Model-based Control for Robot Manipulation Tasks with High-dimensional State Spaces

by

Bilha-Catherine "Bilkit" W. Githinji

B.S., University of Washington (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....
Department of Electrical Engineering and Computer Science
May 13, 2022

Certified by .....
Julie A. Shah
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Model-based Control for Robot Manipulation Tasks with High-dimensional State Spaces

by

Bilha-Catherine "Bilkit" W. Githinji

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Long horizon manipulation tasks are typically composed of sub-tasks with varying complexity. One phase of the task, for example, may require a continuous action space and another may be more efficiently solved using a discrete action space. Similarly, complexity in the state space may require analogous abstractions in order to apply classical planning and control methods; e.g., viewing a symbolic representation versus pixel-based representation. A common approach to addressing long horizon tasks is to develop a hierarchical system with a fixed state representation and a set of discrete and continuous action spaces to solve components of the task. However, tasks with high-dimensional state spaces present a problem for this approach where the fixed representation is ill-fit for solving certain phases of the task. This work motivates an alternative where learnt abstractions of the state space allow a hierarchical system to do coarse-to-fine reasoning of representation information to solve a task more effectively. We demonstrate a prototype of such an adaptive system and compare its performance with a system that has fixed representations. The prototype was tested in simulated table-top experiments as well as physical experiments with the Franka Emika Panda arm. The prototype outperformed the baselines in all long horizon cloth manipulation tasks by a margin of up to 20% and matched baseline performance in the rope domain.

Thesis Supervisor: Julie A. Shah
Title: Professor of Aeronautics and Astronautics

Acknowledgments

To my parents and brother who inspire me to not only dream big, but to pursue my dreams with a steadfast can-do spirit.

Thank you to Professor Julie Shah who graciously offered support and encouragement while I dove into the unknown. I'm indebted to my fellow lab mates at the Interactive Robotics Group. Thank you for entertaining my endless questions and for lifting my spirits during the pandemic.

I also owe thanks to John Schulman for early discussions and encouragement, and Chandler Squires and Ray Chen at the EECS Communications Lab for their kind feedback while writing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robot manipulation is a frontier of robotics research that has recently made rapid progress thanks to advances in machine learning and its application to sensing, perception and reinforcement learning. Research groups like Google Brain and OpenAI have stunned the world with systems capable of human-like dexerity, e.g., solving a Rubicks Cube while balancing it in a single robot hand [30]. The methods underlying these solutions incorporate with machine learning components that efficiently solve seemingly intractable manipulation problems. In particular, deep learning has enabled decision-making algorithms to operate directly over high-dimensional inputs, like pixel observations, and thus largely void the need to manually design state representations [22]. With increasing task complexity, however, researchers have pivoted towards learning low-dimensional inputs from images to increase the efficiency and generalisation capabilities of downstream planning and control frameworks. Determining what information should be summarised in the low-dimensional input is a non-trivial problem, and in cases where elements of the task vary (e.g., visual and physical variations), popular solutions that learn and fix representations are limited [14].

Many decision-making problems are formulated as Markov Decision Processes (MDPs), which define an interactive process between a robot and its environment. The robot's behaviour is determined by solving an MDP using Dynamic Program-

ming, Reinforcement Learning and other optimisation algorithms. This kind of formulation strongly depends on the notion of a state, a summarisation of task-relevant information about a robot and its environment.

1.1 The Notion of State

The problem of defining and estimating the state of the environment is at times posed as a representation learning problem, as frequently seen in computer vision. In many scenarios, the environment state is grounded in image observations. So, there is a natural dimensionality reduction problem; determining how to extract information from high-dimensional data into a succinct state description that characterises it, or, for our purposes, best aids task execution. As the environment becomes more complex, however, more variables are needed to capture an expressive summary, which instantiates a curse of dimensionality problem. That is, optimisation with respect to more complex state descriptions becomes more challenging, and often intractable. It is desireable then to learn state representations with as low dimensionality as is appropriate for solving a task. Many robotics problems involve systems whose state is fairly straightforward to describe in low dimensions, but let's consider the richer space of object manipulation.

1.2 Object Manipulation

Dealing with objects introduces a combinatorial aspect to manipulation problems and it contributes significantly to the curse of dimensionality. There is, virtually, an unbounded variety of object types, and within each class is a vast range of shapes and sizes, for example. Object manipulation usually require contact with the environment, so, depending on the task, a manually defined state description will encapsulate an object label or identifier, simple geometric parameters, a pose, etc. The number of dimensions of these representation spaces grow exponentially with increasing object multiplicity.

To simplify manipulation tasks, researchers often restrict the range of objects to a small set within the same class or adjacent classes. Another example, tasks like pick-and-place, where the robot transports an object from one location to another, often scope out complex object geometries to more directly address the grasping problem. One simplification that this work concerns is the strong assumption that researchers typically make about object rigidity.

Assuming object rigidity helps to reduce state descriptions to a small set of variables that convey key information, but is unreasonable for solving real-world object manipulation. While useful for addressing established research questions, these assumptions don't transfer well to questions that consider real-world settings where objects have high-dimensional state. In such domains manually deriving a concise state descriptions becomes a massive bottleneck. There is little intuition about how to identify relevant information in high dimensions and the standard hand-crafted representations used by many controllers don't necessarily have clear analogues in high-dimensions - e.g., the meaning of "pose" becomes elusive. Richer state descriptions are necessary for real-world manipulation tasks, but one paralysing obstacle is the lack of intuition about what information is relevant and how to quantify it in lower dimensions.

1.3 Motivation

Simple learnt representations are desired for integration with the optimisation algorithms underlying most controllers. They provide a means of automatically filtering task-relevant information from the environment observations. However, they come at the cost of expressivity. When addressing tasks with high-dimensional state spaces, this cost becomes untenable. It induces a task-conditioned representation selection problem, whereby an optimal representation is chosen from a set of candidates. If the controller were applied to a task that deviates from the conditional task, then the representation may be sub-optimal to a potentially detrimental extent. This scenario

commonly occurs in multi-stage tasks where popular methods use multiple controllers to execute parts of the task, or an adaptive controller that engages different modes depending on the task phase. In this work, we consider the latter approach and propose to view the variable modes as fluid representations.

1.4 Overview

This thesis proposes a novel approach to solving mult-stage tasks with controllers than optimise over adaptive representations. Our main contributions include the following:

- An analysis of the trade-offs between employing various representations for solving short-horizon and multi-stage tasks.
- An implementation of a proof of concept adaptive control framework that dynamically filters state information to tailor trajectory tracking solutions to local parts of the state space.
- A benchmark against baselines that demonstrates the task performance limitations of fixing the learnt representation for the duration of a long horizon task.
- A physical setup that testes the prototype in a real world setting.

Chapter 2 contextualises our proposed approach by drawing relations to representation learning, hierarchical control and adaptive control. It reviews various deformable object representations that were successfully used in classical planning and control and explains how integrating machine learning has enabled these approaches to address increasingly more complex manipulation tasks. Further, it highlights the limitations of using past methods in long horizon task settings.

A hybrid model-based controller with a mutable representation of the environment is described in Chapter 3 along with details of our problem formulation. Fundamen-

tally, our approach leverages techniques from representation learning to overcome the challenges of modelling high-dimensional systems and draws ideas from the divide-and-conquer paradigm to implement a coarse-to-fine strategy for long horizon trajectory tracking.

The proposed hybrid controller is evaluated in a suite of simulation experiments and benchmarks presented in Chapter 4. The benchmarks show that the hybrid controller outperforms baselines that exist in strong prior art. The chapter closes with interprets of the results in using visualisations and case studies. In preparation for real-world experiments, Chapter 5 provides a description of a physical system that reproduces the simulated experiments. Figure 5-6 lays out the architecture of a physical implementation of the system.

Chapter 6 concludes this report with suggested extensions of our approach to related deformable object manipulation problems and areas beyond robot manipulation where control optimisation is hindered by high-dimensionality of the state; e.g., doing manipulation with changing number of objects (assembly/disassembly), soft robotic control, and humanoid control.

Chapter 2

Background

The work in this thesis advances ideas from representation learning, hierarchical control and adaptive control. Sections 2.1-2.2 review prior works that leveraged low-dimensional representations to reduce state complexity for object manipulation tasks. Hierarchical approaches to long horizon control problems are addressed by Section 2.3.

2.1 Representations for Non-Rigid Objects

Previous works in object manipulation have developed low-dimensional representations for use in traditional planning and control frameworks to reduce the complexity of systems with large and high-dimensional state. Early work often optimised the representations to contain the minimal information required to complete a task. For instance, graphical and topological state descriptions were successfully used for planning complex rope knotting sequences [24, 55, 38]. These sparse representations contained information about the topology; e.g., the number and identity of crossings in the rope configuration. As a result, these representations were insufficient for tying functional knots (e.g., fastening a shoe lace). Manipulation tasks gradually increased in complexity and demanded motion planning and control solutions. There was a need for representations that took into account the dynamical properties of ropes, cloth and 3D deformable objects. Lower-level representations such as linear

and mesh mass-spring models were designed for surgical knotting and cloth folding tasks [51, 57, 50, 40]. Other representations include deformation models that were used by pick-and-place planners to solve quasi-static tasks; precise rope and cloth shaping, for example [10, 19]. Some even addressed dynamic tasks like high-speed knot tying, dynamic cloth unfolding and "in air" knotting [54, 11, 38]. While these representations offer higher expressivity, their fitness for a specific task hinges on careful parameter tuning - an impracticality for scaling to real-world domains.

Machine learning has played an increasingly important role in robotic control, especially in deformable object manipulation, to solve state estimation problems including system identification and state representation learning. Deisenroth et al. (2011) and Power & Berenson (2020) proposed probabilistic dynamics models that increased the efficiency of model-based policy search algorithms by steering them away from states with high uncertainty [6, 33], as estimated by Gaussian Processes or an ensemble of neural models. Simple mechanical models (the classical and inverted pendulum) were used in both cases to represent complex systems; e.g., a real world unicycle system and a ball on the end of a string. Integrating data-driven system identification with traditional state representations is a viable approach to control problems with complex systems. However, the same treatment does not translate well to systems in the deformable object domain where hand-crafted analogues typically yield high aleaotoric uncertainty. This thesis is partially motivated by limitations in this area.

2.2 Learning State Representations

Modelling the behaviour of deformable objects is particularly challenging because they are under-actuated systems with high dimensional state and non-linear dynamics. We take inspiration from state representation learning literature, in particular [22, 27, 56, 29], to develop the dynamics model described in Chapter 3. Supervised learning methods have been used to learn how to represent and estimate the dynamics

of rope and cloth. Contrastive learning, however, has provided an avenue for learning object-centric representations in a self-supervised manner [25]. Sundaresan et al. (2020) extended the technique in Manuelli et al. (2020) to learn task-specific geometric representations by incorporating supervision for untying dense knots [45, 9]. Their task-oriented representation was a key addition to established knot-theoretic planning algorithms [50, 24], but it relied on human supervision to identify the task-relevant features, which does not scale well to real-world applications. The core of these approaches was the idea of learning key points that summarised important information about grasping objects. The key points indicated parts of the object that were useful for a particular task; e.g., identifying the heel of a shoe as a graspable region. Yan et al. (2020) approached the representation learning problem from a different angle applying ideas from contrastive predictive coding to learn a low-dimensional state representation and non-linear dynamics jointly for rope and cloth manipulation [56, 29]. In this framework, latent features were optimised for maximising forward prediction accuracy for their Model Predictive Control (MPC) algorithm. This body of work showed that learning representations in an self-supervised manner was an highly effective approach to reducing high-dimensional state. However, there is little work that extends these approaches to long horizon object manipulation tasks in the real world. Long horizon, or more generally multi-stage tasks, can often be composed into a sequence sub-tasks. The individual sub-tasks may require a continuous action space or a discrete action space in order to complete a task more efficiently. For example, a table clearing requires discrete actions for selecting items to be moved and continuous actions for physically removing them. Similarly, complexity in the state space may require analogous abstractions in order to apply classical planning and control methods; e.g., viewing a symbolic representation versus pixel-based representation.

2.3 Multi-Stage (Long Horizon) Tasks

Typically a hierarchical approach is required to effectively and efficiently perform long horizon tasks. A long history of approaches have achieved impressive results on rope shaping and cloth folding tasks, some of which have a long horizon [5, 26, 51, 53, 56, 15]. One of the most common hierarchical approaches is to apply learning from demonstration to solve DOM tasks; i.e., to use human-guidance to scope the search space for motion planners and controllers. The key idea is to provide a either reference trajectory or a sequence of way-points (i.e., sub-goals) in the action space that a policy mimics, dramatically simplifying object modelling and cost function estimation [4, 1, 41]. The reference may also be grounded in the observation space, in which case a control algorithm tracks the trajectory [40, 39, 28]. One limiting factor of these approaches is that a controller may not be able to reach the sub-goals effectively if there is enough distribution shift such that the reference context is no longer relevant at execution time. Hence, it is of primary interest in this thesis to invest in object modelling and to leverage latent derivatives of the observation space to solve multi-stage tasks with robust demonstration tracking. Furthermore, we postulate that these latent representations should not be fixed over the duration of long horizon tasks. Inspired by divide-and-conquer paradigms computer vision that quickly reduce the problem search space, we propose an adaptive control framework that varies the low-dimensional state representation in a coarse-to-fine manner to benefit local tracking of a long horizon trajectory [36, 31, 58]. We find a similar approach in model-free RL that implemented a coarse-to-fine strategy to augment an RL agent that attends over pixel inputs via Q-attention [18, 17]. This prior work does not address deformable object manipulatoin.

An alternative to trajectory tracking is a recent innovation by Lin et al. (2020) called DiffSkill, a hierarchical approach to solving food preparation tasks using tools. This framework was developed to solve sequential manipulation tasks that involved shaping and cutting cooking dough with tools. The general idea was to learn action

primitives from a set of simulated demonstration trajectories. The action primitives, dubbed "skills", are a system of neural models that represent a policy, reward estimator, feasibility estimator and an image encoder. Similar to the proposed architecture in Chapter 3, the low-level system that executes the skills operates over a latent space generated by the encoder. To conduct a long horizon task, a planner optimised over the action primitives to produce a high-level action sequence: e.g., moving the dough to a target location, rolling out and cutting it in half. The differences between this work and our proposed method lie in the nature of the task and the architecture of the hierarchical approach. For example, degree of precision is higher in our proposed set of tasks, in that the interactions between the controller and the object are more nuanced; e.g., while both are challenging, creating a rectangular fold requires different reasoning than using tools to make large deformations of the object. Additionally, while DiffSkill decomposes the long horizon task at the action level, learning individual neural systems for action primitives, our approach is interested in decomposition at the representation level.

The following Chapter establishes control theoretic formulations for high-dimensional trajectory tracking and presents our unique approach which leverages ideas from prior work that reason about spatial information in a hierarchical manner.

Chapter 3

Hybrid Model-based Control

In light of the methods covered in Chapter 2, we suggest an alternative approach to solve non-rigid object manipulation tasks that uses time-varying representations of the state. In particular, we concern ourselves with goal-conditioned manipulation tasks that require multiple stages of low-level reasoning under guidance of a high-level policy; e.g., a plan generated by a human. One underlying problem in this domain is related to the difficulty of reasoning about high-dimensional state spaces, e.g., object deformation or decomposition, where guidance in the form of trajectory tracking is computationally arduous. We propose to address this problem using an adaptive control strategy that modulates the coarseness of information conveyed by a low dimensional analogue of the state to track trajectories. Along the same line of thinking as James et al. (2022) [17], who developed a novel Reinforcement Learning algorithm Coarse-to-fine Q-attention, we aim to develop model-based controllers that dynamically select the granularity of state information to address deformable object manipulation tasks that span multiple stages. Our approach consists of two key components: a learning framework for modelling non-rigid objects and a prototype for this novel adaptive controller.

This chapter introduces a formal problem description and details the proof of concept adaptive control scheme. It lays out the theoretical foundation for our hybrid control scheme and elaborates on how representation learning was incorporated into

this framework to address the problem of high-dimensional state estimation.

3.1 Preliminaries

Sequential decision making is a fundamental problem in robotics. It is most commonly formalised using Markov Decision Processes (MDPs) in planning and control. The formalisation depends on the notion of an environment, its state, actions, an agent, and an evaluation function (i.e., reward). Together these elements frame the decision making problem of learning to achieve a goal from interacting with the environment [46]. In the control regime, the analogous elements are: a system, its state $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{d_x}$, control inputs $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^{d_u}$, a controller $\pi : \mathcal{X} \rightarrow \mathcal{U}$, and a cost function $c(\mathbf{x}) \in \mathcal{C} \subset \mathbb{R}$.

A key feature of MDPs is the Markovian property which makes it possible to model interactions between the controller and the environment, i.e., the dynamics. The Markov property enforces a strong constraint on the state definition - that a state must convey all past information necessary to recover information about future states, thus ensuring that the system is memoryless.

Provided the Markovian state assumption, it is possible to define forward transition functions to relate sequential states by control inputs. Let $X \in \mathcal{X}_t$ and $U \in \mathcal{U}_t$ be random variables representing the environment state and the control input. The dynamics then are given by:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \Pr\{X_{t+1} = x_{t+1}|X_t = x_t, U_t = u_t\}, \tag{3.1}$$

or in a deterministic view, $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$. It is important to note the two schools of thought at the core of many MDP solutions that are based on the burden or utility of dynamics estimation. Those dubbed the "model-free" approaches avoid the estimation problem entirely. On the other hand, "model-based" approaches allocate

significant effort to characterise the system dynamics in order to restrict the optimisation search space. Both sides make trade-offs that are worthwhile in some contexts, but prohibitive in others. The high-dimensional manipulation problem we considered in this work would benefit more from a model-based approach, as described in the following sections.

3.2 Model-based Control

Optimal control is an instance of an MDP that is widely used to formulate optimisation problems in robotics where a sequence of control inputs must be determined to optimise an objective. The dynamics are a useful tool that constrains the optimisation algorithm to evaluate future states in a manner that is consistent with the system's behaviour. They typically arise, for example, from the equations of motion for a simplified mechanical system that represents the robot and its environment. As previously mentioned, the system dynamics are fundamental to the model-based optimal control problem.

A generalised form of the model-based optimal control problem (OCP) involves solving for a set of decision variables that minimise a prescribed cost objective [35]. Typically, the decision variables are a state and control input trajectory. They must also adhere to a set of constraints that ensure that the optimal trajectory is feasible. More precisely, let f be the system dynamics function, $\mathbf{x}_0$ and $\mathcal{X}_{final}$ be an initial state and terminal state set, ℓ and V be the instantaneous cost function and terminal cost, $\mathbb{Z}$ be a set of admissible trajectories, and N be a finite time horizon. In the discrete time setting, the goal of model-based optimal control is to find a feasible trajectory $\tau_{x,u} = ((\mathbf{x}[0], \mathbf{u}[0]), \ldots, (\mathbf{x}[N-1], \mathbf{u}[N-1]), (\mathbf{x}[N], \mathbf{0}))$ that satisfies the

following optimisation problem:

$$\min_{\tau_{x,u}} \sum_{t=0}^{N-1} \ell(\tau_{x,u}[t]) + V_{\text{final}}(\tau_{x,u}[N]) \tag{3.2}$$

$$\text{subject to} \tag{3.3}$$

$$\mathbf{x}[0] = \mathbf{x}_0 \tag{3.4}$$

$$\mathbf{x}[t+1] = f(\tau_{x,u}) \tag{3.5}$$

$$\mathbf{x}[N] \in \mathcal{X}_{final} \tag{3.6}$$

$$\tau_{x,u}[t] \in \mathbb{Z}, \forall t \tag{3.7}$$

A cost minimising trajectory must satisfy the constraints 3.4-3.6. Equations 3.4 and 3.6 state that the state trajectory at time $t = 0$ must begin with the initial state x_0 and at time $t = N$ it must terminate at a state that is contained in the pre-defined set of final states. Additionally, all points in the trajectory $(\mathbf{x}[0], \mathbf{u}[0])$ must be contained in a pre-defined set of admissible decision variables $\mathbb{Z}$ (Equation 3.7). Finally, transitions between states along the trajectory must be explained by the system dynamics (Equation 3.5). Due to the tightly regulated solutions to Equation 3.2, satisfying these hard constraints may be intractable in problems with large and complex state spaces. The next section presents another view of optimal control that simplifies this optimal control problem.

3.3 Model Predictive Control

A variant of this problem reduces the decision variable set to $\tau_u = (\mathbf{u}[0]), \ldots, \mathbf{u}[N-1])$ by embedding the dynamics function constraint directly into the objective. The state trajectory can be computed by a function Φ provided f, $\mathbf{x}_0$ and τ_u. The result

is a sequential optimal control problem:

$$\min_{\tau_u} \sum_{t=0}^{N-1} \ell(\Phi(k; \mathbf{x}_0, \tau_u) + V_{\text{final}}(\Phi(N; \mathbf{x}_0, \tau_u)) \tag{3.8}$$

$$\text{subject to} \tag{3.9}$$

$$\mathbf{x}[0] = \mathbf{x}_0 \tag{3.10}$$

$$(\Phi(t; \mathbf{x}_0, \tau_u), \mathbf{u}[t]) \in \mathbb{Z}, \forall t \tag{3.11}$$

$$\Phi(N; \mathbf{x}_0, \tau_u) \in \mathcal{X}_{final} \tag{3.12}$$

The constraints in this problem are analogous to those in the OCP setting (Equation 3.2). The main difference being that the intermediate states and the final state are computed by the forward simulation function Φ, which implicitly enforces the dynamics constraint. As a result, the optimisation is done over only the control input trajectory τ_u.

The Sequential Optimal Control Problem (Seq-OCP) is a more appropriate formulation than OCP in situations that are characteristic of our application. Namely, if there lacks sparsity in the set of optimisation variables. Consider a classic deformable object manipulation task; folding a piece of cloth. The cloth may be modelled as a set of particles connected with springs or a mesh. The state may consist of the positions and velocities, as well as orientation, of those particles. The optimisation problem in Equation 3.2 would require solving for the myriad elements in the state, which, depending on the resolution of the mesh, would be intractable. There are, however, simplified formulations that we can leverage for problems where the state space is large and high-dimensional.

Real-world contexts are laden with non-linear system dynamics that require complex models to characterise them reliably. OCP and Seq-OCP rely on exact characterisations of the system dynamics in order to converge. Potential errors introduced by poorly modelled dynamics can make these problems impossible to solve. Such

uncertainties can occur when a model class that lacks sufficient representation power (aleotoric uncertainty) or there is insufficient data to estimate the dynamics with a reasonable model (epistemic uncertainty). For this reason, it is convenient to treat optimal control as a two-stage problems: solve a state estimation problem then apply its solution to optimise OCP. We directly apply this paradigm to our approach in Section 3.5.

3.4 Problem Formulation

Consider a set of multi-stage table-top manipulation tasks deformable objects. Each task requires the objects to be arranged into specific configurations. An oracle (e.g., a human expert) will provide an M-length sequence of visual sub-goals $s_g^M = \{o_g^i\}_{i=[1:M]}$ that eventually reach the target arrangement (see Figure 3-1). The visual sub-goals are birds-eye view observations of the scene taken at intermediate stages of task execution. The sub-goals do not explicitly contain any information about the oracle's actions. Given an initial observation o^0 and $s_g^M = \{o_g^i\}_{i=[1:M]}$, an end-effector controller must optimise a trajectory of control inputs to manipulate the objects in scene from their initial configuration to sequentially match the configurations observed in s_g^M.

More formally, let $\pi(\mathbf{u}_t|o_t)$ be a controller that takes as input image observations $o_t \in \mathcal{O}$ and outputs control inputs $\mathbf{u}_t \in \mathcal{U} \subset \mathrm{SE}(3)$. The controller optimises control inputs $\tau_u = \{\mathbf{u}_t\}_{t=[0:N-1]}$ to sequentially shape the objects in the scene to match those in S_g^M until the goal configuration is reached. The controller is evaluated according to a goal-conditioned cost $\ell(o_t, s_g^M)$ over finite time horizon N.

For brevity, the variable notation in subsequent sections assumes time dependence; i.e., $o = o_t$, for example. Prime notation is used as short-hand for the subsequent time step; i.e., $o' = o_{t+1}$.

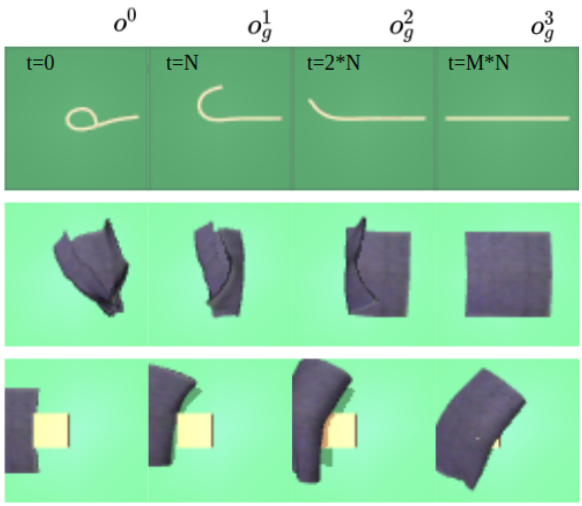


Figure 3-1: Examples of multi-stage tasks. The examples shown are for rope shaping, cloth unfolding and covering an object with a piece of fabric. Given an initial observation and a sequence of sub-goals, a controller must manipulate the objects from their initial configuration to match each stage in the sequence until the goal is reached.

3.5 A Hybrid Control Approach

The problem in Section 3.4 corresponds to learning a sequence of paired primitive actions that satisfies a task specification prescribed by a sequence of sub-goals drawn from an oracle (see Chapter 4). Since the search space over states and actions is not sparse the OCP underlying the proposed problem is extremely challenging to optimise. This hindrance motivates our choice to use a model predictive control algorithm to solve instances of Seq-OCPs to track the observed sub-goal sequence s_g^M. Consequently, it was necessary to define and estimate the state and dynamics.

Machine learning is a critical tool for estimating the high-dimensional state induced by the non-rigid objects and as well as characterising the under-actuated and non-linear dynamics [55]. We rely on neural models to generate a low-dimensional state representation and to compute a dynamics function that coincided with observed transitions. Additionally, we hypothesise that using a single low-dimensional representation would limit tracking performance as the non-rigid object underwent drastic deformation and various contact modes. So, we propose a simple hybrid control ar-

chitecture with components taken from Yan et al. (2020) [56] that would emulate an adaptive controller with a dynamic feature set.

3.5.1 Observation and action space

As explained in Section 3.6, self-supervised representation learning to derive a low-dimensional state representation and forward dynamics estimator. Two models, discussed in the following section, are jointly optimised to yield these components. The system of models take as input observations of the scene $o \in \mathbb{R}^{W \times H \times c}$ with dimensions width, height and channel depth W, H and c. They encode the observations into embeddings $z \in \mathcal{Z} \subset \mathbb{R}^k, k \in \mathbb{N}$. The space $\mathcal{Z}$ represents a mutable state representation. Optimal control cost was evaluated entirely in $\mathcal{Z}$. As part of pre-processing, the pixels in o were normalised and transformed to the range $[-1, 1]$. These transformations result in model inputs that were consistent and zero-centered, a desirable property for model training.

The hybrid controller generated actions in the normalised pixel space. The actions consisted of a 3D pick pixel and pull direction, $\mathbf{u} = [u, v, d, \Delta v, \Delta u, \Delta d]^T \in \mathbb{R}^6$, where u, v, and d are pixel values along each dimension of the image. The components of $\mathbf{u}$ were defined in the range $[-1, 1]$ to correspond to the pixel locations in o. They were used to compute 3D pick-and-place actions $a^{\text{pick}}(\mathbf{u}) = \mathcal{T}(\mathbf{u}_{(u,v)}, P)$, where $\mathcal{T}$ defines a mapping from the observation space to world coordinate frame using camera calibration matrix P. The component d depends on selection of u, v. The place actions were computed as $a^{\text{place}}(\mathbf{u}) = a^{\text{pick}}(\mathbf{u}) + \gamma \mathbf{u}_{(\Delta v, \Delta u, \Delta d)}$, where γ is a scalar gain in the world coordinate frame.

3.5.2 Controller Design

Multi-stage tasks are composed of multiple phases of low-level control that can be formulated as Seq-OCPs (Equation 3.8). With deformable object manipulation tasks

in mind (see Chapter 4), we propose a hybrid model-based controller that tracks the sub-goal sequence s_g^M where each stage is formulated as an Seq-OCP with a low-dimensional state representation. The controller modulates the number of features used to represent state information in order to reach and stabilise around each sub-goal. Let k denote the dimensionality of the embedding space generated by a pre-trained model and $\mathcal{K} \subset \mathbb{N}$ be the set of feature sizes corresponding to each model. For example, $\mathcal{K}$ may contain dimensions $\{2^i\}_{i=[3:12]}$, for image inputs with $W = 64, H = 64$. A set of pre-trained models $\{h_\theta^k : \mathcal{O} \to \mathcal{Z} | \mathcal{Z} \subset \mathbb{R}^k, k \in \mathcal{K}\}$ are used to instantiate a set of low-level controllers $\{\pi_k : \mathcal{O} \to \mathcal{U} | k \in \mathcal{K}\}$. To refer to the components of the model independently, we use the model parameters ψ and ϕ to index into the encoder $h_\psi^k = g_\psi^k$ and forward model $h_\phi^k = f_\phi^k$. The observation sequence s_g^M is mapped by the pre-trained models to a set of latent sub-goals $\{z_g^{k,M}\}$. To simplify notation, the superscript k will be implicit; i.e., the embeddings $z_g^k, z^k \in \mathbb{R}^k$ will be denoted by $z_g, z \in \mathbb{R}^k$ instead.

To emulate a controller with an adaptive representation, the hybrid controller implicitly optimises the control input trajectory by invoking a controller drawn from $\{\pi_k\}$ to drive the state towards individual sub-goals $z_g^i \in z_g^M$, for $i = 1, \ldots, M$. It optimises the following objective with free choice over a control mode trajectory $\lambda = \{k_{i,t}\}_{t=[0:N-1], i=[1:M]}, k_{i,t} \in \mathcal{K} \subset \mathbb{N}$ which determines which controller is active at each time step:

$$\min_{\lambda} \sum_{i=1}^{M} \left[c(z_0, z_g^i) + \sum_{t=1}^{N-1} \ell(z_t, \pi_{\lambda_{i,t}}(z_t), z_g^i) \right] \tag{3.13}$$

$$\text{subject to} \tag{3.14}$$

$$z_{t=0} = z_0 \tag{3.15}$$

$$(\Phi(j; z_0, \pi_\lambda), \pi_{\lambda_j}) \in \mathbb{Z}, \forall j = [0, \ldots, MN-1] \tag{3.16}$$

$$\Phi(j; z_0, \pi_\lambda) \in z_g^M, \forall j = i * N, \tag{3.17}$$

where $\Phi(\cdot; z_0, \pi_\lambda)$ computes a sequence of forward predictions $\{h_\phi(z_n, \pi_{\lambda_n}(z_n))\}_{n=[0:t]}$, j is a sequence index and π_λ is the control input trajectory from .

The hybrid controller executes a policy selection scheme (Algorithm 1) to invoke a low-level controller according to a condition on the latent state trajectory; e.g., state trajectory's convergence $G_{\epsilon\text{-conv}}(z_t, z_{t+1})$ or its proximity to the sub-goal $G_{\epsilon\text{-prox}}(z_t, z_g^i)$:

$$G(z, z') = \begin{cases} 1, & \text{if } \|z - z'\|_2^2 \leq \epsilon \\ 0, & \text{otherwise.} \end{cases} \tag{3.18}$$

The controller carries out a coarse-to-fine strategy where the state space is represented with a small number of dimensions when it is far from the goal and becomes more refined near the goal. In lines 5 and 7, the model with the lowest dimensionality is used to generate coarse state representations of the goal and current observation. The policy selection criterion G is then evaluated with respect to the coarse representations. Generally, the mode selection mechanism, indicated by the function SelectMode (Line 8), is a switch over models $\{h_\theta^k\}$. Suppose there are three choices $k_{\min} \leq k_1 < k_2 \in \mathcal{K}$. An example may be a binary selector that uses $G_{\epsilon\text{-prox}}$ to select between a control models k_1, k_2 depending on whether the trajectory is within ϵ distance of the goal. An n-step MPC routine runs model-based optimal control using model $h_\theta^{\lambda_t}$, denoted from now on as $h_{\psi,\phi}^{\lambda_t}$ with ψ parameters of the encoder model and ϕ parameters of the forward model.

The n-step MPC procedure optimises a control input trajectory to satisfy the objective $\min_{\tau_u} \sum_{t=1}^{n} \|h_\phi(z_t, \tau_{u,t}) - z_g\|_2^2$. It uses h_ψ to encode observations and uses Algorithm 2 to optimise this objective. Following methods from Yan et al. (2020), we optimise highly non-convex objective using Cross-Entropy Method (CEM) to avoid local optima. In line 2, the controller samples a batch of randomly inputs $\hat{\mathbf{u}} \sim U_{[-1,1]^6}$ and evaluates them to find the cost minimising input with respect to

Algorithm 1 Hybrid Control

procedure RUN(s_g^M, $\{\mathrm{h}_{\psi,\phi}^k\}$)
  Initialise $h_\psi^{k_{\min}} \in \{h_{\psi,\phi}^k\}$ with $k_{\min} = \min(\mathcal{K})$ ▷ Define coarsest model
  Define $n \in [1, \ldots, N]$ ▷ Set MPC horizon
  for $i = [1, \ldots, M]$ **do**
    $z_g^i \leftarrow h_\psi^{k_{\min}}(o_g^i)$ ▷ Coarse goal state
    while $t < N$ **do**
      $z_t \leftarrow h_\psi^{k_{\min}}(o_t)$ ▷ Coarse observed state
      $\lambda_t \leftarrow \text{SelectMode}(G_{\epsilon\text{-prox}}(z_t, z_g^i)) \in \mathcal{K}$
      $\tau_u \leftarrow \tau_u \cup \text{RunMPC}(o_t, o_g^i, h_\theta^{\lambda_t}, n)$ ▷ Optimise control inputs
      $t \leftarrow t + n$
    end while
  end for
  return τ_u
end procedure

Algorithm 2 n-Step MPC

procedure RUNMPC(o_t, o_g^i, $\{h_{\psi,\phi}^k\}$, n)
  Initialise empty U^* ▷ n-step solution
  Define $\ell(z, \mathbf{u}, z_g) := \|h_\phi(z, \mathbf{u}) - z_g\|_2^2$
  Define SampleBatch(S) := $\{\hat{\mathbf{u}} \sim U_{[-1,1]^6}\}^S$
  $z_0 \leftarrow h_\psi^k(o_0), z_g^i \leftarrow h_\psi^k(o_g^i)$ ▷ Estimate state
  for $t = [0, \ldots, n-1]$ **do**
    $\hat{U} \leftarrow \{\{0\}^6\} \bigcup \text{SampleBatch(S=1000)}$ ▷ Sample candidate inputs
    $\mathbf{u}_t = \arg\min_{\mathbf{u}_t \in \hat{U}} \ell(z_t, \mathbf{u}, z_g)$ ▷ Select cost minimising input
    $U^* \leftarrow U^* \cup \mathbf{u}_t$ ▷ Update solution
    $z_t \leftarrow h_\phi(z_t, \tau_{u,t})$ ▷ Update forward prediction
  end for
  return U^*
end procedure

$\ell(z, \mathbf{u}, z_g) = \|h_\phi(z_t, \tau_{u,t}) - z_g\|_2^2$. This process repeats for each step along the horizon n. In addition to the low-level control in Algorithm 1, the baseline controllers in our experiments (Chapter 4) implemented this algorithm with fixed k and horizon $n = 1$.

3.6 Dynamics Estimation

Recent work in representation learning has demonstrated that neural models are a powerful tools for extracting useful features from large datasets for application in complex tasks like image classification and language parsing. State representation learning is concerned with extracting features that convey task-relevant information about the environment. Many robotics applications have scarce training corpora, let alone labelled data, so self-supervised representation learning is highly desireable. Contrastive learning offers methods that depend less on human supervision than other forms of representation learning. These offerings make it well suited for state estimation problems where representations with high utility are unintuitive and difficult to hand-craft.

3.6.1 Model Training

The work proposed by Yan et al. (2020) [56] gave inspiration for the model architectures and training setup that we used to learn a state representations and a dynamics estimator. This paper applied contrastive learning theory in Oord et al. (2018) [29] to embed environment observations in such a way that preserved mutual information between the latent space and environment state. The key idea was to use temporal distance between images as a learning signal to encourage similar observations to be clustered in the latent space while dissimilar observations were dispersed. The contrastive learning objective they used applied a constraint on the latent space which made it possible to traverse in a way that was consistent with the observed system dynamics.

Specifically, we applied their framework to train a model h_θ with components $g_\psi : \mathcal{O} \to \mathcal{Z}$ an image encoder and $f_\phi : \mathcal{Z} \times \mathcal{U} \to \mathcal{Z}$ a forward dynamics estimator. We relied on a simulated environment to generate a dynamics data by randomly perturbing objects in the environment and recording trajectories the tuple $\{(\mathbf{u}, o, o', \{\tilde{o}\}^n)\}^L$ at each timestep, where L was the trajectory length, o was the observed image, o' was the subsequent image after applying perturbation $\mathbf{u} \sim \mathcal{N}(0, I)$, and $\{\tilde{o}\}^n$ were n examples of incorrect subsequent images. Let $\mu : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ be a similarity measure between states in the embedding space. The joint model h_θ was trained using the following contrastive loss which seeks to localise latent states based on their relative distance according to measure μ:

$$\mathcal{L}_{\text{contrastive}} := -\mathbb{E}_{(o,o',\{\tilde{o}^n\},\mathbf{u})\sim\mathcal{D}} \left[\log \frac{\mu(z^+, z')}{\sum_{i=1}^{n} \mu(z^+, \tilde{z}_i)} \right], \tag{3.19}$$

where $z^+ = g_\psi(o')$ is the embedding of the next observation, $z' = h_\theta(o, \mathbf{u})$ is the model's forward prediction and $\tilde{z}_i = g_\psi(\tilde{o}_i)$ is the embedding of an incorrect example of the next observation. Given disjoint batches of trajectories $b_x, b_y \sim \mathcal{D}$, the observations o, o^+ were drawn from b_x and negative examples $\tilde{z}_i$ were either drawn from b_y. As a result, minimising $\mathcal{L}_{\text{contrastive}}$ encouraged the models to learn a state representation and transition function such that the control input trajectory τ_u (see Section 3.1) yields a latent trajectory $\{z_{t+1} = h_\theta(o_t, \mathbf{u}_t) | \mathbf{u}_t \in \tau_u, t = 0, \ldots, L\}$ that has high mutual information with the observed trajectory $\{o_{t+1} = f(o_t, \mathbf{u}_t) | \mathbf{u}_t \in \tau_u, t = 0, \ldots, L\}$.

3.6.2 Model Architecture

The model architecture we used for the system h_θ was taken directly from Yan et al. (2020) [56]. The image encoder g_ψ was a seven-layer convolutional neural network that reduced observations $o \in \mathbb{R}^{W \times H \times c}$ to embeddings $z \in \mathbb{R}^k$. It was composed of an input layer and six layers of convolutional filters partitioned by Leaky ReLU activations. The dynamics estimator f_ϕ was a three-layer feed-forward neural network with one input layer and two hidden layers. An embedding $g_\psi(o) \to z$ and a control input

u were concatenated to generate the forward model's input $y \in \mathbb{R}^{k+6}$ which computed the outcome of the control input as embedding $z' \in \mathbb{R}^k$. Though fairly lightweight compared to state-of-the-art representation learning models (visual transformers, and the like), these model architectures sufficiently exhibited good training performance and were capable of modelling test systems well enough to do optimal control.

3.6.3 Augmentations

Minor modifications that we made to the original model included adding a depth channel and a separately trained decoder model. The depth channel was an important addition that enriched the learnt representations with geometric information. Such information was necessary to distinguish between observations where the object was heavily self-occluded, e.g., in the case of cloth folding. Additionally, we train a decoder model $g^{-1}_{\psi_{\text{dec}}} : \mathcal{Z} \to \mathcal{O}$ alongside h_θ as an evaluation tool. As described later in Section **??**, this inverse map made it possible to evaluate latent predictions across different models using metrics in a common space. The model was optimised according to the reconstruction loss:

$$\mathcal{L}_{\text{recon}} := \mathbb{E}_{(o,o',\{\bar{o}^n\},\mathbf{u})\sim\mathcal{D}} \left[\sum_n \sum_{u,v} (g^{-1}_{\psi_{\text{dec}}}(z)_{u,v} - o_{u,v})^2 \right], \tag{3.20}$$

where $u \in \{0, \ldots, H-1\}$ and $u \in \{0, \ldots, V-1\}$ are pixel indices and $n \in \{0, \ldots, |\mathcal{D}|-1\}$is a sample index. Again, note that $\mathcal{L}_{\text{recon}}$ did not make contributions to optimising Equation (3.6.3).

E This chapter reviewed the problem on which this thesis centers and introduced our approach to solving it. It grounded the standard mathematical formulation for model-based optimal control in the problem of multi-stage deformable object manipulation. It also detailed the architecture of our proposed solution. The next chapter presents results on extensive simulation experiments that tested our proposed state estimation and optimal control framework.

Chapter 4

Experiments & Analysis

The control design described in Chapter 3 was implemented and tested in a collection of simulated object manipulation tasks. We sought to understand the trade-offs between the representations generated by model variations like changing the similarity measure, dimensionality of the embedding space, and training parameters. We studied these factors in experiments involving short horizon tasks. The findings motivated a key design choice for the adaptive control prototype: to change the dimensionality of the controller's representation in an online fashion. The hybrid control design was analysed in a set of multi-stage task experiments. Sections 4.1-4.2 establish the data collection pipeline, task environments, our experimental setup and evaluation metrics. Subsequent sections discuss the simulated experiments that were directed at the following questions:

(Section 4.3) What kinds of variation in the state representation yield dynamics models with low prediction error?

(Section 4.4) How do different state representations affect the quality of optimal control solutions on simple object reconfiguration tasks?

(Section 4.4) What strategies are useful for the adaptive hybrid controller?

(Section 4.5) For tasks with multiple objects, how does the quality of optimal control solutions found with adaptive state representations compare with those found using a fixed representation?

These experiments reveal that the state space dimensionality is an effective way to modulate the granularity of information conveyed by the state representation. We find that increasing the dimensionality of the latent space improves the quality of optimal control solutions up to a surprisingly low limit, after which performance on object reconfiguration tasks drops drastically. Furthermore, the results suggest that, when using a fixed representation, the quality of optimal control solutions depends on the state space region; e.g., whether a rope is flat or coiled. These outcomes affirm our hypothesis that fixed representations are sub-optimal when compared with adaptive representations. Additional experiments measure task performance of the hybrid controller against that of baseline controllers with fixed representations. In most cases, the hybrid controller outperforms the baselines.

4.1 Training Data Collection

Under the assumption that large temporal distance corresponds to high object displacement and deformation, it was possible to automatically generate a rich dataset for training the models in Section 3.6. The data generation process required a simulation environment with realistic physics and tuneable physical parameters. Given such a tool, we simulated interactions between a virtual end-effector and the scene by randomly sampling pick-and-place actions and recording the relevant data needed to optimise Equation 3.6.3.

Training the joint model h_θ reliably depended on access to a large dataset of interactions between a controller and the system. We designed the data collection pipeline in environments derived from the Deep Mind Control Suite (DM Control Suite) [48, 49] task environments in the same manner as Yan et al. (2020). The control suite was integrated with MuJoCo 2.0, a high-fidelity physics engine, that produced realistic behaviours for both contact rich settings and environments with

deformable objects like rope and fabric. Though the MuJoCo framework could incorporate robot models into the simulation environment, we chose to represented the robot's end-effector with a mass-less point that could exert forces on the scene; the motivation being that the trained models would more likely generalise across robot platforms. Section 5.2 elaborates on how the model h_θ are applied when the physical robot's end-effector is present in the scene.

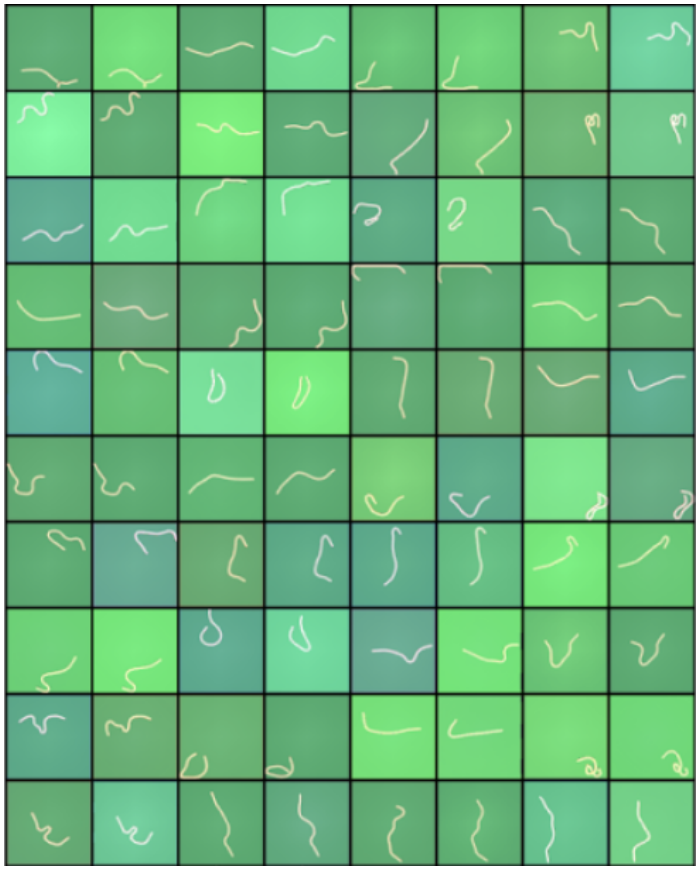


Figure 4-1: Rope Domain: Samples drawn from the a rope dynamics dataset. Each sample consists of an observation o, the next observation o' and a set of negative examples drawn randomly from other rollouts. Domain randomisation, in the form of lighting, mass, inertia and friction variation, was applied to increase model robustness to these variations during inference.

The models were trained on data generated in task environments from the DM Control Suite. Each environment provided access to observations that included the physics engine's full state, color images and depth images. The full state for non-rigid

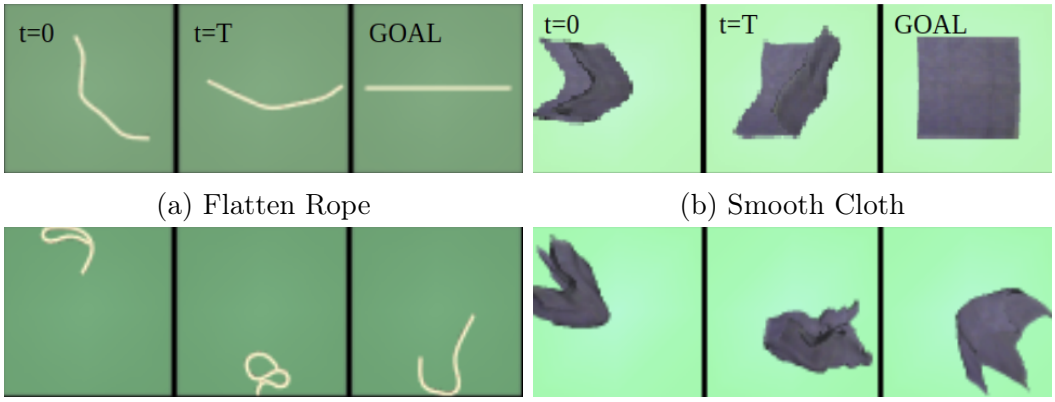
objects was given by mesh representations similar to those introduced in Section 2.1; e.g., a piece of fabric was defined by an $n \times m$ grid of particles and joints. A dynamics dataset $\mathcal{D}$ of 200k samples was collected by taking the union of r rollouts generated by a random policy $\pi_{\text{rand}} := \mathbf{u} \sim [\text{Uniform}(-1, 1)]^6$. This policy simulated the control input trajectory that would be produced by a physical robot's end-effector. The policy ran in a randomly initialised environment for L timesteps. The full state information was used to apply forces according to the output of π_{rand} and to adjust the physical properties of the scene elements. This pipeline made it possible to quickly generate data from many scenes with a range of physical variations including lighting, object mass, friction and inertia.

Once the rollouts were gathered, a pre-processing pipeline reorganised the trajectories into training samples as described in Section 3.6. The data points consisted of the observations o, the random policy's output $\mathbf{u}$ and the consequential observations o' for $i = \{1, ..., n\}$, where n is the dataset size. We initialised rollouts by loading a scene and applying a random perturbation on the deformable object; specifically, selecting two points and applying a large force in z. We then shuffled and partitioned the data into subsets $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ with a 0.8 split ratio (Figure 4-1).

4.2 Experiment Setup

The control design introduced in Section 3.5.2 was tested on rope and cloth manipulation tasks. Inspiration for our experiment design came largely from prior work [15, 28, 41, 43] which proposed human-guided control frameworks. Each experiment was based in a task environment that resembled the data collection environments. The experiment setup generally consisted of a table-top scene with a solid background and a set of objects (see Figure 4-2). Observations were captured from an overhead image sensor that provided c-channel images observations $o \in \mathbb{R}^{64 \times 64 \times c}$. The control input trajectory τ_u generated by each controller was scaled up to the simulated sensor's native resolution and small forces were applied iteratively to the scene elements ac-

cording to the 3D pull direction. The experiments involving rope manipulation gave controllers a horizon of 20 timesteps and those based on cloth manipulation gave 40 timestep horizon to complete the task.



(a) Flatten Rope (b) Smooth Cloth

(c) Random Rope (d) Random Cloth

Figure 4-2: Example observations taken from short horizon tasks in Shape Rope and Shape Cloth environments. Three example tasks are shown for each task type. The middle column (e.g., marked with "t=T") are examples of the final configuration a test policy achieved. Each task has a pre-defined initial object configuration and goal configuration. The "Flatten" rope and "Smooth" cloth tasks have fixed goal configurations across all trials. "Random" object tasks have randomised initial configuration and randomised goal configurations.

4.2.1 Short Horizon Task Suite

An initial set of experiments were required to validate model performance on short-range tasks. Simple reconfiguration tasks were an adequate setting to analyse the effects of varying model hyper-parameters on the quality of control input trajectories. They involved shaping a single target object into a pre-specified goal configuration. The target configurations were achievable in the allotted horizon. The goal of these tasks was to provide a setting for model evaluation (see Section 4.3).

An important assumption made in Algorithm 2 is that sampling space is actually restricted to pixels belonging to the object; i.e., those in an object segmented frame. Otherwise, the control input sampling scheme may cause the controller's performance

to degrade if the object size relative to the scene background is small. The probability of selecting a pick point from the background could be disproportionately high. For this reason, we chose to aid the action sampling scheme by providing a mask segmentation from which to sample - especially since we are concerned with testing model performance, which is independent of the action sampling scheme. Aside from this important assumption, all of the following experiments follow suite from work in Nair et al. (2017), Sundaresan et al. (2020) and Yan et al. (2020) [28, 45, 56].

Shape Rope Domain: Flatten Rope

This task was the simplest out of the short-range tasks. The scene was initialised with a random rope configuration. However, the same "flat" goal configuration was used for all trials. This provided a controlled setting for comparing different controllers. A more stringent controlled setting called "Flatten Rope-X" used fixed initial and goal configurations for qualitative comparison of executed trajectories (see Figure 4-7). A control horizon $T = 20$ was used for all rope shaping tasks (Figure 4-2a).

Shape Rope Domain: Random Rope

The initial configuration and goal configuration were both randomised in this task. In some cases, the goal specification required looping the rope and in others the rope was positioned at the boundary of the field of view. These tested the models' ability to handle, quite literally, the corner cases. It exemplified the more difficult and general case of rope shaping (Figure 5-4a).

Shape Cloth Domain: Smooth Cloth

Similar to the Flatten Rope task, the initial configuration was randomised, but the goal configuration was held across trials. A "Smooth Cloth-X" task was also available for qualitative comparison of executed trajectories. A control horizon $T = 40$ was used for all rope shaping tasks (Figure 4-2b).

Shape Cloth Domain: Random Cloth

This task was the most challenging for model evaluation. Often the worst-case difference, as measured using the full state (see Section 4.1), between the initial and goal cloth configuration was highest. (Figure 4-2b).

4.2.2 Tracking Task Suite

Hybrid controllers were evaluated on multi-stage task experiments. Each task tested the quality of trajectory tracking with respect to the metrics described in Section 4.2.4. The fixed-representation controllers were implemented by a hybrid controller that optimised Equation (3.13) with a singleton decision variable set. Hybrid controllers were given a set of two low-level controllers π_{k_1}, π_{k_2}, where k_i denotes different latent space dimensions. Both types of controllers were given a sub-goal sequence s_g^M that was generated using Algorithm 3. The sub-goal control horizon, i.e., the time allotted to reach a sub-goal, was 20 time steps for rope tasks and 40 for cloth tasks.

(a) Rope Unloop

(b) Cloth Unfold I

(c) Cloth Unfold II

(d) Cloth Fold

Figure 4-3: Example reference trajectories generated an oracle for simple tracking tasks.

Rope Unloop

We used this task to conduct low tier tracking experiments that tested whether the controllers could achieve non-trivial intermediate shapes to reach a goal configuration. Given a $M = 4$ length demonstration sequence, the goal of the task was to shape the rope to match each configuration in the sequence. We relied on this task for qualitative analysis as visual inspection of the alignment between the manipulated object and the intermediate configurations was more viable than in the rest of the suite.

Cloth Unfold I

Inspired by classic multi-stage cloth manipulation tasks [52, 41, 15, 42], the goal of this task was to unfold a piece of fabric that had been folded three times over. Each stage of this task requires a coordinated sequence of actions to align the cloth into a particular folded, or unfolded, configuration. For example, alternating between pulling a subset of the cloth's "corners" to the left and another subset downward. Furthermore, the intermediate configurations involve heavy self-occlusion.

Cloth Unfold II

In this task, another sequence of precise actions are required to unfold a piece of fabric whose "corners" been folded towards the center. High self-occlusion was the primary feature of this task, as seen in the second frame. In addition to a specific action sequence to achieve folds, as seen in Cloth Unfold I, this task required reasoning about the fold sequence itself. In the second frame, for example, the probability of successfully completing the task depends on whether the left fold is unfolded versus the right fold.

Cloth Fold

Similar to Cloth Unfold I and II, a precise action sequence is required to configure the cloth into a fold.

4.2.3 Oracle Policy

Sub-goal sequences s_g^M (Figure 4-3) for each task were generated by an oracle policy. The oracle was given an image observation of the scene where all the objects

are in the target configuration. It then applied a sequence of actions to rearrange the objects to the initial configuration specified in the task. The sequence was highly structured in the sense the actions were optimised to minimise the effort required to reshape the target configuration into the initial configuration, much like what a human would do in the real world. For example, for the Rope Unloop task, the environment was initialised in the "flat" state and the oracle applies a small set of actions to the flattened rope to form a looped shape.

Algorithm 3 Oracle Controller

procedure RUN(M)
  $r_g^M \leftarrow \{o_g^M\}$ ▷ Initialise with final goal
  for $i = [1, \ldots, M]$ **do** ▷ For M sub-goals
    for $j = [1, \ldots, L]$ **do** ▷ For horizon L
      $\mathbf{x}_t \leftarrow \text{FindKeyPoints}(o_t)$ ▷ Detect key points
      $\mathbf{u}_t \leftarrow \text{TaskStateMachine}(i * j, \mathbf{x}_t)$ ▷ Apply action
    end for
    $r_g^M \leftarrow r_g^M \cup \{o_t\}$ ▷ Store current observation
  end for
  $s_g^M \leftarrow \text{Reverse}(\{r_g^M\})$ ▷ Reverse the sequence
  return s_g^M
end procedure

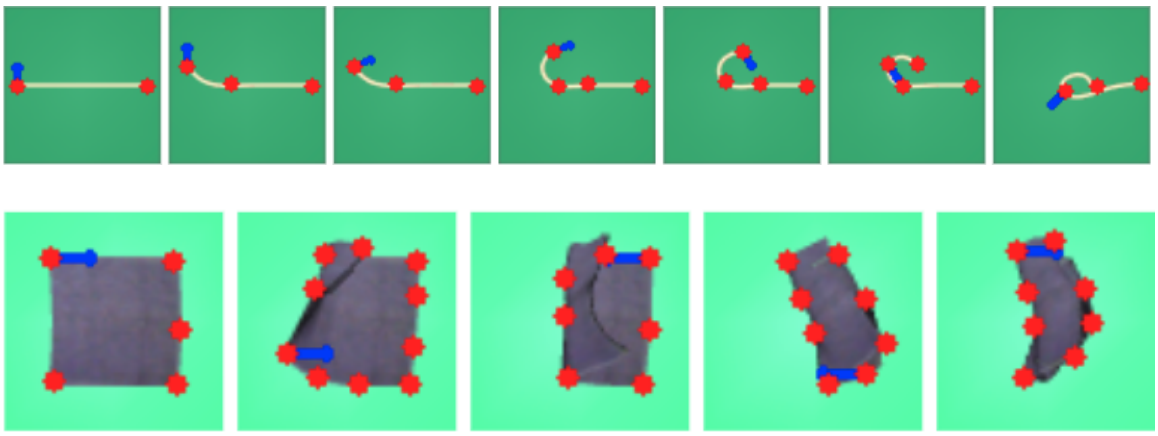


Figure 4-4: Annotated demonstrations for Rope Unloop and Cloth Fold that were generated by the oracle policy in Algorithm 3. The red points correspond to keypoints that are defined as part of the task state machine. Given an the key points and an index, the state machine outputs **u** indicated by a blue arrow.

To achieve the structured action trajectory, the oracle computed a set of key

points for each object and runs a state machine to generate an action (see Algorithm 3). Figure 4-4 illustrates the key points given as input to the oracle, e.g., the (red) endpoints for the rope and (red) corner points for the cloth. The oracle executed a state machine that yielded control inputs (blue arrows) that shape the object into the desired initial configuration with high probability - despite selecting actions with unknown object dynamics. As an example of the TaskStateMachine (Line **??**) for Cloth Fold, the oracle would alternate between picking the top left corner and the bottom left corner, and apply a large force to the right to each corner until a single fold is completed. Target corners (e.g., "top left" corner) were identified by their distance to the corners of the frame. The oracle policy was able to generate a wide range of demonstrations such as looping a rope, folding fabric and stacking objects.

4.2.4 Performance Metrics

The contrastive loss and reconstruction loss (see Section 3.6) described in the training scheme were useful indicators of model quality. A low contrastive loss showed that the model h_θ preserved, to a greater degree, mutual information between observations and their embeddings than a model with higher contrastive loss. In other words, the state predictions made by model were more consistent with the observed transitions. Reconstruction loss, on the other hand, provided an indication of the amount of information lost in the dimensionality reduction carried out by g_ψ.

In addition to the training task performance, our models were evaluated based on their efficacy in the optimisation a control input trajectory for solving the tasks in Section 4.2. All metrics were computed in the pixel space in order to compare metrics consistently across latent trajectories residing in different spaces. We assumed high contrast between foreground objects and the background to simplify object segmentation for constraining the controller's action sample space as well as computing coverage metrics.

Namely, we used the Jaccard distance, otherwise known as Intersection-over-Union

(IoU), to measure similarity between image observations:

$$\text{IoU}(o_1, o_2) = \frac{\sum_{u,v}(m(o_1) \cap m(o_2))_{u,v}}{\sum_{u,v}(m(o_1) \cup m(o_2))_{u,v}} \tag{4.1}$$

, where $m(\cdot)$ is a function that returns a binary masked segmentation of the object in o. One important note: computing distances at the pixel level proved difficult for tasks with objects that have a small footprint. For example, "thin objects" were disproportionately penalised in terms of IoU if their configuration satisfied the target shape, but was not perfectly aligned with the goal configuration. Yan et al. (2020) [56] reported the intersection of masked pixels instead with the assumption that the object shape (or size) remained consistent. Despite these nuances of the IOU metric, we chose to use this standard measure.

We also used Mean Squared Error (MSE) to measure reconstruction error and prediction error:

$$\text{MSE}(o_1, o_2) = \sum_{u,v}(o_{1,u,v} - o_{2,u,v})^2. \tag{4.2}$$

These metrics provided an approximate gauge for information lost in the encoding process. More importantly, they provided a means for comparing modeling errors. It was difficult, for example, to judge whether the prediction error measured in, say, 8 dimensional state space was actually lower than prediction error measured in a 64 dimesnional state space (see Figure 4-6). Prediction error measured in the pixel space (via MSE) was more aligned with reconstruction error than when it was measured in the latent space.

Lastly, the trajectory's proximity to the goal, as measured by L2-distance in the state space $\mathcal{Z}$, is used as a discussion point when we study the effects of increasing k on task performance (e.g., Figure 4-5). Though not as prudential as the aforementioned metrics, it was a useful point of comparison for understanding the behaviour of the latent trajectory.

4.3 Model Evaluation

Our initial experiments evaluated task performance of models with variations, ranging from sensor input, latent space dimensionality, similarity metric, extended training epochs, etc., to find a mode of variation that would be conducive to the hybrid control system. We used tasks from the short horizon Task Suite as a benchmark (Section 4.2). The models were trained from scratch in each of these environments and evaluated on the corresponding reconfiguration task. Each model was initialised with three different random seeds and trained with the similarity measure $\mu = \exp(-\|z - z'\|_2^2)$ (squared-exponential) for 30k training steps, by default. Note that for model architectures with $c = 3$ inputs, the control inputs $\mathbf{u}$ were restricted to the table-top surface by setting the d and Δd components to zero.

Table 4.1: Raw goal coverage (IoU) at time $t = N$ on simple reconfiguration tasks of single controller with various model types defined by the set of hyper-parameters in column 1. The first row presents our reproduction of the method of Yan et al. (2020), which did not report IoU. Subsequent rows report results produced with the following variations: c number of input channels, k-dimensionality of Z, training epochs, μ similarity measure (e.g., using the dot produce). The mean coverage and standard deviation were computed over 33 trials. Models marked with * were trained with three different random seeds for sample size=99.

	$\overline{\text{IoU}}(\sigma)$	
Hyper-Parameters	Flatten Rope	Smooth Cloth
Yan et al. (2020)	0.129 (0.064)	0.586 (0.068)
k=16, c=3	0.093 (0.068)	0.678 (0.051)
k=16, c=1	0.078 (0.047)	0.749 (0.069)
k=8, c=4*	0.124 (0.108)	0.614 (0.127)
k=16, c=4*	0.107 (0.084)	0.631 (0.133)
k=32, c=4*	0.075 (0.054)	0.636 (0.120)
k=64, c=4*	0.061 (0.075)	0.642 (0.116)
k=256, c=4*	-	0.603 (0.127)

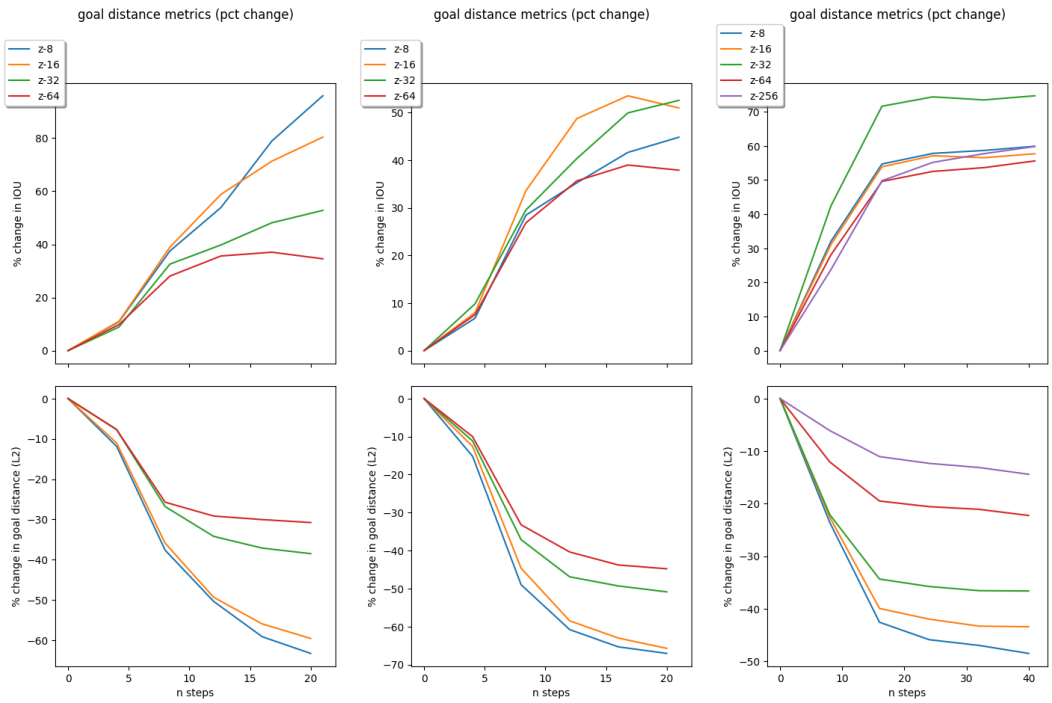
A collection of parameter sweeps are summarised in Tables 4.1 and 4.2, which list the raw performance as final goal-coverage (IoU) averaged over 33 trials for the

Flatten Rope and Smooth Cloth tasks. The average coverage and standard deviation were reported for the original model parameters in Yan et al. (2020) followed by variations on the input channels c, dimensionality of the latent space k, training time, and similarity measure μ (specifically, the dot produce). The input channels $c = 1, 3, 4$ correspond to Depth-only, RGB and RGB-D sensor input. Note that results marked with * were aggregated over three random training seeds for a total of 99 trials. We found that varying sensor input to include depth information as input increased performance in the both domains. Depth was a more useful signal than texture alone for distinguishing configurations where the object was self-occluded. However, we observed that RGB-D yielded the best performance in the rope setting which was not as serverely hindered by self-occlusions. Interestingly, in the Flatten Rope task there was up to a 50% performance drop off with 4x increase in k. The trend was less evident in the Smooth Cloth task where final performance floated around a tight range of 60-64%. To mitigate the performance drop for high k we attempted to train the models with dot-product similarity in the contrastive loss (see Table 4.2). Neither modification yielded better results than varying k.

Table 4.2: Raw goal coverage (IoU) at time $t = N$ on rope reconfiguration tasks of single controller with various model types defined by the set of hyper-parameters in column 1.

	$\overline{\text{IoU}}(\sigma)$
Hyper-Parameters	Flatten Rope
k=32, c=4, epochs=60	0.060 (0.045)
k=64, c=4, epochs=60	0.053 (0.042)
k=8, c=4, $\mu = z^T z'$	0.039 (0.027)
k=64, c=4, $\mu = z^T z'$	0.053 (0.038)

Recall that the short horizon test suites use random initial configurations. To make comparisons between the controllers more consistent, we report the % change in IoU and latent goal distance in Figure 4-5 for Random Rope and Random Cloth tasks. In this view, the highest performing models used $k = 16$ and $k = 32$ for the rope and cloth tasks, respectively. Figure 4-5a shows higher performance from models



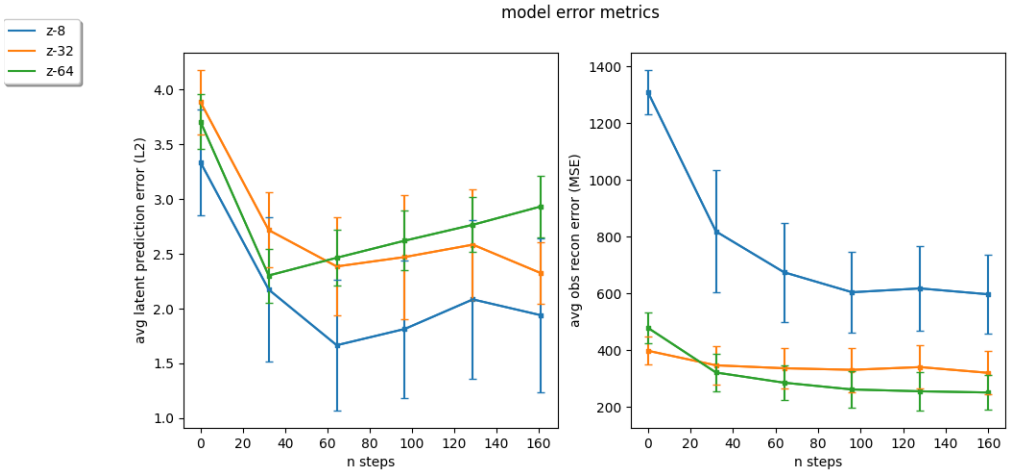
(a) Flatten Rope (b) Random Rope (c) Random Cloth

Figure 4-5: Shape Task: Goal coverage (% change in IoU) (top) and % change in latent goal L2-distance (bottom) over N time steps for controllers using model with $k \in \{8, 16, 32, 64, 256\}$. The upper bounds $k = 64$ for $k = 256$ represent the extreme case where the model would resemble a visual forward model; i.e., the encoder does no compression. Notice that there was a significant drop off in performance as measured by the latent L2-distance for both domains. The goal coverage metric indicated that the best performing models used $k = 16$ for rope and $k = 32$ for cloth tasks.

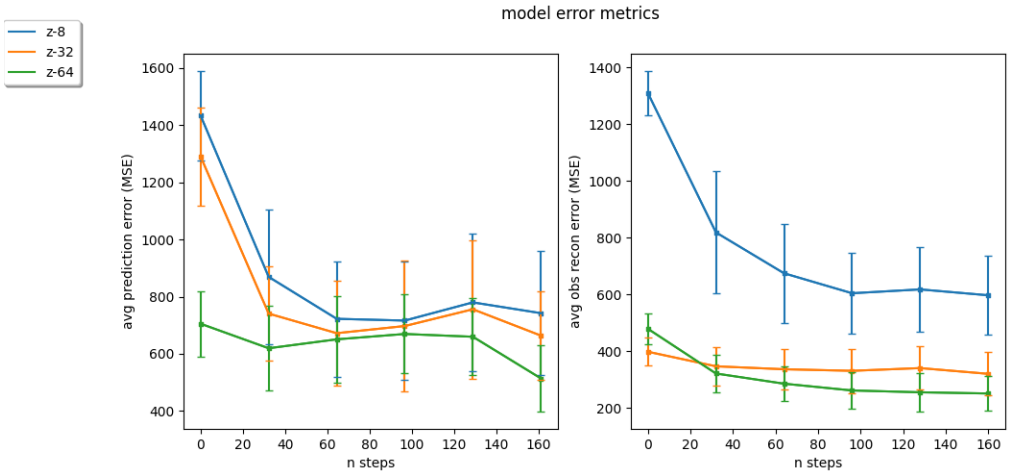
trained with $k = 8$ for the Flatten Rope task. An emerging trend was that lowering k yielded higher performance.

In general, increasing the k-dimensionality of the latent space was detrimental to optimal control in the Shape Rope setting and, to a lesser extent, the Shape Cloth setting. Surprisingly, there was an unexpected drop off in performance for models that exceeded $k = 32$ in both domains. Though reconstruction error (see Figure 4-6) decreased with increasing k, it did not always yield decreasing prediction error. Potentially, increasing k may have inadvertently widened the information bottleneck enforced by the contrastive loss in Equation 3.6.3, allowing the encoder to inject task-irrelevant features into $\mathcal{Z}^k$. For the rope shaping tasks, there was little tolerance for

widening the bottleneck. For the cloth shaping tasks, increasing k to 32 was admissible before a large performance decrease (as measured by L2-distance to the latent goal).



(a) Prediction Error (L2-distance) and Reconstruction Error (MSE)



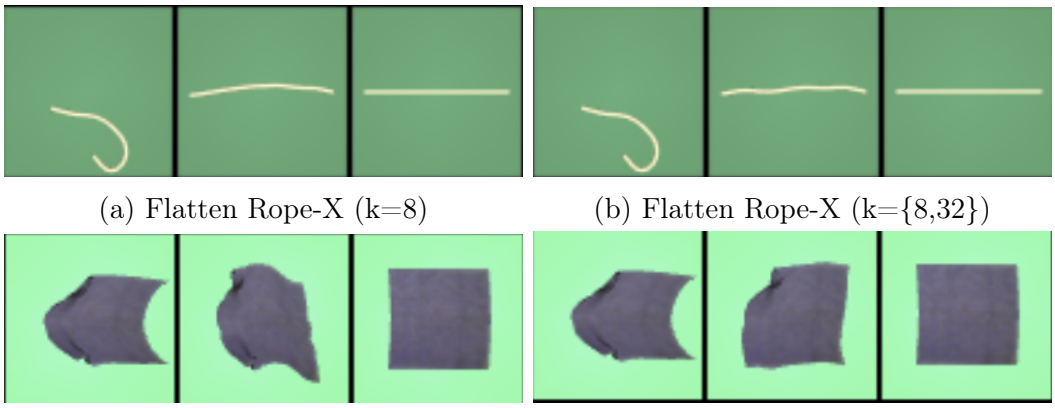
(b) Prediction Error (MSE) and Reconstruction Error (MSE)

Figure 4-6: Cloth Unfold I: Latent prediction error measured in $\mathcal{Z}^k$ (top), latent prediction error measured in O (bottom), and reconstruction error for baseline controllers equipped models trained with $k \in \{8, 32, 64\}$.

The primary insight from these initial results was that varying the dimensionality of the latent space, for the specific models introduced in Section 3.6.2, is an effective way to vary the representation to increase the efficacy of optimal control. Another key insight is that one cannot merely increase k, even within a low range of values,

to increase optimal control performance. This finding coincides with prior work in Section 2.2 that emphasised the importance of learning simple representations over highly expressive ones. Furthermore, the common model selection strategy for selecting and fixing k for all tasks may not be a reliable way to address the tracking problem. The Flatten Rope task demonstrated that reducing k for shaping into flat configurations was better than fixing it at the value prescribed by model selection, $k = 16$. This observation suggests that the efficacy of the fixed representations varies depending on the object configuration. Therefore, we chose to define variants of the models used in the hybrid control system with respect to k dimensions of the latent space.

4.4 Controller Evaluation: Short Horizon Setting

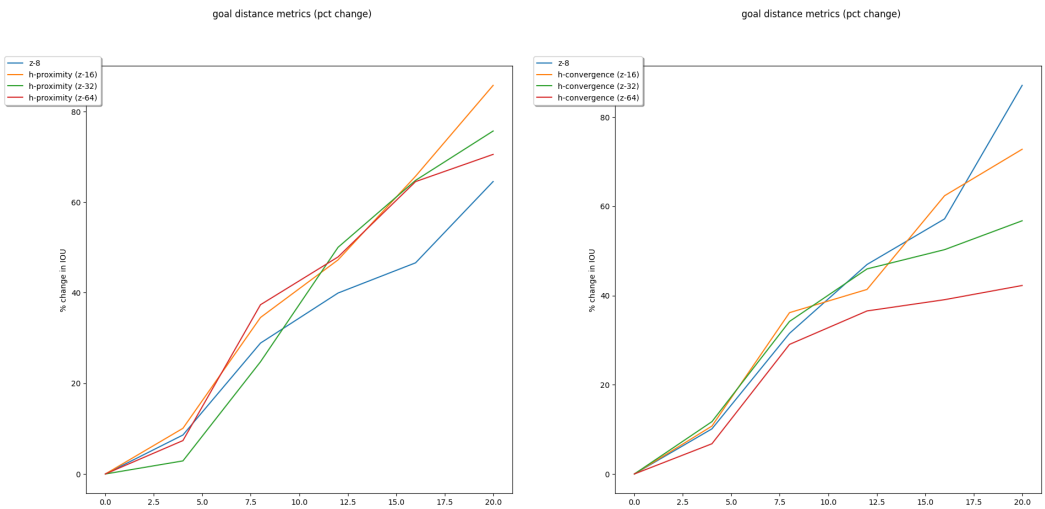


(a) Flatten Rope-X (k=8) (b) Flatten Rope-X (k={8,32})

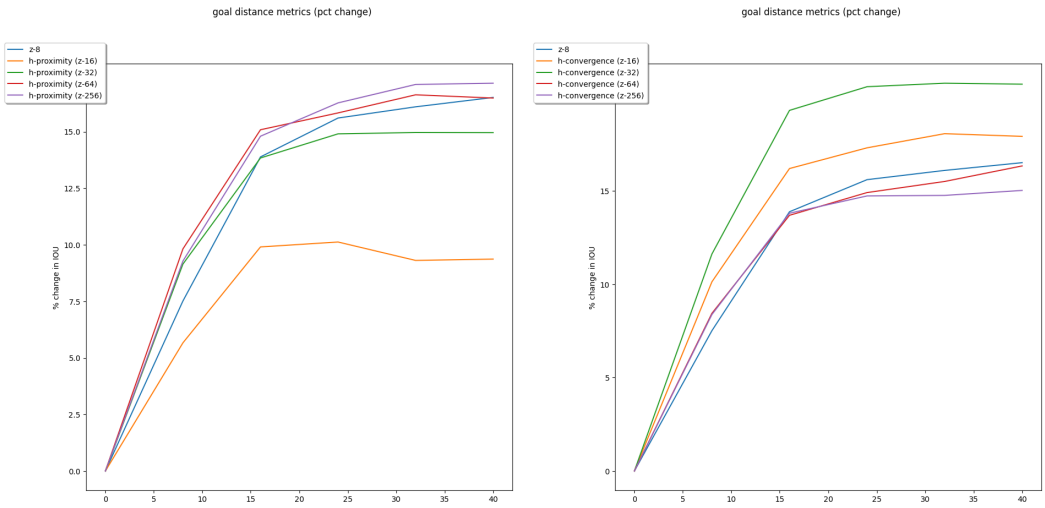
(c) Smooth Cloth-X (k=32) (d) Smooth Cloth-X (k={8,32})

Figure 4-7: Object Shaping Tasks: A comparison of object shaping trajectories generated by the best baseline controllers and hybrid controllers in the rope and cloth shaping task suites.

The hybrid controller presented in Algorithm 1 was implemented with two pre-trained models $\{h^{k_1}, h^{k_2}\}$, where $k_1, k_2 \in \{2^i\}_{i=[3:6]}$. We used both the criteria $G_{\epsilon\text{-prox}}$ and $G_{\epsilon\text{-conv}}$ (Section 3.5) to optimise the high-level output of the controller. The first round of experiments were carried out in the simplest setting; Flatten Rope and Smooth cloth. There, we set $j = 8$ and instantiated hybrid controllers $\pi(z; G, \{h^{j=8}, h^k\})$ with $k \in \{16, 32, 64\}$. Across all experiments $\epsilon = 8$ for the goal-



(a) Flatten Rope (Proximity) (b) Flatten Rope (Convergence)



(c) Smooth Cloth (Proximity) (d) Smooth Cloth (Convergence)

Figure 4-8: Shaping Task: Goal coverage (% change in IoU) over N time steps for baseline (blue) and hybrid controllers on Flatten Rope (top) and Smooth Cloth (bottom). Hybrid controllers outperform the baseline controllers if the appropriate policy switching criterion is used. In the case of cloth, the convergence criterion yielded nearly a 5% gain on cloth smoothing. However, the controller using the proximity criterion made a 20% gain on the baseline.

proximity criterion; otherwise $\epsilon = 1$. Table 4.3, Figure 4-7 and Figure 4-8 contrast performance of hybrid controllers $\pi(z; \cdot, \cdot)$ with that of baseline controllers $\bar{\pi}(z; h^k)$. The baselines were equipped with the best models, $h^{k=8}$ for the rope and $h^{k=32}$ for cloth, according to model selection (in the Flatten/Smooth tasks). Due to storage constraints, the results in the plots were computed over 33 out of 1000 uniformly

sampled evaluation trials. They showed that $G_{\epsilon\text{-prox}}$ was the superior strategy (by about 20%) for rope tasks, while $G_{\epsilon\text{-conv}}$ outperformed the baseline by 5% on the cloth smoothing. The implications of these results encouraged further experimentation in the multi-stage task setting. We expected the hybrid controller to reason about the different sub-goals in a similar manner as what was observed in these short horizon experiments.

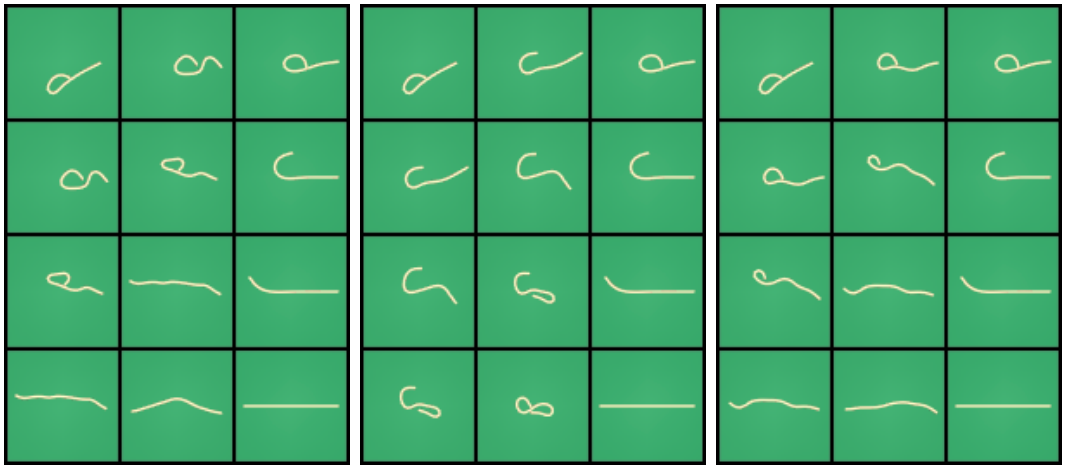
Table 4.3: Raw goal coverage (IoU) at time $t = N$ on short horizon tasks for the baseline $\bar{\pi}(z; h^8)$ and hybrid controllers $\pi(z; G_{\epsilon-(\cdot)}; \{h^{k_1}, h^{k_2}\})$ which varied by k_2 and the criterion $G_{\epsilon-(\cdot)}$. Note that these metrics were computed over randomly initialised object configurations. Results summarisied in Figure 4-8 show that the hybrid controller outperformed the baseline $\pi(z; h^8)$ in three out of four settings.

	$\overline{\text{IoU}}(\sigma)$	
Controller Type	Flatten Rope	Smooth Cloth
$\pi(z; h^8)$	0.124 (0.108)	0.614 (0.127)
$\pi(z; G_{8\text{-prox}}, \{h^8, h^{16}\})$	0.109 (0.079)	0.591 (0.129)
$\pi(z; G_{8\text{-prox}}, \{h^8, h^{32}\})$	0.116 (0.094)	0.601 (0.118)
$\pi(z; G_{8\text{-prox}}, \{h^8, h^{64}\})$	0.097 (0.085)	0.605 (0.120)
$\pi(z; G_{1\text{-conv}}, \{h^8, h^{16}\})$	0.111 (0.087)	0.604 (0.110)
$\pi(z; G_{1\text{-conv}}, \{h^8, h^{32}\})$	0.079 (0.056)	0.611 (0.113)
$\pi(z; G_{1\text{-conv}}, \{h^8, h^{64}\})$	0.067 (0.043)	0.627 (0.112)

4.5 Controller Evaluation: Tracking Setting

Experiments in the trajectory tracking suite suggested that the coarse-to-fine strategy carried out by the hybrid controller was more effective in the cloth domain and as effective in the rope domain than using the most performant baseline controllers (Table 4.3). We tested two hybrid controllers with varying $k \in \{8, 32, 64\}$, with $k = 8$ being the dimensionality of a coarse representation, and fixed the action selection criterion to G_{4-prox}. Their performance was measured on the Rope Unloop and Cloth Unfold I tasks against the baselines $\{\bar{\pi}_1(z; h^8), \bar{\pi}_2(z; h^{32})\}$, the highest performing controllers in the Flatten Rope and Smooth Cloth tasks. In both multi-stage

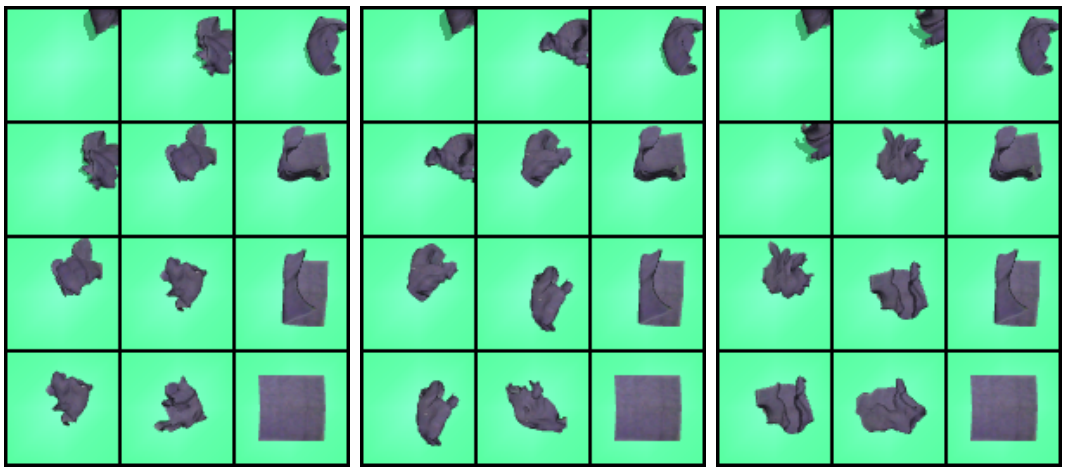
tasks, the initial configuration was sufficiently far from the goal (i.e., with IoU=0 at t=0). And though similar disparities existed in the short horizon setting, the multi-stage task introduced the further challenge of stabilising about each sub-goal to achieve the final goal configuration. We also benchmarked the controllers in the Cloth Unfold II and Cloth Fold tasks. The level of difficulty in these settings was greater because there was self-occlusion in the intermediate configurations and alignment with the goal configuration required specific action sequences (see Section 4.2).



(a) $\bar{\pi}_1$ (b) $\bar{\pi}_2$ (c) $\pi(z; G_{4-prox}; \{h^8, h^{32}\})$

Figure 4-9: Rope Unloop: Comparison of trajectory execution for (a) $\bar{\pi}_1(z; h^8)$, (b) $\bar{\pi}_2(z; h^{32})$ and (c) $\pi(z; G_{4-prox}; \{h^8, h^{32}\})$. A four-stage demonstration (see column three) was given as input to the controllers. At the first stage in the demonstration, the controller was given an initial frame (column one) and executed a trajectory resulting the final frame shown in column two. The next phase of trajectory was initialised using the final frame from the previous stage. The hybrid controller (right) successfully achieved the final "flat" configuration (row four) by leveraging a coarse-to-fine representation to stabilise about each intermediate goal. The baselines, on the other hand, either failed to reach the intermediate goals or were unable to stabilise around them.

Visual comparison between the executed trajectories indicated that the hybrid controllers were able to trade-off the utility of different representations to achieve the goal. Figures 4-9 and 4-10 show qualitative tracking results for $\bar{\pi}_1$ (left), $\bar{\pi}_2$ (middle) and the hybrid controller $\pi(z; G_{\text{4-prox}}, \{h^8, h^{32}\})$ (right). Each sub-figure corresponds to a full trajectory, where the task stages span the rows. The columns indicate the



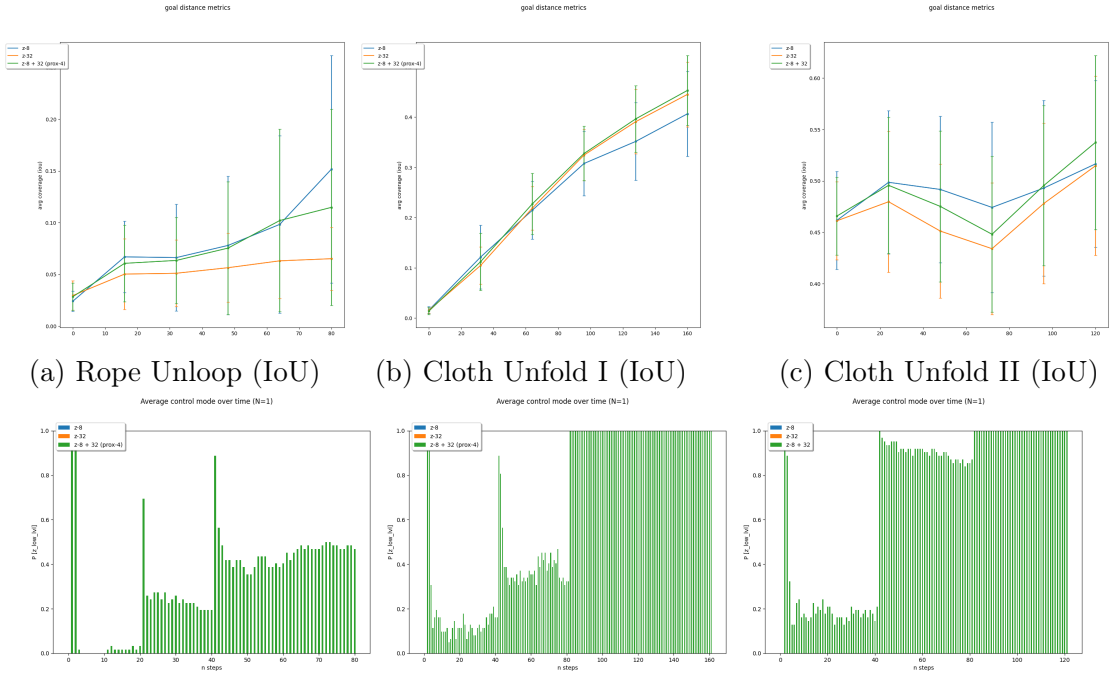
(a) $\bar{\pi}_1$ (b) $\bar{\pi}_2$ (c) $\pi(z; G_{4-prox}; \{h^8, h^{32}\})$

Figure 4-10: Cloth Unfold I: Comparison of trajectory tracking (a) $\bar{\pi}_1(z; h^8)$, (b) $\bar{\pi}_2(z; h^{32})$ and (c) $\pi(z; G_{4-prox}; \{h^8, h^{32}\})$. Rows indicate each stage in the demonstration trajectory (column three). The initial and final configurations during execution are shown in columns one and two, respectively. The baseline controllers were able to reposition the cloth. Only one baseline was able to smooth the cloth, but not as effectively as the hybrid controller which was able to use a coarse representation to position the fabric and attend to lower level features to smooth the cloth in later stages.

initial, final and goal configuration at each stage of the task. In the Rope Unloop task, the baseline controller $\bar{\pi}_1$ was able to achieve configurations that were close to the target, but failed at accurate alignment (see Figure 4-9a). On the other hand, the second baseline controller (see Figure 4-9b) achieved accurate partial alignment with early intermediate configurations, but struggled to reason about the later configurations where the current and target configuration were highly dissimilar. The hybrid controller was able to closely shape the rope to match the goal configuration by using the coarse representation first and switching to the more expressive representation to stabilise around the goal configuration. Similar behaviour was observed in the Cloth Unfold I task. The first baseline was only able to translate and reorient the fabric to maximise coverage. The second baseline used a more detailed representation to position and smooth the fabric. However, the hybrid controller leveraged the coarse representation to position the fabric close to the goal where it could attend to finer details to maximise goal coverage.

Table 4.4: Tracking Task: Raw goal coverage (IoU) at time $t = N$ on trajectory tracking for baseline and hybrid controllers with varying criterion $G_{\epsilon-(\cdot)}$. These metrics were computed over 500 trials where the reference trajectory was held constant. We refer the reader to Figure 4-11 for further comparison.

	$\overline{\text{IoU}}(\sigma)$		
Controller Type	Rope Unloop	Cloth Unfold I	Cloth Unfold II
$\bar{\pi}_1(z; h^8)$	0.154 (0.120)	0.416 (0.081)	0.521 (0.083)
$\bar{\pi}_2(z; h^{32})$	0.067 (0.036)	0.451 (0.064)	0.517 (0.088)
$\pi(z; G_{\text{4-prox}}, \{h^8, h^{32}\})$	0.124 (0.104)	0.457 (0.070)	0.540 (0.085)
$\pi(z; G_{\text{4-prox}}, \{h^{32}, h^{64}\})$	-	0.424 (0.073)	0.506 (0.040)



(a) Rope Unloop (IoU) (b) Cloth Unfold I (IoU) (c) Cloth Unfold II (IoU)

(d) Rope Unloop ($\Pr\{k = 32\}$) (e) Cloth Unfold I ($\Pr\{k = 32\}$) (f) Cloth Unfold II ($\Pr\{k = 32\}$)

Figure 4-11: Rope Unloop, Cloth Unfold I and Cloth Unfold II: Raw goal coverage over $N \times M$ time steps for baseline (blue) and hybrid controllers on Rope Unloop (left) and Cloth Unfold I (right). All hybrid controllers used criterion $G_{\text{4-prox}}$ and models $\{h^8, h^{32}\}$ while all baseline controllers used h^8. For the majority of task stages, the hybrid controllers performed at par with the baseline controllers. In the case of cloth unfolding, the hybrid controller exceeded the baseline by 4%.

(a) Cloth Unfold II

(b) Cloth Fold

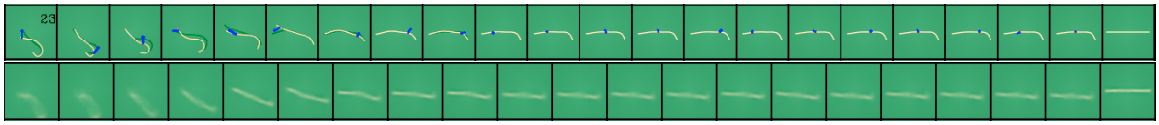
Figure 4-12: Cloth Unfold II and Cloth Fold: Snapshots of the control inputs generated by the hybrid controller $\pi(z; G_{4-prox}; \{h^8, h^{32}\})$ during execution of $M = \{3, 4\}$ stages (rows) over time horizon $N = 40$ (columns). A red dot overlays the cloth configuration to indicate the selected pick point. The box in the lower left corner of the image shows the selected pull direction which corresponds to the force vector that will be applied at the pick point. The next frame shows the resulting cloth configuration. At the beginning of each stage, the controller applies large forces on the cloth to position it closer to the sub-goal configurations. As the cloth approaches the sub-goal, the force decreases suggesting that the controller was reasoning over the finer representation to align the cloth with the target configuration.

Quantitative results showed that the hybrid controller yielded similar or higher performance in Cloth Unfold I and II to the strongest baseline, $\bar{\pi}_2(z; h^{32})$. It was expected that $\bar{\pi}_2$ would outperform all models based on model selection in the Shape Cloth suite. However, goal coverage metrics summarised in Table 4.4 indicated that both baseline controllers were sub-optimal compared to the hybrid controller $\pi(z; G_{\text{4-prox}}, \{h^8, h^{32}\})$. Figure 4-11 shows the final goal coverage during each time step of the trajectory execution. The histograms in the figure convey the probability that the hybrid controller switched to a fine-grained representation. The coarse-to-fine strategy lead to a 4% increase in performance over baseline π_1 and outperformed both controllers by a similar margin in Cloth Unfold II. A sample of the control inputs generated by the hybrid controller along each stage of the Cloth Unfold II and

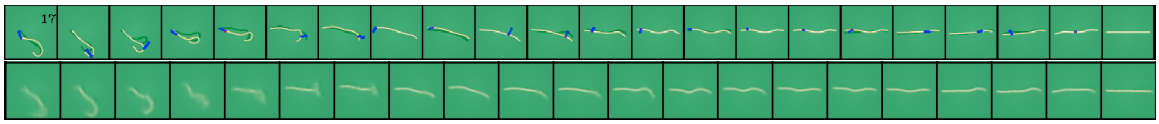
Cloth Fold demonstrations are shown in Figure 4-12. In early stages of tracking, the outputs hybrid controller were relatively larger than at later stages where it reasoned about the more detailed representation to refine alignment between the current and goal configuration.

These experiments showed that a simple hybrid control design with mutable state representation effectively used a coarse to fine strategy to outperform fixed-representation controllers. Figure 4-13 presents reconstructions of latent trajectories (with $N = 20$ time steps) taken by the constituents of the hybrid controller. Though we have yet to report direct measures of mutual information between the observed and latent dynamics, these reconstructions give a sense of the amount of information lost during the encoding phase. Based on this qualitative analysis, the representations with lower k conveyed coarser spatial information. Consequently, controllers with coarser representations selected control inputs that were effective at quickly maneuvering the object to the correct location to match the goal configuration. They failed, however, to make the necessary fine adjustments to achieve perfect alignment. On the other hand, representations with larger k retained information that enabled more precise alignment. In Figure 4-13b, a model with $k = 16$ achieved near perfect alignment about four timesteps sooner than the allotted horizon, but the lower dimensional counterpart achieved the "flat" shape sooner. Refining the representation by further increasing k to 32 dimensions allowed information about complex geometries to percolate to the optimisation algorithm, however the embedding space appears to convey extra information that manifests as slightly distortion in the reconstructions. With the appropriate control mode selection strategy, the hybrid controller was able to chose representations presented the right level of information for increasing the accuracy of the trajectory tracking solution.

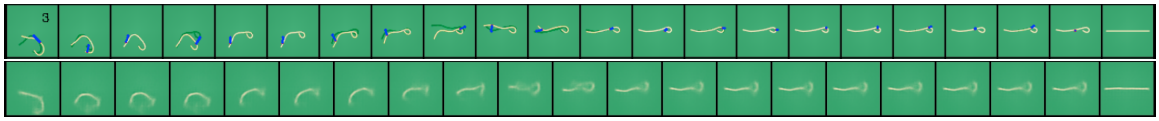
Our choice to vary the latent dimensionality as a means of adapting the state representation is analogous to a simple attention mechanism. The hybrid controller's strategy could be interpreted as attending over high-level information about the ob-



(a) $k = 8$



(b) $k = 16$



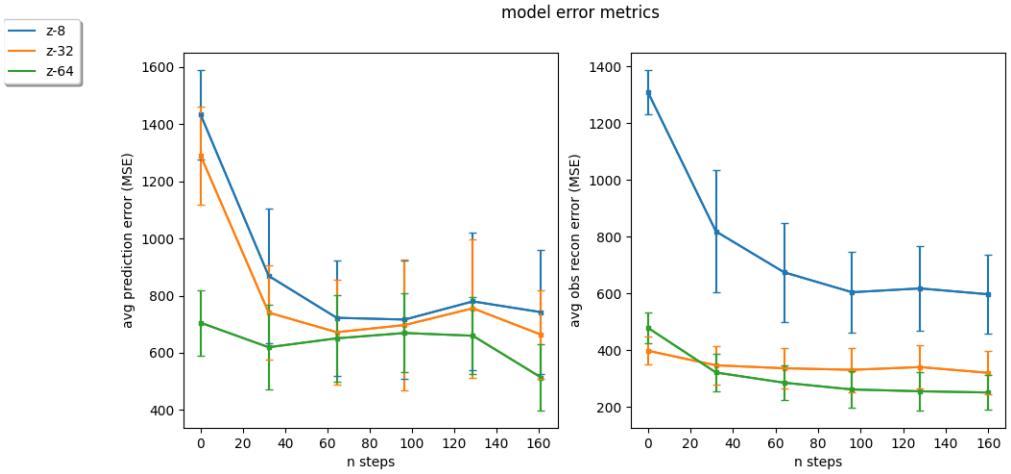
(c) $k = 32$

Figure 4-13: Flatten Rope: Planned versus executed trajectories during rope flattening task execution for models trained with $k \in \{8, 16, 32\}$. Each row shows an annotated trajectory of observations and actions. In each frame, the action is indicated by a blue arrow and a ghost of the next frame is overlaid to show the effect of the action. Reconstructions of the planned latent trajectory $\{z_t^k\}_{t=0}^{N-1}$ (lower) are shown for each observation and action pair.

ject's location and orientation when the current configuration is far from the target. As it approaches the target configuration, the controller attends over more complex aspects of the object's geometry to reach and stabilise around the target. Observe the behaviour of model prediction error for the Cloth Unfold I task in Figure 4-14. Recall that there was high self-occlusion in the initial configuration of the unfolding task. The models with higher k had relatively low prediction error despite the cloth's configuration. Lower dimensional models lacked sufficient details to make good state predictions for complex cloth configurations. The reconstruction error suggests that either the encodings were too lossy or that features were encoded that contained enough geometric information, but were irrelevant for dynamics prediction for complex configurations. The model with highest k captured sufficient information for maintaining consistently prediction errors for a variety of configurations. A limitation of using this ensemble-style architecture is the lack of regulation over the features with respect to the execution task. In other words, our current models learn features to maximise forward model prediction. The only way to sub-select features for, say, computing goal distance at different stages in the task is to reduce k and therefore

lose prediction performance.

Rather than prototyping an adaptive controller using an ensemble of models, we would consider implementing a single model with high k and pre-training it to maximise forward prediction (Figure 4-14). During task execution an online algorithm can employ classical feature filtering methods to sub-select features that are relevant for a particular task phase [37]. We may also explore model-based analogues of the Q-attention algorithm discussed in Chapter 2.



(a) Cloth Unfold I

Figure 4-14: Rope Unloop and Cloth Unfold I: Latent prediction error (MSE) and reconstruction error (MSE) for models trained with $k = \{8, 16, 32, 64\}$.

Overall, we found that a simple hybrid control scheme could successfully trade-off the advantages of coarse and fine representations to outperform controllers with fixed representations. Initial model evaluation uncovered the surprising trade-offs of increasing the latent space dimensionality. Our experiments showed that the hybrid control strategy was more beneficial in domains with objects of higher complexity than 1D deformable objects (e.g., rope). The success of the hybrid controller in simulation motivated further testing in the physical setup described in Chapter 5.

Chapter 5

Physical System

After confirming our hypothesis in the simulation experiments in Chapter 4, we made the engineering effort to reproduce the system on a physical platform. This chapter reviews the components of the physical system and shows examples of task execution in the real world. Section 5.2 describes the perception pipeline that generated model inputs. The architecture of the end-effector controller is explained in Section 5.3. It also provides examples of rope and cloth manipulation in a physical Rope and Cloth Shape environment.

5.1 System Overview

The physical system replicated the simulated table-top setup in Section 4.2. Figure 5-1 depicts a schematic layout of the system. It consisted of an overhead camera and an arm that was mounted to the table. Our platform of choice was the Franka Emika Panda Arm[1], a 7DOF robot that is capable of position and torque control. It's integration with the Robot Operating System (ROS)[2] and rich software interface made it an ideal choice for developing our table-top manipulation tasks. The scene was recorded using a Azure Kinect[3] sensor that was suspended 0.7m above the table surface. Our real-world implementation consisted of two main parts; a perception

[1]https://frankaemika.github.io/docs/franka_ros.html
[2]https://www.ros.org/
[3]https://docs.microsoft.com/en-us/azure/kinect-dk/

pipeline and a control pipeline.

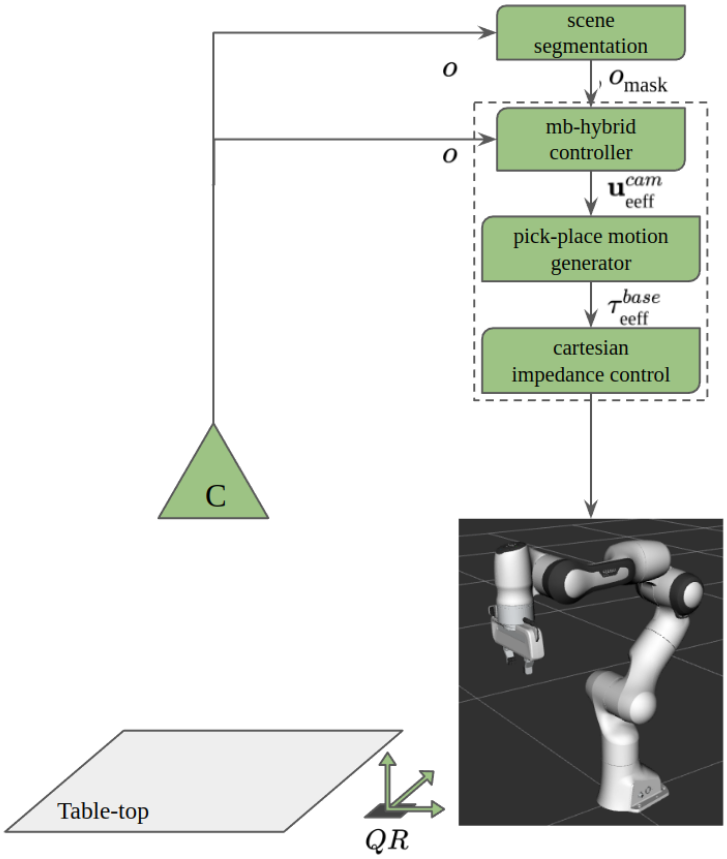


Figure 5-1: System Overview: A physical system used to conduct real world experiments where a robot arm manipulated rope and cloth on a table. The robot platform we used was the Franka Emika Panda arm. It was mounted to a table shown in grey with an overhead camera suspended above the table surface. The system software of the system was composed of a perception pipeline, the hybrid controller (Section 3.5.2), a motion generator and a cartesian impedence controller. We used the Robot Operating System (ROS) to unify these components.

5.2 Perception System

The perception pipeline was responsible for transforming images recorded from the Kinect to a form that was compatible with the hybrid controller's models (see Figure 5-2). First the pipeline pre-propcessed the raw color and depth images into the correct input size. The process began by cropping the color images into square im-

ages. We then used the Gaussian Pyramid implementation in OpenCV[4] to iteratively downsize the color image from the square resolution to a $64 \times 64 \times c$ image, where c=3 and 1, respectively. This custom pre-processing step was necessary to minimise aliasing artifacts that occurred while downsizing the images. During the iterative process, we recorded the scale factor to undo the downsizing operation for individual pixels, as we'll later explain. The same process was repeated for depth images, which required an extra undistortion step to rectify the images generated by the depth sensor.

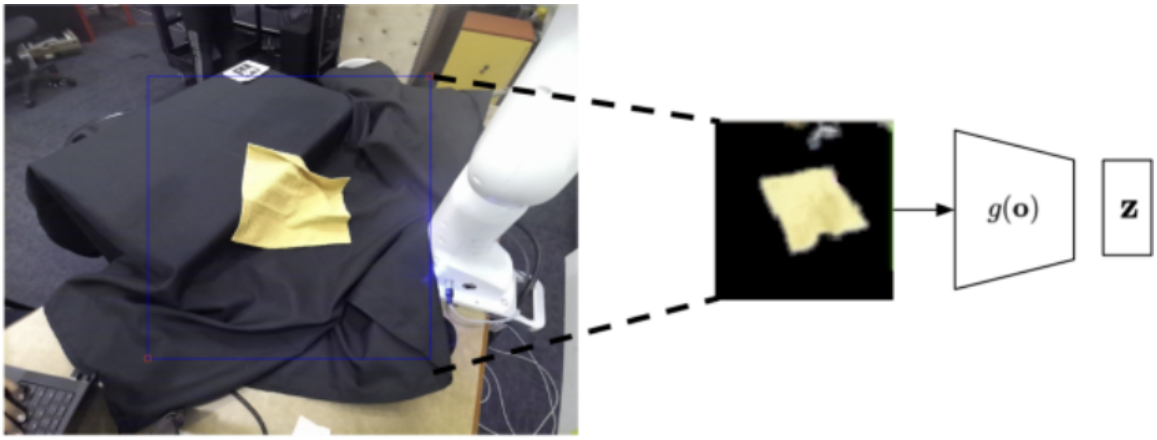


Figure 5-2: A primary function of our perception pipeline was to pre-processes images captured by the Kinect sensor into inputs for the hybrid controller's models. A raw image is shown on the left with a blue box indicating where it was cropped to form the 64×64 image on the right. The cropped image was downsized using standard downsampling methods.

Another important role of the perception system was to generate an object segmentation mask that excluded pixels belonging to the robot. The mask was used to cull the action sampling space for the hybrid controller, as discussed in Section 3.5.2. The object segmentation mask was generated by a custom algorithm based on Watershed Segmentation [47]. We made an important assumption about the scene to simplify the segmentation problem; namely, that the background was uncluttered, untextured and had high contrast with the object (see Figure 5-3). This allowed us to treat object segmentation as foreground segmentation. Consequently, the robot was included in the mask. To remove pixels belonging to the robot, we took advantage of

[4]https://docs.opencv.org/3.4/

the overhead camera setup, assuming that the robot and the object would not coexist in the same depth plane (because of the table). So, we used depth information to remove pixels that corresponded to depth values below an empirically determined threshold.

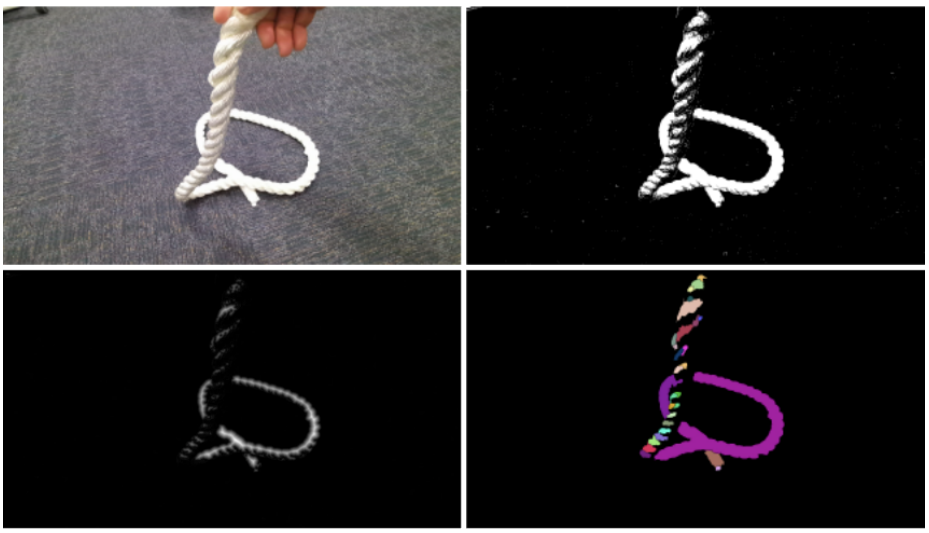
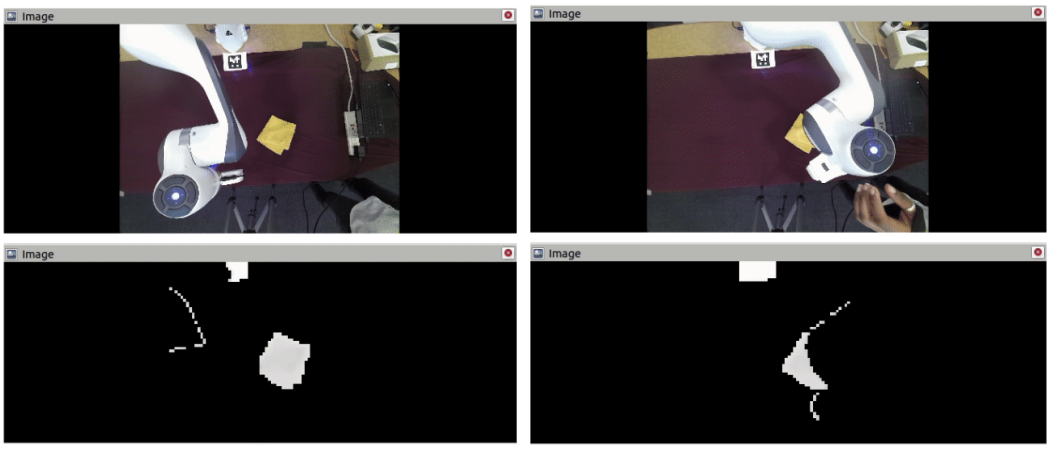


Figure 5-3: Intermediate outputs of the Watershed Segmentation algorithm. The top left image is the input to the segmentor and the bottom right image shows labeled contours of the object. We treated all non-zero labels as the rope object, with the key assumption that the background was uncluttered, had minimal texture and high contrast with the object.

As alluded to before, the perception provided a crucial means to transform control inputs $\mathbf{u}$ determined by the hybrid controller from the normalised pixel space to the images recorded from the sensor. The pixel values corresponding to the chosen action in the downsized images seen in Figure 5-2 were upscaled using scalar values recorded during the downsizing step. Once in the native resolution of the sensor images, we could use the camera calibration data to map the pick points in the image space to the 3D world coordinate frame. The place points were determined by scaling the displacement vector shown in Figure 5-5 by an empirically determined scale factor.



(a) t = 0s (b) t = 2s

Figure 5-4: Example of the robot filter used to mask pixels belonging to the robot arm. The two frames were taken from an overhead recording of the scene where the robot arm was passing over the yellow cloth (top). The segmentation mask (bottom) highlights the cloth pixels that will be a part of the action sampling set in Algorithm 2. Note that the segmentation mask will undergo cropping before being passed to the models. In other words, the QR code highlighted at the top of the image will not be seen by the models.

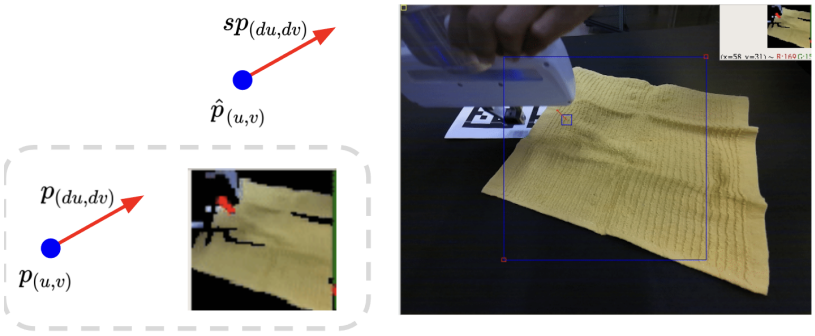


Figure 5-5: Post processing was required to map the control inputs produced by Algorithm 2 to the native image. The input image to the hybrid controller's model is shown on the left with **u** near the top left corner of the image (see small red arrow). The vector is mapped into the native resolution of the image (right) where it can be further mapped into a 3D coordinate frame using the camera calibration data. The mapped vector is indicated in the right image by a blue square with a red arrow near the gripper. The location matches that of its original location and direction in the model input image.

5.3 Control System

The control pipeline shown in Figure 5-6 (see dashed box) consisted of our hybrid controller, a motion generator and a cartesian impedence controller. The pipeline

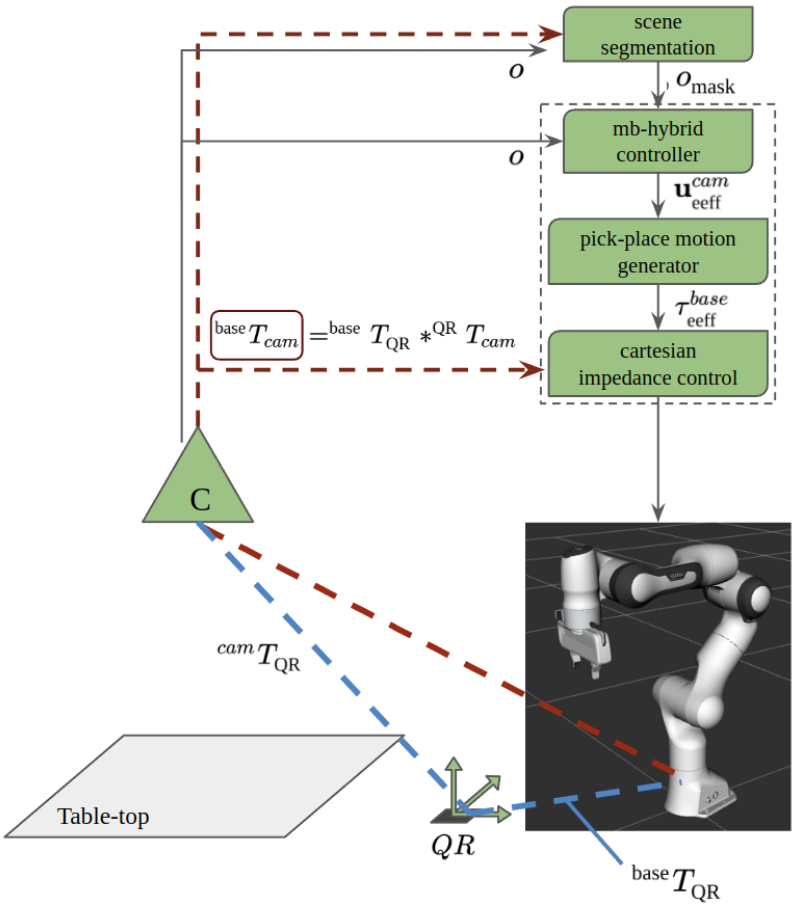


Figure 5-6: A more detailed view of the system architecture, including important transforms for relating the sensor coordinate frame and the robot's base frame.

took image observations that were pre-processed by the perception pipeline in Section 5.2. It produced a sequence of end-effector poses in the robot's coordinate frame that were later relayed to Franka Emika Panda's low-level control pipeline through the Franka Control Interface.

The control loop began with the hybrid controller. Provided a color image, depth image and object mask the hybrid controller generated a coarse latent state and optimised a pick-and-place action in the compressed pixel space. We used the post processing step in Section 5.2 to map the action back to the native resolution of the images captured from the Kinect. This step was crucial as it allowed us to use the camera calibration data to map the pixels in the native image to the world coordinate

frame. We did this using QR-code-based landmark features called April Tags.

The hybrid controller's outputs were mapped to the robot coordinate frame through a sequence of transforms that related the camera coordinate frame with the robot's base. Pixel locations in the camera frames were transformed into the base frame using April Tag landmarks that were mounted in the scene. We used the April Tags software to detect the QR code and localise it in the camera coordinate frame. Provided an accurate pose estimate of the QR code, we could map actions selected by the hybrid controller to the robot's coordinate frame. A manual calibration process was used to measure the transform between the QR code and robot base ${}^{\text{base}}T_{QR}$ (Figures 5-6 and 5-7). The April Tag software was available as a ROS package, so it was easy to integrate into our perception and control pipeline.

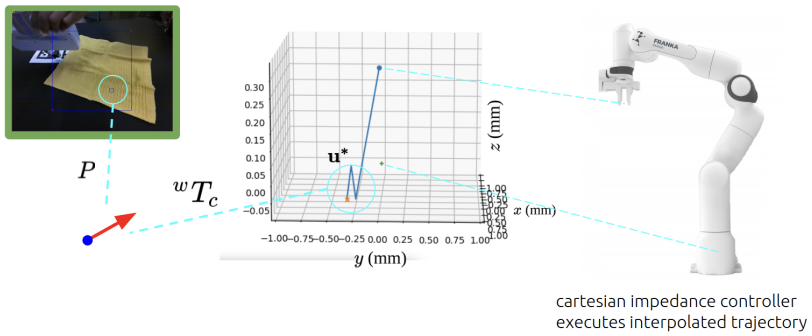


Figure 5-7: An illustration showing how actions in the pixel space are transformed into the robot's base frame. On the left, the selected control input (encircled in cyan) is transformed from the camera coordinate frame to the world coordinate frame. The the custom motion generator computes a rough trajectory (blue lines) to outline the pick-and-place action and interpolates it to produce a smooth trajectory of end-effector poses. The robot arm is shown on the right to indicate the position of the gripper and the base of the robot in the 3D plot.

Once the output of the hybrid was transformed to the robot's base, we generated a motion plan to execute the pick-and-place action. This required implementing a motion trajectory planner that produced a trajectory of end-effector poses that executed the pick-and-place action. The motion generator interpolated a sparse intermediate trajectory shown in the 3D plot in Figure 5-7.

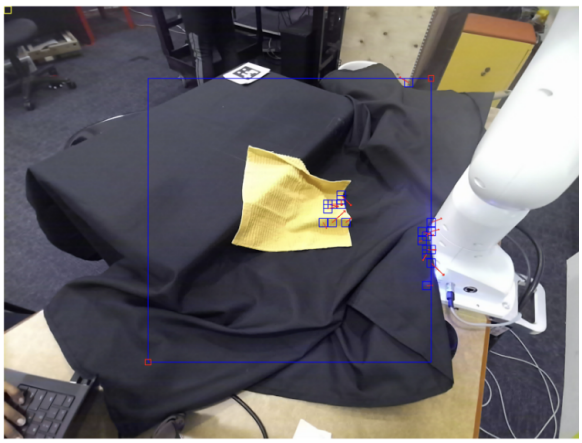


Figure 5-8: A view of the system attempting to pick the corner of a piece of cloth. The blue box indicates the cropped region of the image that is downsized and given as input to the hybrid controller's models. The outputs of the controller are marked by small blue squares with red vectors to indicate the intended displacement.

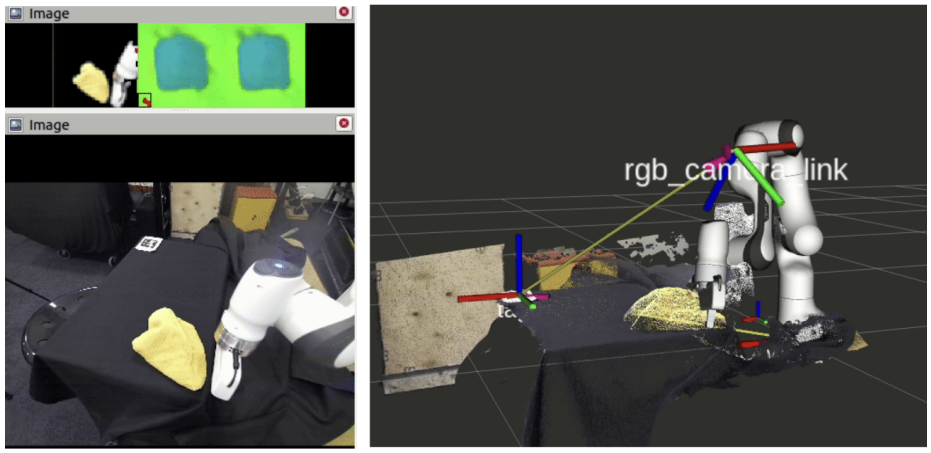


Figure 5-9: Three views of the scene: (right) a rendering of the overhead Kinect sensor, QR code attached to the table surface and the scene point cloud; (bottom left) the raw images inputs to the perception system taken from the camera pose shown on the right; (top left) the (color) output of the perception pipeline and reconstructions of the cloth state generated by models trained in simulation with domain randomisation.

Overall the hybrid control system was able to generate useful outputs despite not being trained on real world data. We rendered its selected control inputs over real images of the scene(see Figure 5-8) and observed that the system reasoned over the shape of the cloth, attempting to pick the right edge of the cloth. There were, however, failure modes caused by the calibration of the April Tag setup that resulted in

the gripper missing a pick (see Figure 5-9). Additionally, the gripper state (whether it is open or closed) was occasionally not synchronised with the location of the pick. As a result there were moments when the gripper would reach the cloth, but the gripper had begun closing before the fabric was reached. These technicalities are resolvable with minor engineering hacks; e.g., ensuring the QR code is consistently fixed relative to the robot's base and using a state machine to align the gripper state with key points in the pick-and-place motion.

This physical setup provided a means for testing the hybrid model in a real world setting. However, the hybrid controller was not rigorously tested on the Cloth Unfold I and II tasks. The robustness of the models to variations in the cloth color, stiffness and texture should also be tested. We delegate these experiments to future work.

Chapter 6

Conclusion

At the beginning of this thesis, we introduced the problem of performing multi-stage manipulation tasks in cases where the state space is high-dimensional. The task domains that we addressed were centered on manipulating deformable objects to change their shape to a high degree. We aspired to design an adaptive controller with the capability of attending to different levels of representational information in order to complete a multi-stage task by following expert guidance. The main challenge in this problem was induced by the presence of deformable objects, which are under-actuated non-linear systems with high-dimensional state space. We formalised the guided multi-stage task execution problem as a trajectory tracking control problem and applied machine learning to overcome the complexity of tracking in high-dimensional spaces. We developed a hybrid controller to prototype the adaptive controller and demonstrated the utility of adapting the state representation in a wide range of simulated experiments. The experiments benchmarked the performance of the hybrid controller against baselines that extend prior work. We also deployed the hybrid controller to a control pipeline on a physical platform as a stepping stone towards doing long horizon object manipulation in the real world. Finally, we envision applications beyond deformable object manipulation where this approach could form the basis for solutions to manipulation problems around construction (disassembly), food preparation, or problems outside of the robotic manipulation domain; multi-agent systems, humanoid control and control of soft robots.

One immediate area of expansion in this work is to run a large suite of physical experiments that validate our proposed system's ability to track in a real-world setting. So far, the physical system runs end-to-end with models trained in simulation using domain randomisation. Recall in Figure 5-9 that the hybrid controller generated sensible pick-and-place actions for smoothing the cloth. However, the reconstructions in the top left visualisation indicate that the model made errors in reproducing the shape of the cloth. A large sim-to-real gap, a common ailment in robot manipulation [59], was likely the cause. Often the gap is dealt with by domain randomisation methods that require prior knowledge about the type of variations and the range of variations that would be anticipated in the deployment context. Our models were trained on fabrics of different color (including yellow), different lighting conditions, as well as different fabric masses and friction values, yet there may have been enough of a difference in the image statistics - given real downsized images instead of crisp simulation renderings - to cause a larger gap than we anticipated. For this reason, we hope to explore more sophisticated sim-to-real transfer techniques that will allow us to make the most of a rich corpora of simulation data [14, 20, 2]. Another possibility is to consider relying more strongly on depth inputs [52, 34].

Another consideration for future work is to enrich our model architecture to achieve higher performance on our task suite. The architectures we described in Chapter 3 are relatively general, thanks to their simplicity. That is, the loss and architecture are not biased towards any particular task, so the models were able to represent a variety of objects - 1D, 2D and 3D. We may consider incorporating terms into the training loss or modifying the architecture to steer the models learn more constrained, and possibly more interpretable features. For example, recent methods use graphical neural networks to learn mesh-based representations of fabric for more accurate state estimation of configurations with high self-occlusion during cloth folding tasks [23, 3]. The model architecture we used for estimating dynamics in the latent space may also be upgraded to emulate Recurrent State Space Models (RSSM),

proposed by Hafner et al. (2019) [12, 13, 44]. RSSM and its derivatives are capable of representing stochastic dynamics, which would be appropriate feature to incorporate to our deformable object models.

Further development of this research direction could benefit manipulation tasks beyond 1D, 2D and 3D deformable object manipulation. This thesis did not address particle-based systems that would characterise deformable objects like food, liquids and granular material. Previous work on manipulation of food items, e.g., peeling a banana, that planning solutions with hand-crafted and fixed models would benefit from extensions of our approach [8, 60]. Another consideration, force sensing and multiple arms could be incorporated into our framework to enable food manipulation tasks that require precise force inputs to successfully complete a task. One example might be to fold a burrito while minimising spillage. In the more general realm of robot manipulation, assembly and disassembly of decomposable objects share similar challenges in reasoning over high dimensional states, where the number of constituents may vary [21]. Transcending manipulation tasks, ideas in this line of work could yield creative solutions to highly related problems in the multi-agent control setting, control algorithms for humanoid robots and soft robots [16, 32, 7].

Bibliography

[1] Baris Akgün, M. Cakmak, J. W. Yoo, and A. Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 391–398, 2012.

[2] Rika Antonova, Jingyun Yang, Priya Sundaresan, Dieter Fox, Fabio Ramos, and Jeannette Bohg. A bayesian treatment of real-to-sim for deformable object manipulation. *CoRR*, abs/2112.05068, 2021.

[3] Solvi Arnold, Daisuke Tanaka, and Kimitoshi Yamazaki. Cloth manipulation planning on basis of mesh representations with incomplete domain knowledge and voxel-to-mesh estimation. *CoRR*, abs/2103.08137, 2021.

[4] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37:286–298, 2007.

[5] Alexander Clegg, Wenhao Yu, J. Tan, C. Liu, and Greg Turk. Learning to dress. *ACM Transactions on Graphics (TOG)*, 37:1 – 10, 2018.

[6] Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 465–472, Madison, WI, USA, 2011. Omnipress.

[7] Cosimo Della Santina, Christian Duriez, and Daniela Rus. Model based control of soft robots: A survey of the state of the art and open challenges, 2021.

[8] Nadia Figueroa, Ana Lucia Pais, and Aude Billard. Learning complex sequential tasks from demonstration: A pizza dough rolling case study. *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 611–612, 2016.

[9] J. Grannen, Priya Sundaresan, Brijen Thananjeyan, Jeff Ichnowski, A. Balakrishna, Vainavi Viswanath, Michael Laskey, J. Gonzalez, and Ken Goldberg. Learning robot policies for untangling dense knots in linear deformable structures. 2020.

[10] L. Guibas, Christopher Holleman, and L. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach. *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, 1:254–259 vol.1, 1999.

[11] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. 2021.

[12] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *CoRR*, abs/1811.04551, 2018.

[13] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *CoRR*, abs/2010.02193, 2020.

[14] Nicklas Hansen, Yu Sun, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *CoRR*, abs/2007.04309, 2020.

[15] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Kumar Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Visuospatial foresight for multi-step, multi-task fabric manipulation. 2020.

[16] S. Iida, S. Kato, K. Kuwayama, T. Kunitachi, M. Kanoh, and H. Itoh. Humanoid robot control based on reinforcement learning. In *Micro-Nanomechatronics and Human Science, 2004 and The Fourth Symposium Micro-Nanomechatronics for Information-Based Society, 2004.*, pages 353–358, 2004.

[17] Stephen James and Pieter Abbeel. Coarse-to-fine q-attention with tree expansion. 2022.

[18] Stephen James and Andrew J. Davison. Q-attention: Enabling efficient learning for vision-based robotic manipulation, 2021.

[19] Andrew M. Ladd and L. Kavraki. Using motion planning for knot untangling. *The International Journal of Robotics Research*, 23:797 – 808, 2004.

[20] Alex X. Lee, Coline Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, Claudio Fantacci, Jose Enrique Chen, Akhil Raju, Rae Jeong, Michael Neunert, Antoine Laurens, Stefano Saliceti, Federico Casarini, Martin A. Riedmiller, Raia Hadsell, and Francesco Nori. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. *CoRR*, abs/2110.06192, 2021.

[21] Youngwoon Lee, Edward S. Hu, Zhengyu Yang, Alex Yin, and Joseph J. Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. *CoRR*, abs/1911.07246, 2019.

[22] Timothée Lesort, Natalia Díaz Rodríguez, Jean-François Goudou, and David Filliat. State representation learning for control: An overview. *CoRR*, abs/1802.04181, 2018.

[23] Xingyu Lin, Yufei Wang, and David Held. Learning visible connectivity dynamics for cloth smoothing. *CoRR*, abs/2105.10389, 2021.

[24] W. Lui and Ashutosh Saxena. Tangled: Learning to untangle ropes with rgb-d perception. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 837–844, 2013.

[25] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning, 2020.

[26] Dale Mcconachie, Andrew Dobson, Mengyao Ruan, and Dmitry Berenson. Manipulating deformable objects by interleaving prediction, planning, and control. *The International Journal of Robotics Research*, 39:957 – 982, 2020.

[27] Astrid Merckling, Nicolas Perrin-Gilbert, Alexandre Coninx, and Stéphane Doncieux. Exploratory state representation learning. *CoRR*, abs/2109.13596, 2021.

[28] Ashvin Nair, D. Chen, Pulkit Agrawal, Phillip Isola, P. Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2146–2153, 2017.

[29] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.

[30] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

[31] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. *CoRR*, abs/1611.07828, 2016.

[32] Rodrigo Pérez-Dattari, Carlos Celemin, Javier Ruiz-del-Solar, and Jens Kober. Continuous control for high-dimensional state spaces: An interactive learning approach. *CoRR*, abs/1908.05256, 2019.

[33] Thomas Power and Dmitry Berenson. Keep it simple: Data-efficient learning for controlling complex systems with simple models, 2021.

[34] Jianing Qian, Thomas Weng, Luxin Zhang, Brian Okorn, and David Held. Cloth region segmentation for robust grasp selection. *CoRR*, abs/2008.05626, 2020.

[35] J. Rawlings, D.Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. 01 2017.

[36] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[37] Irene Rodríguez-Luján, Ramón Huerta, Charles Peter Elkan, and Carlos Santa Cruz. Quadratic programming feature selection. *J. Mach. Learn. Res.*, 11:1491–1516, 2010.

[38] Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *IEEE Transactions on Robotics*, 23:1141–1150, 2007.

[39] John Schulman, Jonathan Ho, C. Lee, and P. Abbeel. Learning from demonstrations through the use of non-rigid registration. 2013.

[40] John Schulman, Alex X. Lee, Jonathan Ho, and P. Abbeel. Tracking deformable objects with point clouds. *2013 IEEE International Conference on Robotics and Automation*, pages 1130–1137, 2013.

[41] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, Katsu Yamane, Soshi Iba, John Canny, and Ken Goldberg. Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor. 2019.

[42] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, Katsu Yamane, Soshi Iba, John Canny, and Ken Goldberg. Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor, 2019.

[43] Bilha-Catherine Githinji & Julie Shah. Generalising deformable object manipulation by learning to learn from demonstrations. 2021.

[44] Nitish Srivastava, Walter Talbott, Martin Bertran Lopez, Shuangfei Zhai, and Josh M. Susskind. Robust robotic control from pixels using contrastive recurrent state-space models. *CoRR*, abs/2112.01163, 2021.

[45] Priya Sundaresan, J. Grannen, Brijen Thananjeyan, A. Balakrishna, Michael Laskey, Kevin Stone, J. Gonzalez, and Ken Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418, 2020.

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

[47] Richard Szeliski. *Computer Vision - Algorithms and Applications*. Texts in Computer Science. Springer, 2011.

[48] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018.

[49] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. pages 5026–5033, 2012.

[50] H. Wakamatsu, E. Arai, and S. Hirai. Knotting/unknotting manipulation of deformable linear objects. *The International Journal of Robotics Research*, 25:371 – 395, 2006.

[51] F. Wang, E. Burdet, R. Vuillemin, and H. Bleuler. Knot-tying with visual and force feedback for vr laparoscopic training. *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 5778–5781, 2005.

[52] Thomas Weng, Sujay Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. 2021.

[53] Yi-Lin Wu, W. Yan, Thanard Kurutach, Lerrel Pinto, and P. Abbeel. Learning to manipulate deformable objects without demonstrations. *ArXiv*, abs/1910.13439, 2019.

[54] Y. Yamakawa, A. Namiki, and M. Ishikawa. Dynamic high-speed knotting of a rope by a manipulator. *International Journal of Advanced Robotic Systems*, 10, 2013.

[55] Mengyuan Yan, Gen Li, Yilin Zhu, and Jeannette Bohg. Learning topological motion primitives for knot planning. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9457–9464, 2020.

[56] W. Yan, Ashwin Vangipuram, P. Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. *ArXiv*, abs/2003.05436, 2020.

[57] Gang Yao, R. Saltus, and A. Dani. Shape estimation for elongated deformable object using b-spline chained multiple random matrices model. *Int. J. Intell. Robotics Appl.*, 4:429–440, 2020.

[58] Ahmed Zgaren, Wassim Bouachir, and Riadh Ksantini. Coarse-to-fine object tracking using deep features and correlation filters. *CoRR*, abs/2012.12784, 2020.

[59] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *CoRR*, abs/2009.13303, 2020.

[60] Fucai Zhu, Akihiko Yamaguchi, and K. Hashimoto. Case study of model-based reinforcement learning with skill library in peeling banana task. 2019.