi, P., Chauhan, R., and Birla, A., "Navigating the realm of generative models: GANs, diffusion, limitations, and future prospects—A review," *Proceedings of Fifth International Conference on Computing, Communications, and Cyber-Security*, Lecture notes in networks and systems, Springer Nature Singapore, Singapore, 2024, pp. 301–319.

[17] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S., "Deep unsupervised learning using nonequilibrium thermodynamics," *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, JMLR.org, 2015, p. 2256–2265.

[18] Kaleta, J., Dall'Alba, D., Płotka, S., and Korzeniowski, P., "Minimal data requirement for realistic endoscopic image generation with Stable Diffusion," *Int. J. Comput. Assist. Radiol. Surg.*, Vol. 19, No. 3, March 2024, pp. 531–539.

[19] Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., and Rombach, R., "Scaling rectified flow transformers for high-resolution image synthesis," *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, JMLR.org, 2024.

[20] Gupta, G., Yadav, K., Gal, Y., Batra, D., Kira, Z., Lu, C., and Rudner, T. G. J., "Pre-trained Text-to-Image Diffusion Models Are Versatile Representation Learners for Control," 2024.

[21] Ai, H. and Sheng, L., "Stable Diffusion Reference Only: Image Prompt and Blueprint Jointly Guided Multi-Condition Diffusion Model for Secondary Painting," 2023.

[22] Zhang, L., Rao, A., and Agrawala, M., "Adding Conditional Control to Text-to-Image Diffusion Models," *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 3813–3824.

[23] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W., "LoRA: Low-Rank Adaptation of Large Language Models," *International Conference on Learning Representations*, 2022.

[24] Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K., " DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation ," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, Los Alamitos, CA, USA, June 2023, pp. 22500–22510.

[25] Technologies, U., "Unity ML-Agents Toolkit — unity-technologies.github.io," `https://unity-technologies.github.io/ml-agents/`, [Accessed 11-01-2025].

[26] "DreamBooth — huggingface.co," `https://huggingface.co/docs/diffusers/training/dreambooth?gpu-select=16GB`, [Accessed 11-01-2025].

[27] Su, Z., Liu, W., Yu, Z., Hu, D., Liao, Q., Tian, Q., Pietikäinen, M., and Liu, L., "Pixel Difference Networks for Efficient Edge Detection," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5097–5107.

[28] Zhao, W., Bai, L., Rao, Y., Zhou, J., and Lu, J., "UniPC: a unified predictor-corrector framework for fast sampling of diffusion models," *Proceedings of the 37th International*

Conference on Neural Information Processing Systems, NIPS '23, Curran Associates Inc., Red Hook, NY, USA, 2023.

[29] "GitHub - facebookresearch/xformers: Hackable and optimized Transformers building blocks, supporting a composable construction. — github.com," `https://github.com/facebookresearch/xformers`, [Accessed 12-01-2025].

[30] Wang, Z., Farnia, F., Lin, Z., Shen, Y., and Yu, B., "On the Distributed Evaluation of Generative Models," 2024.

[31] Kwiatkowski, A., Towers, M., Terry, J. K., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M., Andreas, K., Krimmel, M., KG, A., Perez-Vicente, R. D. L., Pierré, A., Schulhoff, S. V., Tai, J. J., Tan, H., and Younis, O. G., "Gymnasium: A Standard Interface for Reinforcement Learning Environments," 2025.

[32] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N., "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, Vol. 22, No. 268, 2021, pp. 1–8.

[33] Liu, X., Zhang, X., Ma, J., Peng, J., and qiang liu, "InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation," 2024.