# Stabilizing demonstration trajectories of linear deformable objects for robotic shoe tying

by

Michelle Tan

S.B., Computer Science and Engineering, Massachusetts Institute of Technology 2020

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

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021



Author .....
Department of Electrical Engineering and Computer Science
May 20, 2021

Certified by .....
Russ Tedrake
Toyota Professor of EECS, Aero/Astro, MechE.
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

# Stabilizing demonstration trajectories of linear deformable objects for robotic shoe tying

by

Michelle Tan

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

Tying a shoelace knot is commonly seen as a milestone for young children as they learn how to use their hands to execute complex motions. Humans are able to consistently tie a shoelace knot on a wide variety of shoes, even if they haven't seen the shoe before. In comparison, shoe tying is a problem that the robotics community is still very far from solving, since it breaks many assumptions of existing algorithms in robotic manipulation. Some key difficulties of getting a robot to consistently tie any shoe include the complex dynamics, the deformable nature of the shoelaces, the long time horizon, the dexterity needed to manipulate flexible objects, and the large variation between different shoes.

In this thesis, we make progress towards shoe tying by making a robot that is able to improve on a given demonstration by making it more robust to initial conditions of a shoe in simulation. This is motivated by the fact that when humans learn to tie a shoe for the first time, they are carefully taught a procedure for making the knot, including how to hold the shoelaces and how to be able to tell if a shoelace knot is good or bad. The impressive part is that they can quickly adapt this procedure to any shoe. In this thesis, I discuss the following three contributions towards refining a demonstration to work for any shoe. The first contribution is the development of an open-sourced configurable shoe simulator environment that allowed us to tie shoes completely in simulation. The second contribution is a formulation and evaluation of direct policy search via CMA-ES with the goal of optimizing a given shoe tying policy for robustness. The third contribution is a formulation and analysis for learning the dynamics and cost on a latent state and an evaluation of the learned model for control. We found that CMA-ES and learning latent approximate information states were both successful techniques. Both were able to stabilize a demonstration for robustness on initial conditions of the shoelaces $>95/100$ of the time.

Thesis Supervisor: Russ Tedrake
Title: Toyota Professor of EECS, Aero/Astro, MechE.

# Acknowledgments

There are lots of people who made this thesis possible.

First of all, I'd like to extend my deepest gratitude for Russ Tedrake for welcoming me into his lab during a pandemic. His keen eye for choosing captivating and important problems led me towards this project. He constantly led me towards thinking about the hard problems while also helping me feel supported along the way. It's been the biggest honor to get to be a part of this group. On top of that, he also introduced me to the two greatest collaborators one could ever hope for for their MEng.

Mark Petersen and Terry Suh were generous with their time, knowledge and support. They were always down for small reading groups when I had questions about a topic. They were also so willing to help me at all hours of the day when I got stuck. Their inputs during our weekly meetings constantly challenged yet inspired me. They truly gave me all the support I needed to grow and learn so much this year.

I'd also like to thank the Robot Locomotion Group for being so welcoming towards me. Their sharp comments during group meetings always inspired me to think critically. I'm so proud to have been a part of this amazing lab. This year has been a total dream come true.

I'd also like to thank the Drake Developer team for building such an amazing tool and also for their guidance with the shoe simulator.

Finally, I'd also like to thank my roommates, friends, and family for their ongoing support.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The task of tying shoes contains many features that robots are notoriously poor at handling. Deformable objects like ropes serve as a benchmark challenge for robots, which often rely on mathematical state representations to manipulate objects. Representing the exact state of a rope would involve an infinite amount of DOFs, which makes it difficult for robots to reason about. In addition, tying a shoe requires a lot of finger dexterity and finesse, which a majority of robots lack in hardware or don't have the software/algorithmic capabilities to fully utilize yet. Ropes are also incredibly dynamic and flexible, making the problem aggressively underactuated. Unlike many existing papers on deformable object manipulation, we aim to tie shoes in-air with no external rope supports, which means that the grippers have to both be constantly holding the ropes up while also tying the knot. On top of all of that, the large number of precise steps required means that a slight imprecision in the tying might require us to start all over again.

## 1.1 Related work

### 1.1.1 Representing the state and dynamics of deformable objects

Much prior work in robotics depends on having an accurate mathematical way of describing the world, but writing down a controller that uses the exact state and dynamics of deformable objects is an impractical task due to the high number of degrees of freedom involved. Humans are able to manipulate deformable objects with relative ease, despite not knowing the entire state or exact dynamics model of these objects. We intuitively know that only a small subset of the information is important, and we're able to extract what to focus on at any given time. Researchers studying the manipulation of deformable objects have had to find ways of writing down what's important.

One common approach for representing the state of ropes is to utilize ideas from knot theory. [33, 27, 35, 37] all utilize variations of Gauss codes to write the topological state of ropes. [35] also considers analyzing the lengths of the segments in order to tightly tie knots. Knot theory provides an abstract way of representing the toplogy of a specific knot but it doesn't take into account the dynamics of the rope or the specifics of how to tie a knot with a physical robot.

Recently, researchers have been looking into learned representations to represent the state and dynamics of a rope in a compressed way. [31] learns a transferable and interpretable visual representation called dense depth object descriptors(DDODs) and shows its success in policies for simple knot manipulation tasks. [24] uses a deep autoencoder to represent an elastic rod's centerline in a low-dimensional latent state. [22, 21] learn a dynamics model from a large number of random interactions from the environment. [41] learns a latent dynamics model in addition to learning a visual representation model by observing random interactions. [42] learns a full autoencoder to give full output predictions and uses the latent state to get a linear dynamics model. Learning approaches are able to demonstrate results for simple

cases, but using random interactions as training data will likely scale poorly to truly complicated cases. [23] learns an approximation of a rope as a rigid link, which allows them to do a simple rope placing task with less data. However, significant amounts of data will be needed to learn representations that reason about intricate rope topologies, occlusions, dynamics, and rope types needed to tie a shoe.

Some have fused the ideas from knot theory and topology with additional layers for incorporating more physical properties of ropes so that robots can interact with them. [43] formulates a layered latent representation for soft objects. The lowest layer does the initial feature extraction of a real-world object. The middle layers perform compression and obtain more semantic feature information. The highest level contains the full semantic knowledge of the shape for use in shape planning. [40] similarly has a high level topological plan that is robot agnostic. In order to connect this to the real world, they learn topological motion primitives to execute the plan. [9] uses Reidemeister moves and Node deletion moves to formulate an algorithm for untangling dense knots. This had to be paired with a module that learns task relevant keypoints for manipulation in order to be successful on a real rope. They show that dense knot untangling can be completed without estimating or predicting the full state of a rope. In this thesis, we hypothesize that full prediction of the rope isn't necessary for good control while tying shoes, an idea we explore more in chapters 4 and 5.

### 1.1.2 Dealing with complex dynamics of a rope while tying

The difficulty of accurately predicting the dynamics of a hanging rope while it's being manipulated has led many to simplify the dynamics when tying knots in real life. [27] is able to get impressive results for robustly tying a variety of knots in simulation and real-life, but they were forced to make some modifications to the environment to make the dynamics easier to manage. They utilized external supports to hold parts of the rope in place. This allowed them to create motion plans where the arms don't make contact with the rope except for at the end, ensuring that most of the knot stays static while executing moves. [37] considers the concept of utilizing external

supports by theoretically analyzing bounds on the number of fingers and re-grasps needed to tie arbitrary knots. They find that 40 fingers would be sufficient to tie a shoelace knot. Instead of setting up an environment with 40 fingers to tie a knot statically, we're interested in finding clever and robust policies that will work with ordinary parallel grippers. We're interested in tying knots in-air without external supports, similar to how humans do it, which forces us to develop algorithms that don't assume stationary ropes or no contact.

### 1.1.3 Getting reliable perception for deformable objects

When manipulating a deformable object, perception is difficult because of the inevitable occlusions caused by the grippers over the object. The external support approaches in section 1.1.2 eliminate the need for perception at all because the supports prevent the rope from deviating from its expected position. [15, 21, 31, 42, 22, 41] offer promising results in controlling deformable objects through images directly, but they avoid the issue of dealing with occlusions by keeping the rope on the ground so that they can move the gripper away to give the camera a good view of the rope. This severely limits the complexity of possible tasks since the rope has to stay on the ground. [29] attempts to address occlusions by using MAP estimation of point clouds along with integrating information from a physics simulator to try to produce the best estimate of the location of a general deformable object. Even with a perfect point cloud representation of a shoelace knot, it is unclear how to utilize this to tie a shoe. Our approach to perception for the shoe tying task is to utilize simulation with ground truth perception to better understand what kind of perception is needed for control.

### 1.1.4 Our objective in relation to previous work

Although there is promising previous work in controlling deformable objects with a variety of approaches, we feel that none of these approaches are ready to be applied to the task of consistently tying any shoe. Much of the previous work looks at systems

that try to be extensible to a variety of rope tasks, such as designing controllers that can go from a start rope state to a goal rope state. They test on simple tasks such as pushing a rope into a configuration or tying simple knots on a table, hoping that their method will eventually extend to complex knots such as a shoelace knot. Instead, we take focus on tying shoes, a single extremely difficult task, and try to gain mastery over robustly tying a shoe with various initial conditions given a demonstration. This is similar to when a human first learns to tie a shoe, they might get a specific set of steps and hand motions to be successful, but they can then adapt this easily for any shoe. This is a task that we believe captures key problems in executing complex manipulation consistently in the real world. These include the representation of deformable objects for manipulation, reasoning about the high level of variability of objects in the world, the complexity of the dynamics of deformable objects, and the ability to transfer algorithms from simulation to real-life successfully.

# Chapter 2

# Simulation

A simulator for the shoe environment is a useful tool for making progress because it allows us to access simulator ground truths, collect massive amounts of training data, and quickly compare approaches without the physical hardware. A simulator with ground truth allows us to focus on the control and modeling of the shoe rather than the challenges of real-world perception of the shoelaces. A deterministic simulator also allows us to compare algorithms in a reproducible way.

Simulation of a shoe is a challenging task. The large number of contacts is computationally expensive to model and any realistic approximation of the object is extremely high dimensional. In this chapter, we discuss the simulation and modeling of an environment that allows a robot manipulator to interact with a shoe. Our primary design goals with this simulator are visual plausibility, simulation speed, and integration with robots.

## 2.1 Related work

Simulation of physically realistic rope-like objects have been used for biology and medical applications in [36], [6], and [4]. Many working on robotics applications such as [38, 41, 29] simulate a rope by approximating it as a chain of rigid links. [31] focuses on visually modeling a rope to generate depth training data. They approximate a rope as a chain of 50,000 vertices with a 12 control point Bezier curve as the underlying

representation. [3, 17] use the concept of diminishing rigidity to approximate the Jacobian of the deformable object for manipulation instead of relying on a direct model of the object. [36] develops a specialized physics-based thread simulator that is used in [27]. This simulator also relies on approximating a thread as a series of a finite number of nodes, but they additionally consider stretching, compressing, bending, and twisting of the thread. All of these simulators involve tradeoffs between being realistic and being fast. We will similarly work on a balance between having a realistic and fast simulator.

For future work, position-based dynamic simulators or GPU-based simulators could be a possible direction to explore. Nvidia FleX[16] is a particle-based simulation library that can give us real-time simulations of soft objects using the GPU. We found that it can support a real-time rope in a visually realistic way, but it currently lacks support for force-controlled actuators. [11] describes a way to use PBD to simulate a mooring line in real time. [39] combines Cosserat theory and position based dynamics to perform real-time simulation of a surgical thread.

## 2.2 Simulation of the grippers

We are hoping to eventually deploy these algorithms onto two real-life Kuka IIWA LBR arms with parallel Schunk grippers. However, for the purposes of simulation, we use just two floating parallel grippers in order to improve performance. The two floating grippers are controlled with two modules: A high-level velocity-limited setpoint controller and a lower level spatial force PID controller. The desired targets for the gripper are fed into the setpoint controller, which determines a point nearby in the direction of the target that should act as the actual goal for the PID spatial force controller. This ensures that the arm will move smoothly and in a controlled manner at all times regardless of the requested poses. Although this limits the ability of the arms to complete quick or variable speed maneuvers, we think that isn't important for robust shoe tying.

## 2.3 Model of the rope

We simulated the shoe as a MultibodyPlant in Drake[34]. This allows the simulation to be highly configurable, up to date with advances in simulation, and accessible for the wider robotics community. Although a lot of exciting work is being done in the space of realistic rope demonstration, we use a much more approximate rope model by modeling the rope as a chain of rigid bodies. This gives a stable and simple model to prototype algorithms with in simulation. The links in the rope are connected using a series of two revolute joints. We chose not to model the twisting or stretching involved since twisting/stretching motions are not essential to the main shoe tying task and they would make the simulation slower. On the other hand, friction and contacts are very important to tying shoes, which we can model with rigid bodies. Although we are using a less realistic and only visually plausible model, we hope that algorithms that are successful in simulation with this rigid body model can be adapted to work with a more realistic simulation and also in real life.

## 2.4 Shoe simulation parameters

The collision geometry of the shoe is approximated as a large cylinder and the shoelaces are modeled as two multi-link chains with 19 links each. There is a thin cylindrical collision element connecting the two shoelaces so it looks like one continuous rope. Although we modeled the rope as two separate ropes, it was more natural to observe the rope as if it were a single rope. Modeling the shoe with two separate laces allows us to easily vary the initial conditions of the two ropes independently.

Modeling our shoelaces to have the same mass as a realistic shoe would require the links in the rope to be really light. This large relative inertial scaling between the gripper and shoelaces translates to faster dynamics that the solver needs to resolve. Smaller timesteps are needed to resolve these dynamics, which means slower simulation. Since performance was a big design consideration, we made the mass of the rope hundreds of times heavier than a realistic shoelace. This allows us to get results

that look realistic while not needing a smaller simulator timestep than 0.0005s.

In order to maintain the visual plausibility of the simulator despite the heavy ropes, we tuned the grippers such that we would command as much force as needed to closely track the commanded positions. This meant having a significantly larger force limit than we would use in real life. Since maintaining friction while gripping a small and heavy object could be difficult for the simulator, we add blocks to the end of the grippers so the ropes don't slip as easily. This also provides space for the width of the rope to sit between the gripper fingers. These modifications shouldn't hurt transferability to the real world because in the real world, the shoelaces should be so light that we wouldn't have to worry about the robot arm getting weighed down by it. In addition, the gripping strength will be so much more powerful than any friction/gravitational forces that we shouldn't have to worry about the rope slipping out of the gripper.

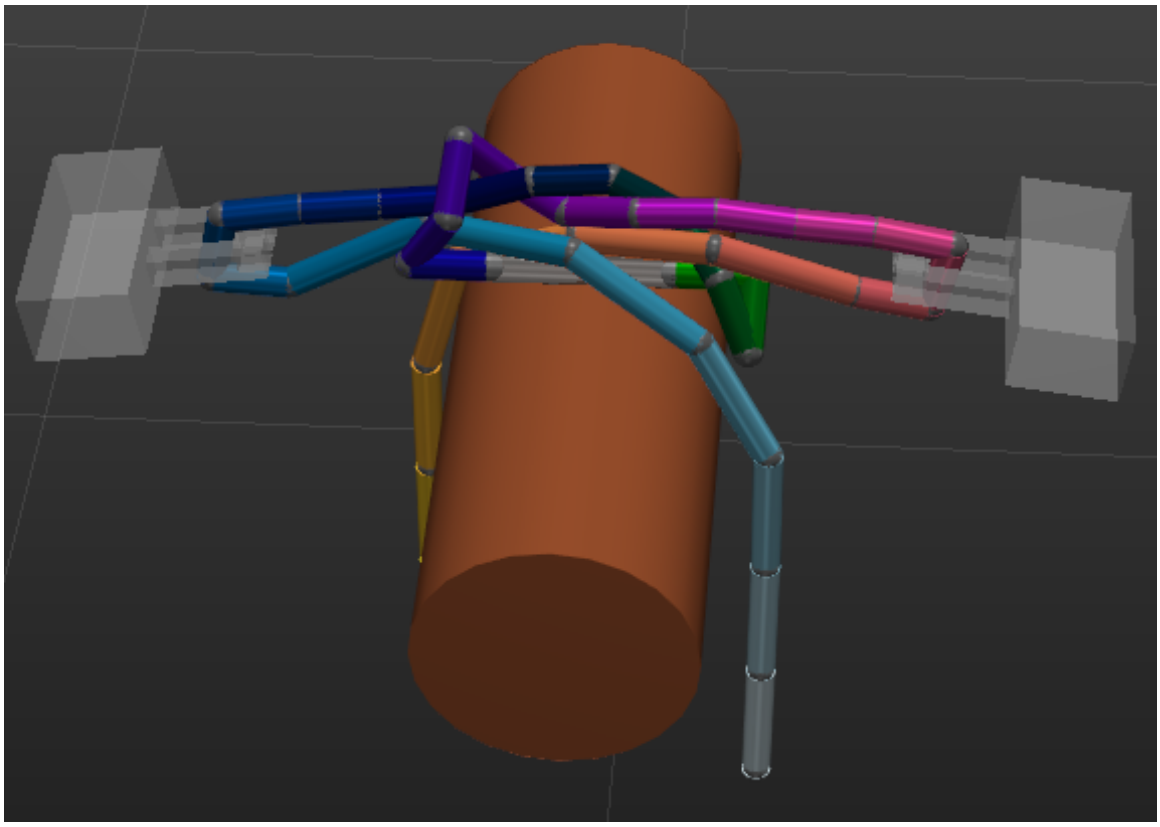

Figure 2-1: Modified gripper geometry holding a successfully completed knot in simulation

The joint damping was tuned to be as large as possible until the system showed unstable behavior. The value of the timestep was fixed at 0.0005s, which was tuned to work with the shoetying task while also being as fast as possible.

Table 2.1: Simulation parameters

| Parameter | Value |
|---|---|
| Rope Radius | 0.01 m |
| Joint damping | 0.001 Nms |
| Timestep | 0.0005 s |
| Penetration allowance | 0.0001 m |
| Stiction tolerance | 0.001 m/s |

### 2.4.1 Performance metrics

In this section, we go through the parameters needed for performant but visually plausible simulation. The overall goal in this section is to show which simulation parameters have the largest effect on performance and provide information about how long the simulator takes us to run. We're hoping this information might help others working with MultibodyPlant in Drake to understand how choose parameters to achieve faster simulation. These metrics reflect the speeds we were able to achieve with version 0.28 of Drake[34]. It's likely that additional tuning and newer versions of Drake will significantly speed up these times, but these times can at least act as a lower bound. It will also provide context about how long the experiments in this thesis took. Specifications of the computer are below:

Table 2.2: Computer specifications

| Part | Spec |
|---|---|
| CPU | Intel Core i9-9900X |
| RAM | 62G |
| GPU | 2 Nvidia GeForce RTX 2080 |

### 2.4.2 Number of rope links

Unsurprisingly, the number of rope links we model the rope with has a huge impact on the performance relative to real time. We graph the simulator rate for a stationary shoe setup with two hanging ropes while varying the number of links per rope and keeping the total rope length constant. Figure 2-2 shows that with this implementation, the simulator rate decreases as number of links increases. This shows that we should select the minimum number of links needed to make the task possible. In order to minimize number of links needed, we vary the size of the links. For the shoe, this means we use larger links at the end of the rope, where the exact bending behavior is less important and smaller links towards the middle of the rope where modeling the bending of the rope is more important. This can be seen in figure 2-1.

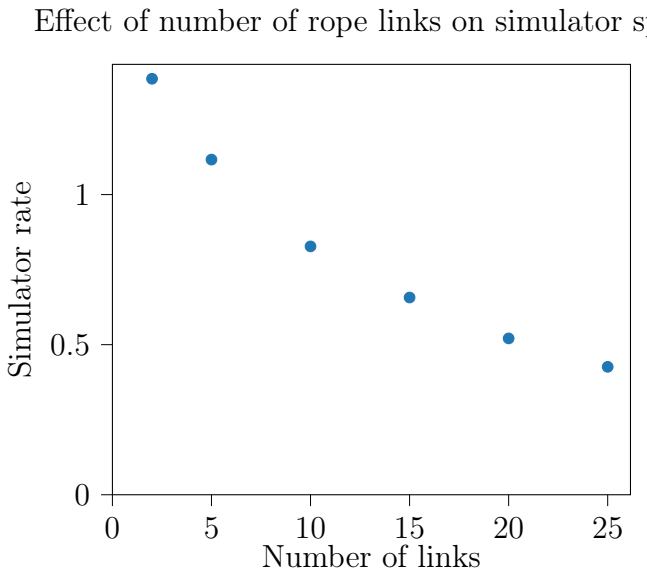


Figure 2-2: As we increase the number of links, the simulator rate generally decreases

### 2.4.3 Penetration Allowance

The MultibodyPlant in Drake uses a rigid contact model, which requires non-penetration constraints between the bodies. Drake enforces this using the penalty method. In doing so, we have to select a penetration allowance that gives a tradeoff between being

fast and being realistic. We measure the average simulator rate for a full shoe tie for various values of penetration allowances. We also note down if the shoe tie was still successful.

| Allowance(m) | Average simulator rate | Success |
|---|---|---|
| $10e^{-5}$ | 0.33 | Yes |
| $10e^{-4}$ | 0.35 | Yes |
| $10e^{-3}$ | 0.35 | Yes |
| $10e^{-2}$ | 0.34 | Yes |
| $10e^{-1}$ | 0.42 | No |

Table 2.3: Observations for various penetration allowances

We found that if the allowance is too large, the rope is able to easily pass through the gripper, resulting in a failed shoe tie and unphysical behavior. At the same time, when the allowance is decreased, the performance starts to get worse because the simulation equations are more stiff. We chose a value of 0.0001m to provide a tradeoff between speed and realistic behavior. Table 2.3 shows that the difference in simulator rate is fairly negligable between $10e^{-5}$m and $10e^{-2}$m so we chose this value somewhat arbitrarily. In general, we wanted to choose as small of a value as possible without significantly hurting performance. This is because we want a minimal amount of penetration between the grippers and the rope.

### 2.4.4 Stiction tolerance

Drake uses the viscous damping of tangential velocities as a continuous approximation of Coulomb's law of friction. We measure the average simulator rate for a full shoe tie as we vary the stiction tolerance while also noting if the shoe tie was successful. We chose a stiction tolerance of $0.001\frac{m}{s}$, because with smaller values, the timestep isn't small enough to resolve the dynamics. However, the results in table 2.4 also show that there is potential for improving the performance with increasing the tolerance and therefore allowing more slip with the rope.

Table 2.4: Observations for various stiction tolerances

| Tolerance | Average simulator rate | Success |
|---|---|---|
| $10e^{-5}$ | 0.44 | No |
| $10e^{-4}$ | 0.46 | No |
| $10e^{-3}$ | 0.35 | Yes |
| $10e^{-2}$ | 0.36 | Yes |
| $10e^{-1}$ | 0.38 | Yes |
| 1 | 0.40 | Yes |

### 2.4.5 Contacts

In this section, we analyze the effect of contacts on the simulator rate after tuning both penetration allowance and stiction tolerance. In order to show this relationship, we run a full shoetie and graph the relationship between simulator rate and number of contacts.

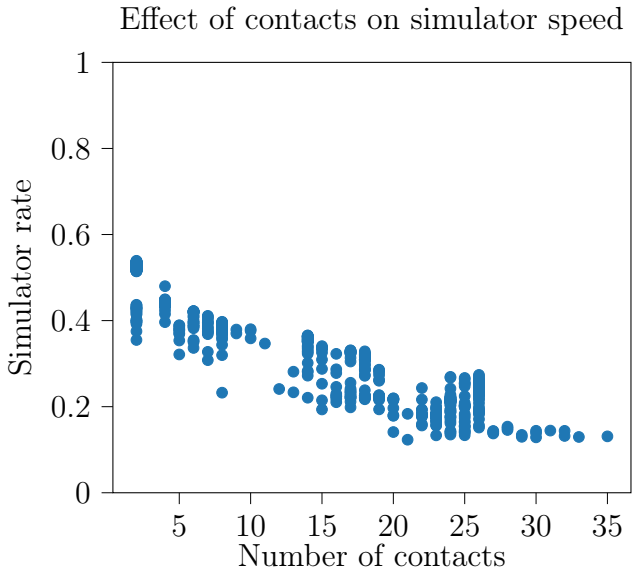


Figure 2-3: The presence of more contacts in the simulator results in a slower simulation rate

This graph shows that the number of contacts has a big impact on the rate of the simulator. As the number of contacts increases, the simulation rate decreases because the simulator has more contacts to resolve. We can also see that the shoe tying task produces a large range of contacts. At the beginning, there is very few contacts because the grippers aren't touching the ropes, but when the knot is tied,

we can get over 30 contacts as the rope collides with itself.

### 2.4.6 Parallelization

We used Ray[19] to run multiple shoe tying simulations in parallel. This allowed us to parallelize easily on a multicore single machine while also making it easy to scale on a full cluster in the future. We varied the amount of cpus and measured the wall clock time for 100 trials. We can get a factor of 10 times speedup on a machine with 10 cores and 20 threads. When we increase the parallelization of Ray to be larger than the number of cpus, our total computation time actually starts to decrease due to the increased parallelism overhead.

Table 2.5: Wall clock time to complete 100 episodes

| Number of cpus | Wall time(s) |
|---|---|
| 1 | 392 minutes |
| 10 | 45 minutes |
| 20 | 40 minutes |
| 100 | 52 minutes |

We haven't optimized the simulation or the parallelization much so there is likely a lot of possibility for speedup here. For example, these times could be dramatically improved by not rebuilding the full simulation diagram every time. Ideally, we would build the simulation once and then just make sure that the threads are using separate Drake contexts.

## 2.5 Conclusion

In this section, we described the design for a visually plausible approximate shoe simulator in Drake that allows direct interaction with robotic grippers. Although a real-life shoe has different dynamics than our approximated simulation, the simulation keeps the key complexities that exist in the rope problem, making it a reasonable proxy for a real environment. We focus on the exploration of shoe-tying in this simulation, leaving the problem of sim-to-real transfer as future work. The code for

this simulator is available in the format of an OpenAI environment so that other researchers can easily build upon our work [1]

[1]Source code: `https://github.com/RobotLocomotion/gym/tree/master/gym/envs/robot_locomotion_group/drake/shoe`.

# Chapter 3

# Human-generated baselines and parameterization

Our first goal was to produce a human-generated demonstration in order to show that the task is possible with the bimanual parallel gripper setup. This demonstration can be used as a reference or initialization for future methods. In addition to finding a way to tie shoes with this setup, we wanted to find simple yet flexible trajectory parameterization that allowed us to tie a shoe well. We designed two parameterizations: one that is open loop on the state of the rope and one that utilizes two positions of the rope to close the loop. The key idea for both parameterizations is expressing the trajectory in terms of higher level motion primitives.

One of the challenges with shoe tying is the long time horizon. It can take over 60 seconds to complete a shoe tie and if we run a controller at 10Hz, this results in a problem that is over 600 timesteps long. This will make the derivative-free direct policy search in chapter 4 difficult because we have to optimize over so many steps. It will also make model-based planning approaches like chapter 5 more difficult because we need to learn a model that can accurately predict hundreds of steps into the future. Therefore, in the section, we aim to create piecewise linear baselines that condense the whole shoe tie into as few moves as possible.

## 3.1 Related work

Many manipulation papers[22, 31, 42, 38] utilize higher level pick/place action parameterizations. For 2D rope pushing cases or even basic cloth folding examples, this is a suitable parameterization. Shoe tying requires a more flexible action space because we must stay in contact with the rope while tying because otherwise the shoelaces will fall. With the presence of gravity, we can't just place the rope in mid-air and expect it to stay. [12] manually specifies and teaches five in-air skill motions that the robot can perform: Grasping/Releasing, Double-grasping, Looping, Twisting, and Sliding. They then tell the robot which primitives to use when tying a knot. We also define motion primitives, but we aim to make our primitive class less specific to just ropes and then try to find policies with this flexible motion primitive class to tie shoes.

## 3.2 Open loop motion primitive

We define a sequence of open loop high-level motion primitives that are able to tie the shoe completely independent of the position of the rope. Each motion primitive contains 8 parameters with 4 per arm. We define $W$ as the world frame, $G$ as the gripper frame before the move, and $G'$ as the desired gripper frame. For each arm, the parameters are as follows for moves $n \in 1, 2, 3..12$

- ${}^{W}p_{G}^{G'}[n] \in \mathbb{R}^3$: desired displacement from current gripper position in meters
- $d[n] \in \mathbb{R}$: desired displacement in gripper width in meters

This parameterization commands the grippers to fixed points in the world regardless of the state of the rope. We keep the angle of the grippers constant to decrease the size of the action space. We rely on the lower level gripper controller described in section 2.2 to perform this motion at a consistent and slow velocity in a straight line.

## 3.3 Closed loop motion primitive

We also define a trajectory parameterization that uses two points on the shoelace as frames of reference for the gripper. Each motion primitive contains 10 parameters, 5 per arm. We define $W$ as the world frame, $G'$ as the desired gripper frame, and $R$ as the point on the rope that the corresponding gripper tracks. For each arm, the parameters are as follows for moves $n \in 1, 2, 3..14$:

- ${}^{W}p_{R}^{G'}[n] \in \mathbb{R}^3$: desired displacement of gripper from rope frame in meters
- $d \in \mathbb{R}$: desired displacement of gripper width in meters
- $p \in \mathbb{R}$: distance from the start of the rope to frame R in meters

The frames can be visualized in figures 3-1. We can see that the closed loop motion primitive tells the left gripper to align itself with a point at frame R on the opposite rope. We are able to command the gripper based on the position of just two keypoints on the rope. This shows that successful shoe ties are possible even if we use a minimal amount of the rope state. Despite the high dimensionality of the rope, it turns out that much of this information isn't useful with shoe tying.

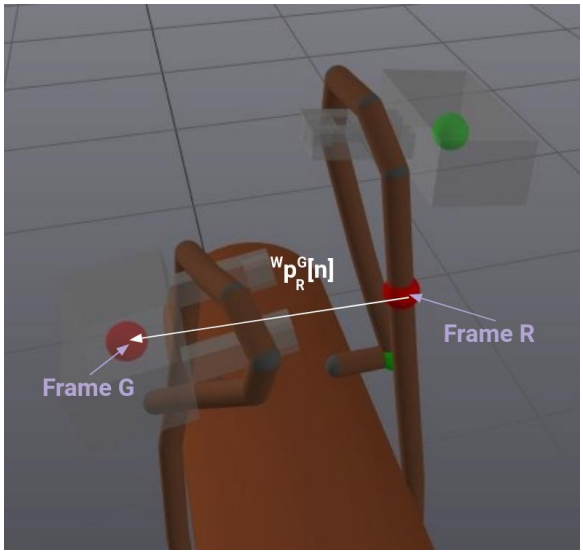


Figure 3-1: We pick a point on the rope to track an offset from that point.

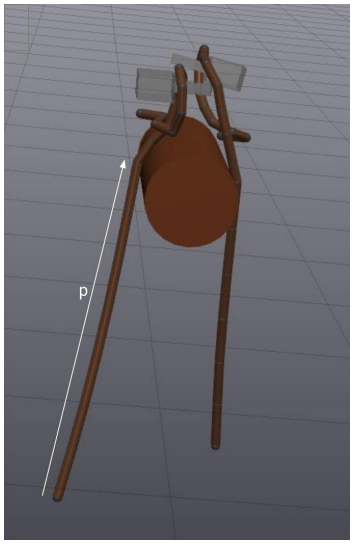


Figure 3-2: Rope frame R is p meters along the rope

## 3.4 Varying initial conditions of the rope

In order to measure the robustness of a policy over initial conditions, we generate 100 random initial rope conditions and count the number of successes. Throughout this thesis, we use the same 100 random rope initial conditions to benchmark all of the policies. Random rope initial conditions are generated by perturbing the initial condition of the rope according to $U \sim (-0.02m, 0.02m)$ for the x, y, z directions for both grippers independently and $U \sim (-0.2rad, 0.2rad)$ for the roll pitch and yaw for both grippers independently.

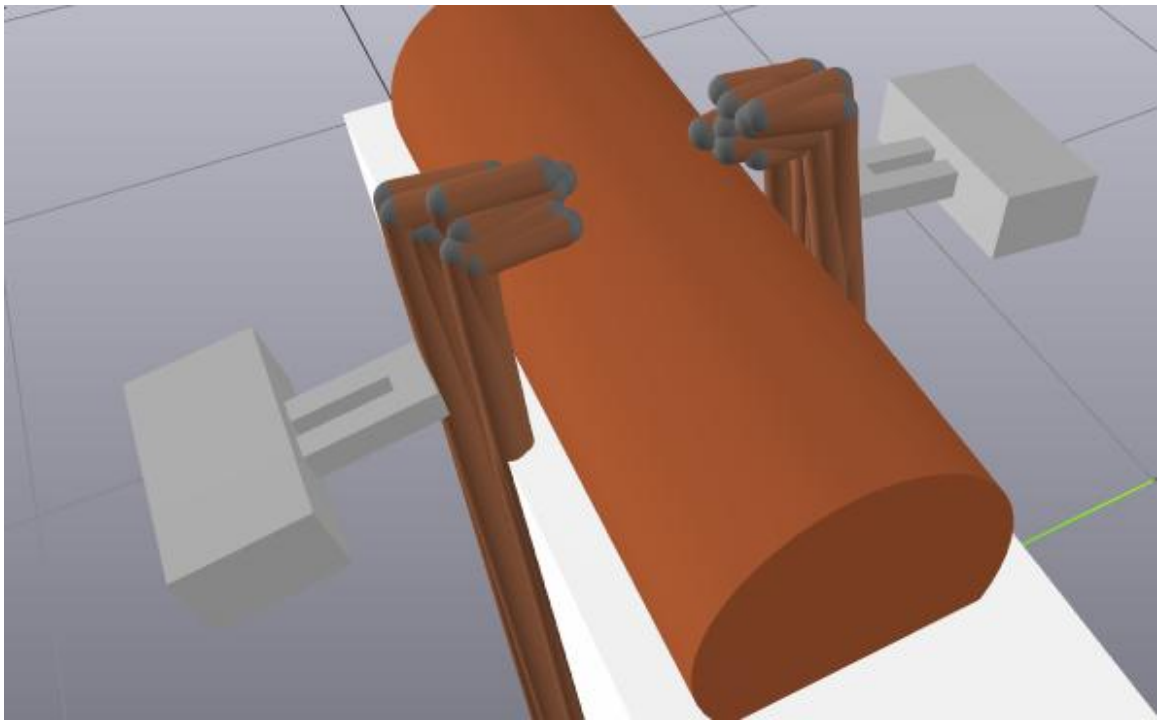

Figure 3-3: Varying the initial conditions of the rope.

## 3.5 Experiments

In this section, we evaluate the success of the open loop and closed loop baselines under perturbations of initial conditions as shown in figure 3-3 in order to estimate the robustness of the baselines. This is meant to give us a point of comparison for

the success of future methods.

Table 3.1: Number of success out of 100 for various models

| Parameterization | Value |
|---|---|
| Open loop | 38 |
| Closed loop | 77 |

The results show the initial closed loop trajectory beating the initial open loop trajectory. Although we can't say that this makes the closed loop parameterization better from just 1 sample, it does make sense that the closed loop trajectory is more robust to initial rope configuration. Figures 3-4 and 3-5 show the open loop trajectory will not at all be able to adjust if the rope is moved. Figures 3-6 and 3-7 show that the closed loop trajectory uses the position of rope keypoints to define the waypoints, so it will adjust if the rope is moved.

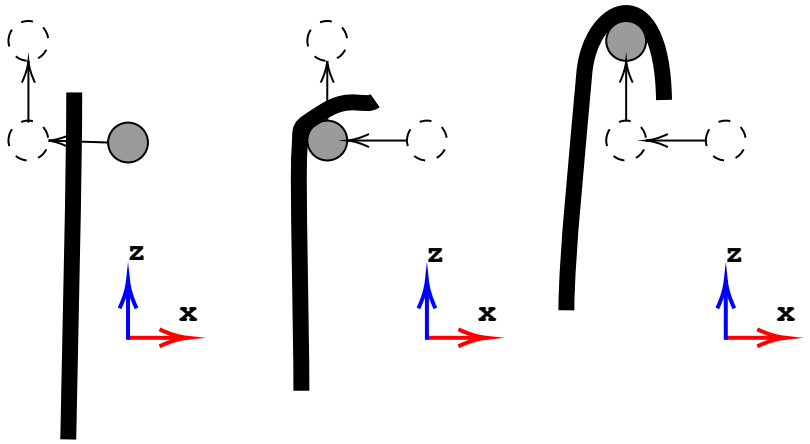


Figure 3-4: 2d example of a gripper successfully lifting the rope using an open loop trajectory

## 3.6 Discussion

In this section, we show that it is possible to encode the class of shoe-tie policies using just a sequence of 10-20 motion primitives with 8-10 parameters each. This makes it feasible to utilize CMA-ES to optimize these actions in the next chapter.

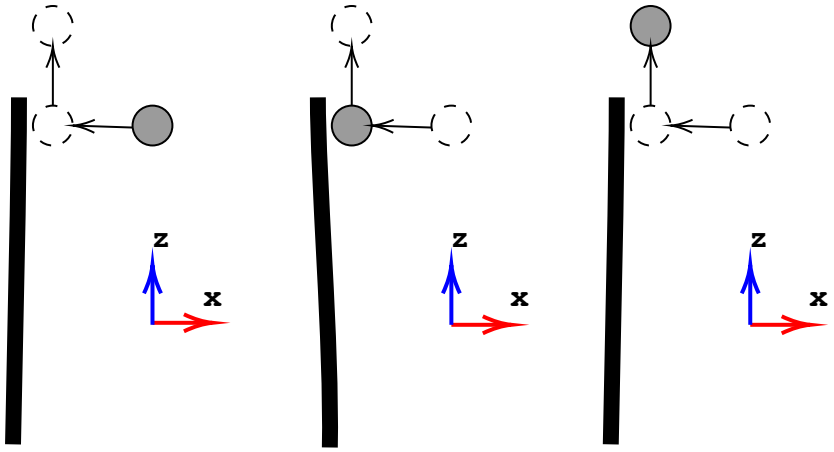


Figure 3-5: 2d example of a gripper failing with the same trajectory because we moved the rope.

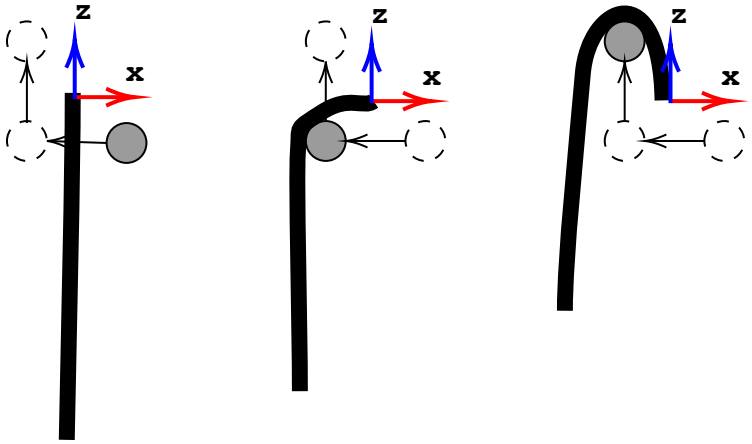


Figure 3-6: 2d example of a gripper successfully lifting the rope using a closed loop trajectory around the base of the rope

By including the observation of two rope keypoints, the closed loop parameterization makes achieving robustness to initial conditions more natural than the open loop parameterization. In chapter 4 and 5, we explore ways to make these parameterizations more robust to initial conditions.

For future work, it would be interesting to test out the parameterizations on hardware. The open loop parameterization is the most straightforward to implement on hardware since we just need to ensure the setup and scale of real-life matches the simulator that the baseline was created with. The closed loop parameterization

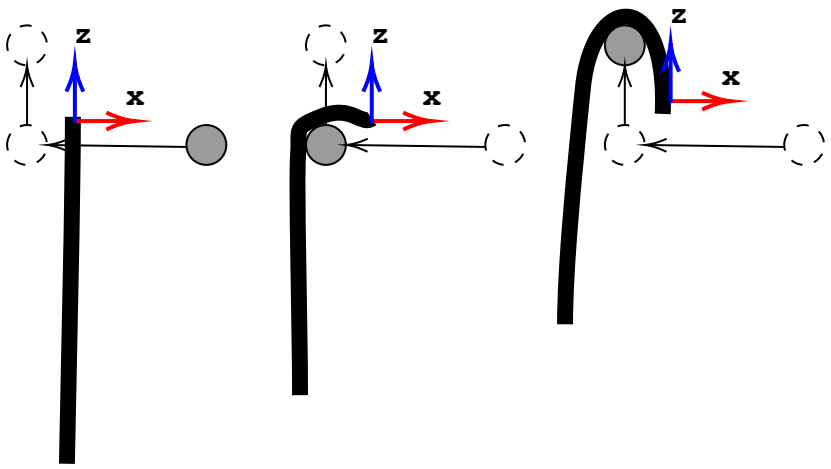


Figure 3-7: 2d example of a gripper successfully lifting the rope using the same trajectory even when the rope moved

requires us to be able to accurately query the pose of a point on the rope given how far away from the start of the rope the keypoint should be regardless of if it's occluded. Dense descriptors[31, 5] would be able to give us this information, but it's unclear how effective dense descriptors would be for shoelaces due to the heavy amount of occlusions that are inevitable. Since we're only analyzing simulation results right now, we are able to query the ground truth exactly from the simulator.

# Chapter 4

# Optimizing for robustness with CMA-ES

In this section, we optimize the baselines in the previous chapter for robustness over the initial conditions of the rope. We used CMA-ES, which gives us the ability to heavily parallelize to optimize noisy and non-convex function. For shoe-tying, CMA-ES allows us to directly optimize for the policy in the simulator without having to model the complexity of state and dynamics representations for deformable objects. Our contribution in this chapter is applying CMA-ES to optimize the policy of a complex manipulation task for robustness.

## 4.1 Related work

Previous works in deformable object manipulation show surprisingly impressive results with just open loop. [18], a mechanical engineering senior design project, suggests that there exists a robust open loop policy to tie a shoe that doesn't rely on any perception at all. It is still able to tie the knot with a high success rate. [27] also shows that robust open loop policies for consistently tying knots is possible. In this thesis, we want to build on that work in order to show that we can optimize a given trajectory for robustness over initial conditions. [20] Shows an example of manipulating a deformable whip in open loop. Similarly to the task of tying a complex

knot, getting full and accurate perception of a high-speed whip is neither practical nor necessary. Despite the complexity of whip mechanics, they are able to get a good open loop policy in simulation by designing a motion primitive with five parameters and then optimizing over the parameters. We try to find a robust open loop policy for tying shoes, which has proven to be more challenging than manipulating a whip because it requires us to make and break contact over a longer time horizon.

CMA-ES has been used on various humanoid problems and RL benchmarks, but far fewer people have considered using direct policy search for complex manipulation tasks. [28] suggests evolution strategies as a scalable alternative to reinforcement learning for classic RL benchmarks. By scaling CMA-ES on over 1000 parallel workers, they are able to solve 3D humanoid walking in 10 minutes. In [32], evolutionary strategies were utilized to allow a biped robot to walk from scratch. [14] uses CMA-ES to optimize a kicking motion for a Nao robot. Most of the related work of using CMA-ES in robotics only studies the optimization of trajectories that don't use perception. We designed a closed loop motion primitive where the gripper tracks the state of 2 keypoints on the ropes.

## 4.2 Formulating a fitness function

We define a fitness function for CMA-ES by using distance from a set of demonstrations. We produce a set of good demonstrations $\mathcal{D}$ by randomly perturbing the open loop benchmark in chapter 3 for different initial conditions and randomly select successful ones. To compute the cost for an episode, we sum the corresponding demonstration costs

$$\sum_{1}^{T} \bar{c}_t(y_t) = min_{y_t \in \mathcal{D}} ||y_t - \bar{y}_t||_2^2 \tag{4.1}$$

where $y_t$ is a vectorized state of the shoe including $y_{rope}$ and $y_{gripper}$ as defined in section 5.2.1. Using corresponding costs allows us to get a cost along the way while tying a shoe. Scoring over a demonstration allows us to consider the diversity in possible solutions, ensure that there is representation from various initial conditions

in the demonstration set, and help make sure that good episodes have low costs. A flaw with the corresponding score cost formulation is that the inclusion of time would penalize the robot for tying a shoe at different speeds from the demonstration. Another flaw is that computing the difference to a demonstration using the Euclidean distance between them has many failure modes. The numerical scores we get don't always line up with our intuition of success for the task. In figure 4-2, the fact that the rope isn't wrapped over both sides of the left gripper means that it won't be possible to finish the shoe tie without starting over. However, the Euclidean distance of this scenario is still low.

Figure 4-1: Demonstration rope state

Figure 4-2: Example of bad state

In the future, coming up with a distance metric that is more topological or task-specific would likely help address these failure cases. We want our cost function to pay more attention to higher level structures in the shoe system than coordinates of the bodies, like location of shoelace crossings and the way that the gripper is interacting with the rope. A way to improve on the Euclidean distance score is to use a SVM classifier to attempt to classify overall task success given a $y_t$ position. We can assign a constant penalty to any trajectory that doesn't produce a successful result.

## 4.3 Choosing CMA-ES hyperparameters

We used the CMA-ES wrapper out of Nevergrad [26] to perform a derivative-free direct policy search. In using this implementation, there were several parameters that needed to be tuned.

- $\lambda$: Offspring population size: A large population size will result in slower convergence but decreases the chance of convergence to a local minima. [2] suggests restarting CMA-ES with increasing population sizes to obtain a more global search. However, since we aren't trying to solve the global problem, we fix this parameter to 20, the number of simulations we can run in parallel.

- $\sigma$: This describes the standard deviation of CMA-ES. If this value is too small, we might not be able to find the optimal solution because we're only sampling very close to the initial demonstration. If it's too large, the optimization will take a significantly longer time because we are doing a more global search. We found that a value of 0.02. provides a good tradeoff between the two.

- Initialization: We found that initializing the search with a good solution was crucial due to the difficulty of figuring out how to tie a shoe without a demonstration. There's also a large amount of local minima that are possible. We initialized the optimization with the baselines from chapter 3.

- Elitist mode: With elitist selection, we select the all-time best $\mu$ individuals. Without elitist mode, we only select the $\mu$ best from the current generation, essentially forgetting about previous generations. Elitist mode is prone to local minima but results in faster convergence. We chose to use elitist mode since we are less concerned about local minima since we start with a good solution.

- Termination: The fitness function for shoe tying is noisy and ill-conditioned which means it will take a long time for CMA-ES to converge, especially given the speed of the simulator. A small change in the action trajectory can result in a large difference it output because it could mean the difference between

dropping the shoelaces and holding the shoelaces. Evaluating a trajectory on random initial conditions of the rope is noisy. Therefore, we terminate the algorithm after 1000 iterations and analyze which methods were better at converging towards robust solutions in that time.

## 4.4 Experiments

In this section, we focus on fully understanding CMA-ES's behavior on just the first portion of the shoetie, which we define as stage 1 as shown in figures 4-3 and 4-4.

Figure 4-3: Starting position

Figure 4-4: Goal position for stage 1

### 4.4.1 Sampling schemes for ranking

In each generation of CMA-ES, we generate $\lambda$ offspring and our objective is to select the $\mu$ best. In our case, this means out of 20 possible gripper trajectories, we want to be able to rank them in terms of robustness. In order to evaluate each of the individuals, we evaluate it on $n$ random initial conditions and add their fitness values. We then choose the $\mu$ lowest fitness values, representing the trajectories that produce shoe-gripper states closest to the demonstration.

In order to evaluate a policy on robustness for the purpose of ranking, we need

to evaluate it on multiple random samples. However, to get an accurate robustness measure, we need to sample on a lot of initial conditions. In chapter 2, we showed how expensive the simulator is to evaluate a trajectory on. Another possible way around sampling is to use a fixed training set rather than randomly generating initial conditions every generation. Our function will be less noisy, but we won't be optimizing over the whole distribution of the initial conditions of the shoelace so the solution will likely be less robust.

In this section, we analyze the effect of $n$ on the success of the CMA-ES algorithm where $n$ is the number of initial conditions we evaluate each trajectory on to measure robustness. We generate the random rope initial conditions the same way we did in section 3.4. We ran the CMA-ES algorithm on stage 1 of the shoe tying task as defined in figures 4-3 and 4-4 for the open loop parameterization. We vary the number of samples $n$ and measure the eventual robustness of the optimal policy by evaluating how many successes we get out of 100 random initial conditions. We ran CMA-ES for a fixed 1000 iterations.

Table 4.1: CMA Results for various sample sizes

| Number of samples | Success out of 100 on the task |
|---|---|
| 16 | 100 |
| 8 | 98 |
| 4 | 91 |
| 1 | 92 |

This shows that sampling more helps CMA-ES find more robust solutions, at a huge performance cost. Sampling 8 times will take around 8 times as long as sampling once, but this only gives us success on 6 more tasks. When we don't sample enough, we end up frequently mis-ranking trajectories, which makes it significantly more difficult for CMA-ES to converge to a robust trajectory. [25] provides a survey of various ways of ranking policies in noisy optimization problems, such as ours where we evaluate our trajectories on random samples out of a distribution. A key idea is that we should ideally avoid sampling a lot for trajectories that we already know are bad and instead focus on sampling the better trajectories in order to truly distinguish which are better.

Another possible idea is sampling less in early generations when we are "exploring" and sampling more in the later generations when the goal is refinement. For future work, we'd like to explore these ideas to give us robustness while also conserving the number of evaluations of the fitness function needed.

### 4.4.2 Comparing open loop and closed loop parameterizations

We compare the performance of optimizing the open loop and closed loop trajectories using 4 random initial conditions to evaluate each sample.

Table 4.2: Number of task successes out of 100

| Parameterization | Baseline | Optimized |
|---|---|---|
| Open loop | 46 | 91 |
| Closed loop | 90 | 99 |

These results show that CMA-ES was successful in optimizing both trajectory parameterizations. The open loop parameterization had a larger improvement because it started with signficantly worse initial condition. The closed loop parameterization was initialized with a good baseline, but it was able to further improve on this baseline. Both of these parameterizations are both viable approaches that should be tested on real robots. For future work, this experiment should be run on the entire shoe tie for signficantly more iterations. A larger number of random of initial conditions than 4 should also considered for each trajectory.

## 4.5 Conclusion

In this section, we used CMA-ES to optimize a given baseline trajectory of the first portion of the shoe tie for robustness. We found that increasing the number of samples we evaluate each trajectory on dramatically increases the total number of rollouts we have to do, but it makes CMA-ES converge to a more robust solution. In the future, we hope to extend this to the entire shoe tie and also test the resulting policies on hardware. In addition, we hope to vary the shoes over various dynamics

parameters like friction and damping in addition to optimizing over initial conditions of the rope. In [32], CMA-ES results in simulation failed to work in real life because of slight differences in real life and in the simulator. Our hope is that by optimizing for robustness in the simulator, our policy will transfer more successfully to the real world.

Another important direction for future work is to explore alternative trajectory parameterizations. Finding a parameterization that utilizes haptic data could allow us to formulate an even more robust parameterization. When humans tie shoes, we rely heavily on how the shoelaces feel in our fingers rather than trying to estimate the true position of the rope visually. Many would find tying a shoe with eyes closed to be easier than tying a shoe with gloves such that it's impossible to feel forces and contacts.

# Chapter 5

# Learning Approximate Information State

In this chapter, we explore the feasibility of learning models for shoe-tying that are useful for control. Similarly to the CMA-ES section, our goal here is to stabilize a demonstration to be robust over the initial conditions of the shoelaces. We focus on giving an analysis on the success of this method on the first stage of the shoetie. In this section, we learn a neural network to reduce observations to a latent approximate information state along with a quadratic cost network and a linear dynamics model on the approximate latent state. Then, we utilize these networks to perform Linear MPC.

## 5.1 Related work

[30] defines approximation information state(AIS) as a function of history that is learned from data. They show that if AIS is sufficient to approximately predict the reward and its own dynamics, solving a dynamic program on AIS can approximately match the solution to the original optimal control problem. They learn a policy online in order to simultaneously improve the representation and the policy. Our contribution is modifying their method of producing AIS and the corresponding optimal control for a realistic robotic manipulation problem.

Our formulation is similar to that of [10], where they learn latent spaces for planning through reward prediction. However, we constrain the dynamics to be linear and the rewards to be quadratic, so the resulting optimal control problem is convex.

Our approaches differs from other autoencoder approaches from deformable objects like [42], [24] because we don't require our latent representation to be able to reconstruct the whole output of the rope. Since our goal is to get a latent state observation that is good for control, we only require our latent representation to be able to predict the next latent state and the current cost, which enables the prediction of cost trajectories given an input trajectory.

## 5.2 Defining the system

We have a shoe system with discrete time nonlinear dynamics $x_{t+1} = f(x_t, u_t)$ and $y_t = g(x_t)$ is ground truth observations from the simulator and $u_t$ is the commanded action. The dynamics function $f$ of a shoe is approximated using the simulator in chapter 3, making it expensive to reason about for optimal control. The non-smoothness from contacts and the nonlinear constraints lead to non-convex problems. We treat this simulator a black-box system. The goal is to find the optimal control $u_t$ that is able to tie the shoe while being robust to the initial conditions of the rope.

### 5.2.1 Observation representation

The observation $y \in \mathbb{R}^{183}$ is a concatenation of the following ground truth information from the simulator $(y_{rope}, y_{gripper})$

- $y_{rope} \in \mathbb{R}^{165}$: rope configuration, parameterized by 3D positions of keypoints every 0.05cm along the rope
- $y_{gripper} \in \mathbb{R}^{18}$: gripper rotations and translations for both arms

The original AIS paper used history of observations where we just use a singular observation. This is because we do not believe that history is useful when tying shoes. Given a still frame of the ground truth states of the shoe, it is clear to a

human what the next action to take is, no matter what the velocities on the rope or gripper were previously.

### 5.2.2 Action representation

The action space $u$ is the open-loop high-level action primitive discussed in chapter 3. It contains the changes in x, y, z, and gripper width for both arms, making $u \in \mathbb{R}^8$. It's possible that alternate action representations such as the keypoint-based closed loop parameterization could provide improvements on our work so far. Similarly to chapter 3, we limit the horizon of the task by choosing actions that are as sparse as possible. Although this limits the richness of the trajectory class, it significantly shortens the task horizon. We showed in chapter 3 that the shoelace task can be broken down into a few piecewise linear moves. Figure 5-1 shows what a gripper trajectory might look like with a dense and sparse sampling scheme. Although the dense sampling will look more fluid when deployed on a robot and will have trajectories that perform better, this is not strictly necessary to achieving highly consistent shoe ties.

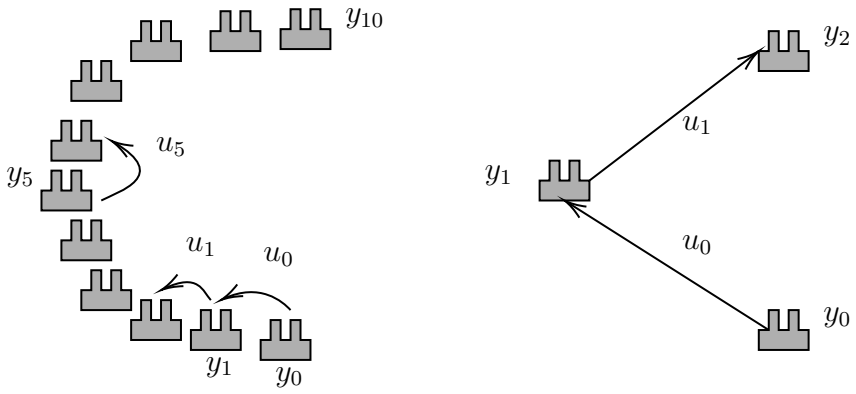


Figure 5-1: Left: Sampling densely, Right: Sampling sparsely

### 5.2.3 Cost formulation

A key idea with learning AIS is that instead of requiring the network to be able to predict the full output like an autoencoder, we only require it to be able to approximate the original cost of the full system. This cost can then be used for approximate

optimal control. In this section, we discuss the cost formulation for the cost we aim to predict with the AIS networks. We use a cost defined on the full observation from the simulator. This cost from a set of demonstrations $\mathcal{D}$ at timestep t is defined as

$$\bar{c}_t(y_t) = \min_{y_t \in \mathcal{D}} ||y_t - \bar{y}_t||_2^2 \tag{5.1}$$

In the next section, we discuss how this cost is used.

## 5.3 Generating AIS

We do system identification to approximate a linear dynamics model and a quadratic rewards model on a latent state $z_t$. We learn an MLP network for $\sigma$ to transform the raw observations $y_t$ from the simulator into a latent space $z_t$. We also learn a cost network $\hat{c}$ with a quadratic structure that predicts a scalar representing how bad the state represented by $z_t$ is. We also learn a dynamics network with a linear structure to take a latent state and action and produce the next latent state. We formulate a cost function $\bar{c}$ in section 5.2.3 that is not action-dependent, which allows us to remove $u_t$ as a parameter to the cost function we learn. We learn a quadratic cost function since it should be estimating a distance from a demonstration. With $A_{100\times100}$, $B_{100\times8}$, and $C_{100\times100}$ as matrices, these are the networks we are trying to learn

$$z_t = \sigma(y_t, t) \tag{5.2}$$

$$z_{t+1} = \psi(z_t, u_t) = Az_t + Bu_t \tag{5.3}$$

$$\tilde{c}_t = \hat{c}(z_t) = z_k^T C z_k \tag{5.4}$$

Similar to the idea of the Koopman operator, $\sigma$ lifts the state space into a space where we can construct linear models of the nonlinear system. The original AIS paper did not restrict $\psi$ to be linear and $\hat{c}$ to be quadratic. However, doing so yields tractable optimization problems when using MPC at planning time. This allows us to learn AIS on an offline dataset and then use the AIS to inform an optimal plan. The inclusion

of time as a parameter to $\sigma$ is a requirement due to our cost formulation of exactly replicating the demonstration trajectory rather than a requirement inherent to the shoe-tying problem.

## 5.4 Dataset generation

We learn the model using an offline dataset. This is generated by perturbing the initial condition of the rope according to $U \sim (-0.02m, 0.02m)$ for the x, y, z directions for both grippers independently and $U \sim (-0.2rad, 0.2rad)$ for the roll pitch and yaw for both grippers independently. Two data augmentation techniques were used. In one, we apply a translational offset to the whole scene so that the model learns the dependence between the position of the rope and gripper rather than the gripper's absolute location in the world. We also apply a 0.001m perturbation to the actions in order to account for noise that will occur in the action. We will analyze the effect of the action augmentation in section 5.7.2.

## 5.5 Loss functions

Given an initial observation $y_1$ and a sequence of actions $u_1, u_2, \ldots$ from the dataset, we can repeatedly apply the $\sigma$, $\psi$, and $\hat{c}$ networks to approximate the costs that would occur.

$$\tilde{z_n} = \sigma(y_n) \tag{5.5}$$

$$\tilde{c_n} = \hat{c}(z_n) \tag{5.6}$$

$$\tilde{z}_{n+1} = \psi(\tilde{z}_n, u_n) \tag{5.7}$$

$$\tilde{c}_{n+1} = \hat{c}(\tilde{z}_{n+1}) \tag{5.8}$$

$$\vdots \tag{5.9}$$

We train on an n-step total loss as defined below

$$\mathcal{L}_{cost} = \frac{1}{n}\sum_{i=1}^{n}(\bar{c}_i - \tilde{c}_i)^2 \tag{5.10}$$

$$\mathcal{L}_{dynamics} = \frac{1}{n}\sum_{i=1}^{n}||\tilde{z}_{i+1} - \sigma(y_{i+1})||^2 \tag{5.11}$$

$$\mathcal{L}_{total} = \mathcal{L}_{cost} + \lambda\mathcal{L}_{dynamics} \tag{5.12}$$

## 5.6 Control

We utilize the learned observation compression $\sigma$, linear dynamics $\psi$, and quadratic rewards $\hat{c}$ to run MPC over a fixed task length of T. At each timestep, we formulate and solve an optimization using Mathematical Program in Drake, which automatically identifies the optimization as a QP. We use the SNOPT solver[7, 8].

Given a task length T and a horizon of H, to run non-receding horizon MPC, we run the following optimization from t to $h = min(T, t+H)$ at every timestep t. After solving the optimization, we execute the first action $u_1$.

$$\begin{aligned}
\underset{u_1,\ldots,u_h,z_1,\ldots,z_{h+1}}{\text{minimize}} \quad & \sum_{i=2}^{h}\hat{c}(z_i) \\
\text{subject to} \quad & z_{i+1} = \psi(z_i, u_i), \quad i = 1,...h, \\
& z_1 = \sigma(y_t), \\
& |\bar{u}_i - u_i| \leq 0.01, \qquad i = 1,...h, \\
& |u_i| \leq 0.3, \qquad i = 1,...h
\end{aligned} \tag{5.13}$$

The first constraint is a dynamics constraint to ensure we're following our learned dynamics model $\psi$. The second constraint is an initial condition constraint to initialize $z_1$ based on the current observation from the simulator. In order to stay in the domain of our training set, we impose a constraint that actions have to be within 0.01m from the demonstration trajectory $\bar{u}$. We also constrain the size of the actions to be less than 0.3.

## 5.7 Experiments

Similarly to the CMA-ES section, we focus on gaining a good understanding of just the first stage of the shoe tie as shown in figures 4-3 and 4-4. Hyperparameters and architectures are described in Appendix A.

### 5.7.1 Number of rollouts to train on

An essential training parameter to tune is the number of rollouts to train on, or $n$ in the training loss defined in section 5.5. We expect that a 1-step loss, or equation error, provides an easier function for the model to approximate. However, any bias in this 1-step error may explode when we use this dynamics model to predict many steps into the future, which can have a big negative impact on control. [1] suggests that training on multi-step simulation error is preferable to training on single-step equation error because it can eliminate bias. In order to see if these expections align with our problem, we trained a model trained on 1-step loss and a model training of 7-step loss, which is also the size of the training episodes and then observe losses over the n-step losses for varying values of n. We measured the n-step errors on a test set of 100 episodes that are generated with the same distribution as the training set, but are completely unseen in training. The n-step losses are computed from the formulas in 5.5. In 5-2, we compare the two extremes of training on 1 step and over the whole task horizon of 7 steps.

### 5.7.2 Task success results

In order to evaluate which was better for real-world performance, we compared 1-step MPC for the model trained on equation error with 7-step MPC for the model trained on simulation error. Our goal here is to see whether a short horizon accurate model or a long horizon approximate model is better for control.

We ran the MPC policy described in Section 5.6 for 100 random initial conditions(generated the same way as the training set but not from the training set) for both models from the previous section. We also count success of when the model is

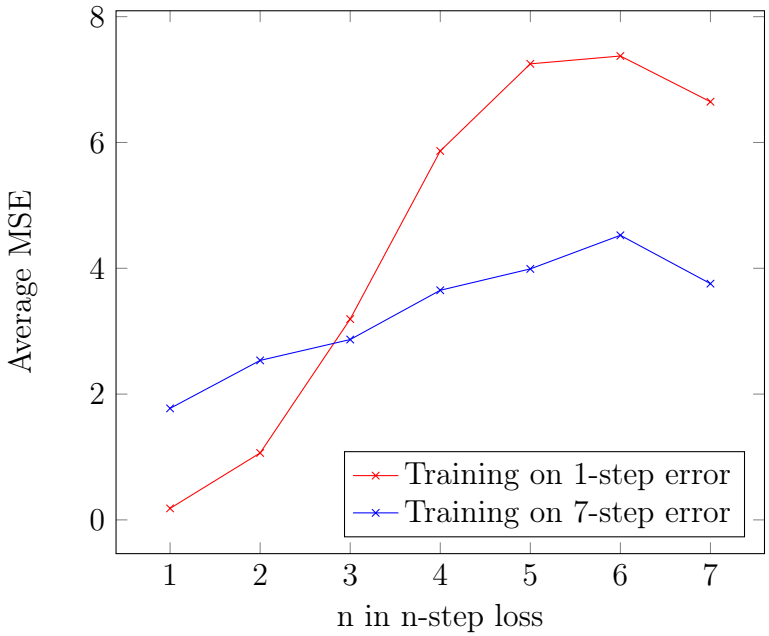


Figure 5-2: When trained on 1-step error, we get small errors on that horizon but the errors blow up past that. When trained on 7-step error, the errors are larger for the small horizons, but the errors don't drift as much as the horizon size increases

trained with and without the action augmentation.

Table 5.1: Number of success out of 100 for various models

| | |
|---|---|
| Open loop baseline | 44 |
| 1-step loss model without augmentation | 50 |
| 1-step loss model with augmentation | 55 |
| 7-step loss model without augmentation | 91 |
| 7-step loss model with augmentation | 98 |

It's clear that applying the action augmentation improves results on all rollouts. Table 5.1 also tells us that training a model on 7-step loss and deploying it on MPC with a horizon of 7 yields a significantly larger success rate than training a model on 1-step loss and deploying it on MPC with a horizon of 1.

### 5.7.3 Analyzing the AIS model

One of the major downsides of the AIS approach is that it lacks interpretability. In this section, we will visualize and analyze the behavior of the best-performing learned AIS model to attempt to understand what it learned.

### Stabilizing the cost for various initial conditions

We plot the costs from both the open loop and AIS policy over time for the random initial rope conditions in our test set. We use the 7-step loss model with augmentation from the previous section.

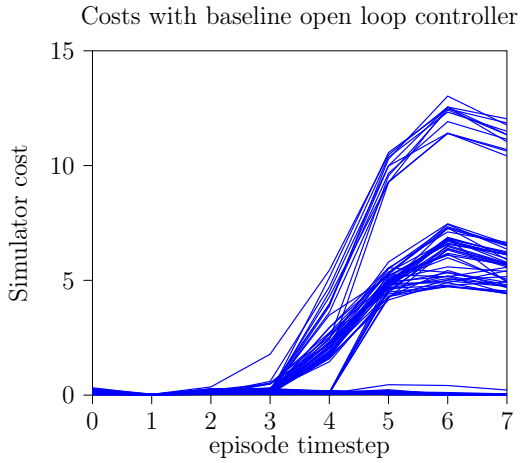


Figure 5-3: Open loop baseline diverges over initial many initial conditions

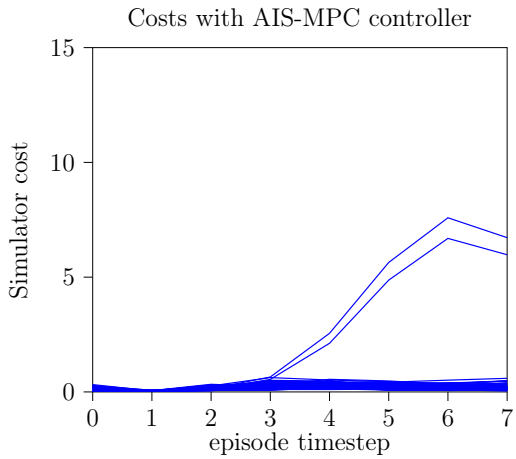


Figure 5-4: AIS-based MPC converges on 98/100 of the episodes.

When running the same open loop baseline for 100 initial conditions, we can see in figure 5-3 that many of the episodes diverge from having a cost near 0. When running AIS-based MPC controller for 100 initial conditions, we can see in figure 5-4 that most of the episodes converge to having a cost near zero. Therefore, we see that AIS is generally able to adjust trajectories to stabilize to a demonstration even when initial conditions change.

### Analyzing prediction error of AIS

In this section, we aim to understand more about how an AIS that has such significant prediction error can still be successful for optimal control. In order to do this, we select a single random initial condition to analyze. From that initial condition, we select 100 random action trajectories and use the trajectories to perform a full rollout

to get the predicted costs. We then plot the sum of the true simulator 7-step cost on the x axis and the sum of the AIS predicted 7-step cost on the y axis as predicted at timestep 0.

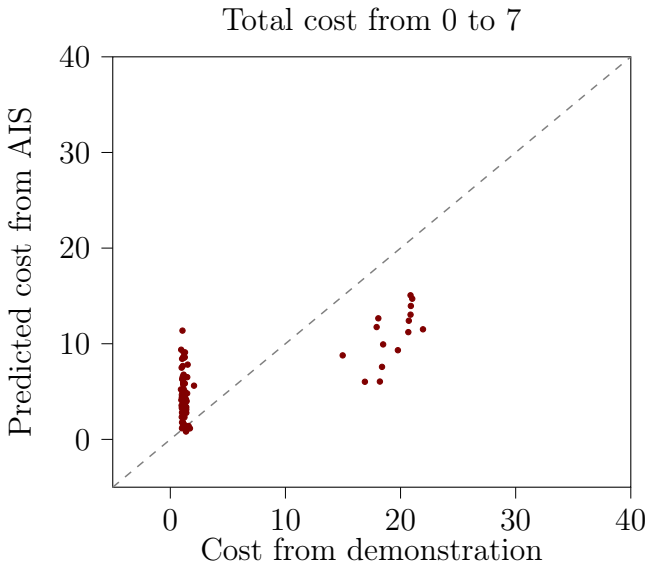


Figure 5-5: Plot of the AIS predicted cost vs the true cost from timestep 0 to 7

With perfect predictions, all of the points would lie on the gray diagonal line. We can visually see the nature of our large prediction error. We can see two distinct clusters. The left cluster represents trajectories with a low cost. We can see that for these good trajectories, the AIS predicts the true cost closely or overpredicts. Out of all good trajectories, AIS correctly recognizes some of them as having low costs, but mistakenly predicts many of them as bad episodes. The cluster on the right represents episodes that are bad. We see that AIS tends to underpredict the cost on these.

The reason why this model with poor prediction is still able to stabilize trajectories is because when choosing a trajectory, the AIS-MPC controller will choose the argmin out of all of its predictions. We see in this graph that the action with the smallest predicted cost also has a small true cost, meaning that it's actually a good trajectory.

This shows that our model doesn't have to be able to accurately approximate the true costs in order to work well for MPC. It just has to ensure that the optimal trajectory based on the AIS models is in fact a good trajectory.

It's surprising that our AIS model learned a model for the cost that has such a high loss, but preserves properties for optimal control. As future work, we would like to further explore why this occurred. Some possible guesses are that maybe we picked a lucky seed or maybe our training set was distributed in a biased way.

## 5.8 Future Work

In this chapter, we established that AIS can stabilize a shoe-tying demonstration to give us significantly more robustness over initial conditions. We only demonstrated these results in simulation for the first part of the shoe tie. In the future, extending this process to the rest of the shoetie will be important, along with getting results in real life, where perfect ground truth won't be available.

There's also lots of potential to improve on parts of the process. It's possible that adding a network to transform the actions could yield better results. Also, including other kinds of observations in the right way is important. We make the shoe tying problem significantly harder by not including forces or contact information. We did attempt adding force and contact into the full state observation through concatenatation, but we weren't able to see any improvement. It's likely we will have to draw upon more sophisticated techniques[13] from the multi-modal learning community to successfully combine these.

# Chapter 6

# Discussion

CMA-ES is a good method to brute force an interpretable solution that can outperform the given demonstration. Intelligent parameterizations of high level motion primitives can allow for the incorporation of various sensor modalities to improve robustness. The resulting trajectories are easily interpretable and verifiable by a human. Direct policy optimization approaches like CMA-ES are powerful and could scale to other complex robotics problems such as those with deformable objects because they are black box methods that can work regardless of the size of the state space or the number of contacts present. CMA-ES will generalize well to tasks that are too complex for existing control methods, but are still simulatable in a realistic way.

However, CMA-ES has limitations. A core limitation of CMA-ES is its dependence on fast simulation. When optimizing for robustness, the computation needed for CMA-ES grows significantly due to the large number of samples needed to rank the policies in a generation by robustness. In chapter 3, we discussed work we've done to make a visually plausible and fast shoe simulator, but there is still a lot more potential to speed this up even more, making CMA-ES a more powerful approach. In this thesis, we mainly focused on working in the simulator, but deploying our solutions onto the real world on real shoes is a significant hurdle that still needs to be passed. This would expose any gaps in how realistic our simulator is and inform us which parameters we need to optimize to be robust against. This method is also

significantly limited by the number of parameters in each action and the total number of actions in the task, since these make up the dimension of the optimization problem. We found that it was possible to encode an open loop policy with just 96 parameters and a closed loop policy that tracks rope keypoints with just 140 parameters. Despite this, without significantly more compute, the shoelace problem likely needs to be optimized in stages. Model-free methods such as CMA-ES allow us to work with difficult problems without having to learn a model, but our program lacks any sort of intuition about the shoe problem.

Learning a good model for a task such as shoe tying is a difficult task. In chapter 5, we showed some results showing that we can get a model for tying a shoelace in a regime around a certain demonstration. One of the challenges that we face with this method is the inability to interpret what the model is learning. Another challenge is collecting a diverse and complete dataset. I focused on only collecting data 0.01m from a demonstration, which is a tiny fraction of the space required to solve this task globally.

AIS fits the data we provide and learns a model for understanding this limited space at a deeper level. It cannot explore or generalize outside this region. CMA-ES on the other hand has no understanding of the problem, but given a starting location to search, it's able to explore complex function spaces to improve on the policy.

For both methods, the general future work is similar. We need to explore the pitfalls when deploying on a real robot. In this thesis, we modeled different shoes by slightly perturbing the location of the laces on the shoe. However, in real life, the variation that occurs between shoes is far more extensive. Just considering collision geometries alone, there is already a huge variety. We also need to find robust controllers for the stages of a shoe tie past the first stage of lifting both ropes and determine if these controllers can be chained together to get a robust trajectory over the whole problem.

# Appendix A

# Network Architecture and Hyperparameters

For the $\sigma$ network, we used 3 fully connected layers with ReLU activation and 100 nodes each. The $\psi$ network is initialized to the identity matrix. The AIS state size is 100. For the loss function, the weight $\lambda$ on the dynamics loss is 100. The Reduce on Plateau learning rate scheduler was used with an initial learning rate of 1e-4. We used batch size of 32. In the future, ensemble learning, dropout, and cross-validation can be used to further improve these results

# Bibliography


[1] Luis Aguirre, Bruno Barbosa, and Antônio Braga. Prediction and simulation errors in parameter estimation for nonlinear systems. *Mechanical Systems and Signal Processing*, 24:2855–2867, 11 2010.

[2] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776 Vol. 2, 2005.

[3] D. Berenson. Manipulation of deformable objects without modeling and simulating deformation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4525–4532, 2013.

[4] Nicholas Charles, Mattia Gazzola, and L. Mahadevan. Topology, geometry, and mechanics of strongly stretched and twisted filaments: Solenoids, plectonemes, and artificial muscle fibers. *Phys. Rev. Lett.*, 123:208003, Nov 2019.

[5] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation, 2018.

[6] Mattia Gazzola, Levi Dudte, A. McCormick, and Lakshminarayanan Mahadevan. Forward and inverse problems in the mechanics of soft filaments. *Royal Society Open Science*, 5:171628, 06 2018.

[7] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.*, 47:99–131, 2005.

[8] Philip E. Gill, Walter Murray, Michael A. Saunders, and Elizabeth Wong. User's guide for SNOPT 7.7: Software for large-scale nonlinear programming. Center for Computational Mathematics Report CCoM 18-1, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2018.

[9] Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Jeffrey Ichnowski, Ashwin Balakrishna, Minho Hwang, Vainavi Viswanath, Michael Laskey, Joseph E. Gonzalez, and Ken Goldberg. Untangling dense knots by learning task-relevant keypoints, 2020.

[10] Aaron Havens, Yi Ouyang, Prabhat Nagarajan, and Yasuhiro Fujita. Learning latent state spaces for planning through reward prediction, 2019.

[11] Xiaobin Jiang, Hongxiang Ren, and Xin He. Simulation of mooring lines based on position-based dynamics method. *IEEE Access*, 7:142796–142805, 2019.

[12] S. Kudoh, T. Gomi, R. Katano, T. Tomizawa, and T. Suehiro. In-air Knotting of Rope by a Dual-arm Multi-finger Robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6202–6207, 2015.

[13] Michelle A. Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950, 2019.

[14] Xuejun Li, Zhiwei Liang, and Huanhuan Feng. Kicking motion planning of nao robots based on cma-es. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 6158–6161, 2015.

[15] W. H. Lui and A. Saxena. Tangled: Learning to untangle ropes with RGB-D perception. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 837–844, 2013.

[16] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics*, 33:1–12, 07 2014.

[17] 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(8):957–982, Jun 2020.

[18] Arturo Mori, Cesar Trujillo, Eric Li, and Rafael Corrales Fatou. *Shoe tying robot*, page 67. Fu Foundation School of Engineering and Applied Science, 2019.

[19] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications, 2018.

[20] Moses C. Nah, Aleksei Krotov, Marta Russo, Dagmar Sternad, and Neville Hogan. Dynamic primitives facilitate manipulating a whip. In *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, pages 685–691, 2020.

[21] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation. *CoRR*, abs/1703.02018, 2017.

[22] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Fred Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-Shot Visual Imitation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun 2018.

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

[24] Jiaming Qi, Guangfu Ma, Peng Zhou, Haibo Zhang, Yueyong Lyu, and David Navarro-Alarcon. Towards latent space based manipulation of elastic rods using autoencoder models and robust centerline extractions, 2021.

[25] Pratyusha Rakshit and Amit Konar. *Recent Advances in Evolutionary Optimization in Noisy Environment—A Comprehensive Survey*, pages 89–169. Springer Singapore, Singapore, 2018.

[26] J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. `https://GitHub.com/FacebookResearch/Nevergrad`, 2018.

[27] M. Saha and P. Isto. Manipulation Planning for Deformable Linear Objects. *IEEE Transactions on Robotics*, 23(6):1141–1150, 2007.

[28] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

[29] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking deformable objects with point clouds. In *2013 IEEE International Conference on Robotics and Automation*, pages 1130–1137, 2013.

[30] Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems, 2020.

[31] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph E. Gonzalez, and Ken Goldberg. Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data. 2020.

[32] A. Suárez, C. Miralles, and S. Medina. On the application of cma-es to biped walk. Technical report iri-tr-17-03,, Institut de Robòtica i Informàtica Industrial, CSIC-UPC.

[33] J. Takamatsu, T. Morita, K. Ogawara, H. Kimura, and K. Ikeuchi. Representation for knot-tying tasks. *IEEE Transactions on Robotics*, 22(1):65–78, 2006.

[34] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

[35] H. Wakamatsu, A. Tsumaya, E. Arai, and S. Hirai. Manipulation Planning for Knotting/Unknotting and Tightly Tying of Deformable Linear Objects. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2505–2510, 2005.

[36] F. Wang, E. Burdet, R. Vuillemin, and H. Bleuler. Knot-tying with Visual and Force Feedback for VR Laparoscopic Training. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 5778–5781, 2005.

[37] Weifu Wang and Devin Balkcom. Knot grasping, folding, and re-grasping. *The International Journal of Robotics Research*, 37(2-3):378–399, 2018.

[38] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. *Robotics: Science and Systems XVI*, Jul 2020.

[39] Lang Xu and Qian Liu. Real-time inextensible surgical thread simulation. *International Journal of Computer Assisted Radiology and Surgery*, 13:1019 – 1035, 2018.

[40] Mengyuan Yan, Gen Li, Yilin Zhu, and Jeannette Bohg. Learning topological motion primitives for knot planning, 2020.

[41] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning Predictive Representations for Deformable Objects Using Contrastive Estimation, 2020.

[42] Wenbo Zhang, Karl Schmeckpeper, Pratik Chaudhari, and Kostas Daniilidis. Deformable linear object prediction using locally linear latent dynamics, 2021.

[43] Peng Zhou, Jihong Zhu, Shengzeng Huo, and David Navarro-Alarcon. Lasesom: A latent representation framework for semantic soft object manipulation, 2020.