Push based Manipulation of a Cubic Block using a Nonholonmically Constrained Planar Pusher

Alrick C. Dsouza

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2021

Reading Committee:

Siddhartha Srinivasa, Chair

Ashis Banerjee

Santosh Devasia

Program Authorized to Offer Degree:
Department of Mechanical engineering

University of Washington

Abstract

Push based Manipulation of a Cubic Block using a Nonholonmically Constrained Planar Pusher

Alrick C. Dsouza

Chair of the Supervisory Committee:
Professor Siddhartha Srinivasa
Computer Science and Engineering

This work aims to study the nonprehensile planar manipulation of a cubic block using a nonholonomic MuSHR race car equipped with a flat planar bumper/pusher. The underactuated and nonholonomic nature of our setup and the uncertainty of physical parameters like friction coefficients between contact surfaces, the pressure distribution of the cubic block, etc. pose challenges for planning and control. To tackle these challenges and the sliding motion of the block along the pusher, our method uses an analytical model to output a set of stable pushes in velocity space to generate trajectories from the block's start to it's goal position using a graph-based planner under a quasi-static assumption. Stable pushes are defined as push actions in velocity space that do not cause sliding motion of the block w.r.t. the pusher. We then deploy an LSTM based MPC with added stochasticity to approximate the pushing dynamics and track the planner trajectories. We conduct pushing experiments and compare our LSTM based MPC against a first-order Markov Gaussian Process and stable pushing-based MPCs for adaptability to different pushing experiments and robustness to initial sliding displacements. Our results suggest that the LSTM based MPC is more robust to initial sliding displacements compared to baseline alternatives and can be used to track planner and non-planner outputted trajectories.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

BUMPER/PUSHER: A planar surface rigidly attached to the front of the MuSHR race car to push the block.

FRICTION CONE: Cone created from the resultant of the normal and friction forces at the contact surface.

GAUSSIAN PROCESS: A non-parametric learning model that makes predictions which are a function of the mean and covariance matrices of the training data.

LIMIT SURFACE: A closed and convex surface containing the origin where forces from the origin to any interior point do not overcome the forces of friction. Forces to a surface point cause motion at a constant velocity, and forces going through the surface cause acceleration.

LONG SHORT TERM MEMORY: (LSTM) network is a neural net that outputs the future state of the system by inputting a set of previous states. The LSTM consists of cell states and hidden states that retain long and short term memory respectively.

MODEL PREDICTIVE CONTROLLER: (MPC) is an advanced process controller that optimizes a cost function under set constraints. The MPC predicts trajectories for a fixed horizon length given input actions, and outputs the input action for the trajectory with the least cost.

MUJOCO: A multi-contact dynamics simulator.

MUSHR RACE CAR: Multi-agent system for nonholonomic racing (MuSHR) are mobile robots used to perform pushing experiments in this work.

STABLE PUSH: A pushing action that does not cause any sliding motion between the pusher and the pushed block.

TRANSFORMER: An attention based neural net model that makes predictions by inputting a sequence of inputs. The Transformer model's attention mechanism weighs the significance of each input in the sequence necessary for making predictions.

ACKNOWLEDGMENTS

I would first like to thank Dr. Siddhartha Srinivasa, whose expertise was invaluable to completing this work. His insightful feedback consistently steered me in the right direction and brought my work to a higher level.

I would also like to thank Matthew Schmittle for guiding me through my research and the writing process. His guidance was of immense help, and I would not be able to complete this work without him. I would like to thank my colleague Tudor Fanaru for his wonderful collaboration.

I would also like to thank Dr. Christoforos Mavrogiannis, Patrick Lancaster and others at the MuSHR team for their guidance and support with the lab equipment and brainstorming sessions.

Finally, I want to thank my parents Cyril A. Dsouza and Flavy Dsouza, my sister Crissel J. Dsouza, Sai Pavan Chitta, Arshdeep S. Brar, Prasanna S. Raut, Ekta U. Samani, Chinmay K. Ajnadkar, Gaurav S. Mahamuni and many others. This accomplishment would not have been possible without them. Thank you.

Chapter

INTRODUCTION

Pushing is a primitive motion that can increase a robot's manipulative capabilities and ease planning and control efforts in crowded environments [Rockett, 2020]. Consider a robot manipulator procuring a milk carton placed at the back of a fridge compartment [Stüber et al., 2020]. Pushing obstacles out of the way is more efficient than picking and displacing each obstacle at a time. Humans often use pushing to close doors, move boxes, clear clutter on desks, especially in scenarios where the accuracy of the applied force is not of the highest priority. Pushing is useful to move or reorient large and heavy objects and reduce uncertainty to grasp objects in cluttered environments. Pushing is a nonprehensile action and does not need any special grippers, which makes it very versatile. It's essential for robots to possess both prehensile and nonprehensile actions to operate seamlessly in human environments [Stüber et al., 2020].

The effectiveness of humans combining pushing with prehensile tasks has inspired a significant amount of research. Humans can effortlessly transfer learned skills to objects of different shapes and sizes. However, there are several challenges while transferring the skill of pushing to robot platforms, including non-linear dynamics of pushing, the uncertainty of friction coefficients and contact regions, toppling [Mason, 1986], and noisy sensor measurements [Rockett, 2020]. In this thesis, we focused on planar push-based manipulation of a cubic block of fixed dimensions using a nonholonomic MuSHR race car with an attached flat bumper in the front acting as an end-effector. The end-effector is allowed to have a planar contact with the block.

This work contributes a planner and Model predictive controller (MPC) for pushing a cubic

block using a nonholonomically constrained pusher. Further, we conducted an experimental comparison of two data-driven and an analytically modeled MPC for pushing and robustness to initial sliding displacements. Lastly, we performed an experimental evaluation to determine if the quasi-static and Markov assumptions are reasonable for long-horizon planning and local control respectively. We conducted pushing experiments both in MuJoCo Physics engine [Todorov et al., 2012] and in the real world with MuSHR [Srinivasa et al., 2019], a race car manufactured by the Personal Robotics Lab at the University of Washington, Seattle. We detected real-world poses of the objects in our experiments using Motion Capture IR cameras (mocap).

Chapter

BACKGROUND

There has been a variety of work in push-based manipulation using robots, ranging from analytical to data-driven learning approaches.

2.1 Analytical models of pushing

The initial methods to model pushing were analytical and made assumptions about the nature of forces, the pressure distribution of the pushed object, and contact regions. [Mason, 1986] proposed the Voting Theorem as a fundamental theorem to acquire the sense of rotation of the pushed object. Adding to Mason's work [Peshkin and Sanderson, 1988a] found bounds on the rotation rate of the pushed object using a single-point push method. [Mason, 1990] analyzed pushing motions to slide blocks along a wall and, [Narasimhan], [Yoshikawa and Kurisu, 1991] and [Lynch and Mason, 1996] studied planar pushing using point contact.

The friction cone displays the range of friction forces that do not cause relative motion between two surfaces. [Goyal et al., 1991] introduced the concept of limit surface, which derived a relationship between the sliding of a pushed object and friction forces under the assumption of known pressure distributions and quasi-static pushing. Quasi-static pushing assumes the absence of inertial forces and that sliding motion occurs only due to friction forces. [Robert D. Howe, 1996] developed further approximations of the limit surface for common pressure distributions. Following this, [Lee and Cutkosky, 1991] proved that an ellipsoidal approximation of the limit surface is it's lower bound.

A better understanding of pushing mechanics lead to the development of push-based planning algorithms. [Agarwal et al., 1997] proposed a planning algorithm to push a disk-shaped

object using a point contact. [Miyazawa et al., 2005], [Cappelleri et al., 2006] developed RRT planning algorithms for pushing using Mason's pushing equations. [Dogar and Srinivasa, 2010] introduced a planner to isolate objects by pushing in a cluttered environment and later grasping them. [Lee et al., 2015] presented a hierarchical method to plan for sequences of prehensile and nonprehensile actions. Analytical models provided key insights into the nature of pushing and are still used in several push based planning algorithms.

2.2 Data-driven models of pushing

Analytical models provided a deeper understanding of the pushing dynamics under several assumptions about the nature of forces and type of contact, however they did not consider the stochastic nature of pushing and uncertainty of physical parameters like friction coefficients, pressure distributions, etc. Data-driven methods aim to estimate physical parameters and pushing dynamics through observations. [Lynch, 1993] and [Yoshikawa and Kurisu, 1991] proposed estimating friction parameters by performing pushing experiments. [Jiaji Zhou et al., 2016] developed models to represent planar friction and proposed a framework for representing planar sliding. [Yu et al., 2016] used a Non-parametric Gaussian Process model to approximate the forward dynamics. In studies with discrete action space, the concept of *affordances* gained popularity. The term was invented by [Gibson, 1979] and referred to the possibility of performing an action in a given environment. [Ugur, 2011] and [Moldovan et al., 2012] learned affordance models for various distinct objects in their work.

[Chang et al., 2017], [Watters et al., 2017], [Fragkiadaki et al., 2016], [Stüber et al., 2018], etc. developed deep learning models to learn planar pushing dynamics. [Zeng et al., 2018] learned synergies between pushing and grasping using deep reinforcement learning to ease grasping. Thus, data-driven learning approaches provide pushing models which take into account the stochastic nature of pushing and uncertainty of physical parameters for a variety of scenarios with the extra work of collecting training data.

Our work consists of two frameworks, namely the planner and the MPC. The planner is used for longer horizon planning and is inspired by [Lynch, 1992]. The local planner is implemented as a MPC framework and inspired by data-driven methods like [Yu et al., 2016] and [Busch et al., 2020]. However, our problem differs in the following important ways: (a) our pusher is a flat surface allowing for planar contact with the block, (b) and is nonholonomically constrained, limiting it's manipulability [Rockett, 2020]. We use a graph-based planner instead of a grid-based planner for better computational time and learn the pushing dynamics using an LSTM to account for propagated error in the MPC rollouts.

Chapter

PROBLEM STATEMENT

The goal of this thesis is to to manipulate a cubic block using a flat-surface pusher attached to the front side of the MuSHR race car as shown in Fig 3.1 from a start position in front of the pusher to a goal position. Let $\{G\}$ be the global frame and $\{P^c\}$ frame be attached to the center of the MuSHR race car as shown in Fig 3.1. Represent the state vector $p^c = \{x^c, y^c, \theta^c\} \in SE(2)$ as position and orientation of $\{P^c\}$ in frame $\{G\}$. We denote $u = (u_s, u_\phi)) \in U \subseteq \mathbb{R} \times \mathbb{R}$ as the control input to the race car, where u_s is speed and u_ϕ is the steering angle.

The kinematics of the race car is represented using the differential equation provided by the Bicycle model [Michael, 2020].

$$\begin{aligned} x^c_{t+1} &= x^c_t + \frac{L}{tan\phi}(sin\,\theta^c_{t+1} - sin\,\theta^c_t) \\ y^c_{t+1} &= y^c_t + \frac{L}{tan\phi}(-cos\,\theta^c_{t+1} + cos\,\theta^c_t) \\ \theta^c_{t+1} &= \theta^c_t + \frac{v}{L} tan\phi \Delta t \end{aligned} \tag{3.1}$$

where v, ϕ are the speed and steering angle of the MuSHR race car respectively. Finally, fix the frame $\{B\}$ on the center of block and denote it's state vector as $p^b = (x^b, y^b, \theta^b) \in SE(2)$ w.r.t to frame $\{G\}$.

The flat-surface pusher acts an end effector constrained nonholonomically which manipulates the block through planar pushing. The pusher has a length of 0.225 m, and it's front side is glued with sandpaper to increase the friction coefficient between the block and pusher. The cubic block is placed at the center of the pusher at the start, measures 0.1 m each side and is made of wood. We test our pushing experiments in MuJoCo [Todorov et al.,

2012], a contact dynamics simulator and measure poses in the real world using a Motion capture system with an accuracy of 0.04088 mm. In order to tackle the pushing problem, we develop a planner to generate a set of trajectory points $(p_0^b, p_1^b, p_2^b, ...p_n^b)$ in $SE(2)$ where p_0^b and p_n^b are the start and goal positions of the block. Next we design a MPC modelled by a learned pushing dynamics given by

$$f : SE(2) \times SE(2) \times \mathbb{R}^2 \rightarrow SE(2)$$

s.t.

$$f(p_t^c, p_t^b, u) = p_{t+1}^b$$

where the arguments for f are the block and car states and current action, and it's output is the block's future state. Finally, we assume that our pushing experiments occur in a quasistatic regime, implying that the MuSHR race car moves slow enough to ignore inertial forces.

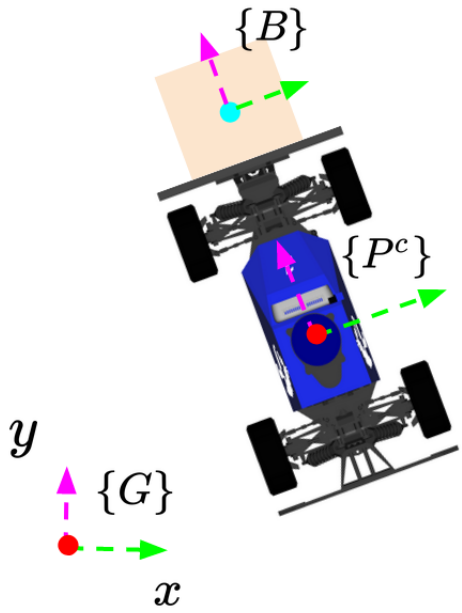


Figure 3.1: Nonholonomic pusher-slider system

Chapter

APPROACH

Our approach consists of two stages. The first includes learning the dynamics of pushing and integrating the learned model in the MPC framework as discussed in 4.1. In the second stage given in 4.2, we develop a planner to generate trajectories for stable pushing.

4.1 Model Predictive Controller for Pushing

The Model Predictive Controller [García et al., 1989], [Richalet et al., 1978] is an iterative, finite-time horizon controller with many applications in controls and robotics. The MPC provides excellent flexibility in model choice and is used to control linear as well as nonlinear systems. In this section, we describe how we use the MPC to address the pushing problem.

The MPC can be expressed as an optimization problem to minimize a cost function in a finite time horizon. The MPC first generates a set of predictions for a finite horizon called rollouts, given inputs $u = (u_s, u_\phi)$ and the model of the system. The first input of the rollout with the least cost is returned. This process is repeated until an end condition is met.

The MPC optimization is described below

$$\begin{aligned} \min_{J} \quad & J = \sum_{t=0}^{t=t_H} C(x(t), x_{ref}(t)) \\ \text{subject to} \quad & u(t) \in U, x(t) \in X, \\ & x(t+1) = f(x(t), u(t)), \\ & x(0) = x_0 \end{aligned} \tag{4.1}$$

where C is the cost function, x_{ref} is a reference lookahead waypoint on the trajectory, t_H is the time horizon, $x(t)$, $u(t)$ are the state and the input control vectors respectively at time t . A lookahead waypoint is a reference point at a set distance ahead of the robot along the trajectory. We use a lookahead waypoint instead of the nearest point on the trajectory to the block to smoothly track the trajectory.

We can express the position of the block w.r.t. to the car as,

$$p_t^b = p_t^c + p_t^{b \to c} \tag{4.2}$$

where $p_t^{b \to c}$ is the block state w.r.t to the car at time-step t. Using Euler approximation method and the bicycle model in 3.1, we can approximate the car dynamics as

$$p_{t+1}^c = p_t^c + \begin{bmatrix} v\,sin\,\theta \\ v\,cos\,\theta \\ \frac{v}{L} tan\,\phi \end{bmatrix} \Delta t \tag{4.3}$$

Combining 4.2 and 4.3 the state of the block at time-step t+1 is given by

$$p_{t+1}^b = p_t^c + \begin{bmatrix} v\,sin\,\theta \\ v\,cos\,\theta \\ \frac{v}{L} tan\,\phi \end{bmatrix} \Delta t + p_{t+1}^{b \to c} \tag{4.4}$$

The only unknown in 4.4 is $p_{t+1}^{b \to c}$. This unknown relative position can be approximated using a parametric function as shown below.

$$p_{t+1}^{b \to c} = f((p_t^b, p_t^c), (p_{t-1}^b, p_{t-1}^c), ..., u_t) \tag{4.5}$$

We found through experimentation that it is more efficient to predict the position of the block w.r.t. the MuSHR car instead of predicting the block pose in global frame $\{G\}$. Further, our hypothesis is that learning a function which inputs a set of previous states instead of just the previous state as shown in 4.5 will account for propagated error while generating rollouts in the MPC.

4.1.1 Cost Function

The cost function for the MPC optimization problem is a combination of a weighted L1 norm of the difference between the rollouts and reference lookahead waypoint, and the magnitude of sliding along the pusher. We designed the MPC cost function emperically through monitoring controller performance during pushing experiments. Our cost is defined as follows:

$$C(p^b, p^b_{ref}(t)) = \sum_{t=0}^{T} [(\mid p^b_t - p^b_{ref})^T \, Q \mid + \mid p^{slide}_t \mid] \tag{4.6}$$

where p^b_{ref} is the reference lookahead waypoint, p^b_t is the rollout prediction at time step t, p^{slide}_t is the displacement of the block along the pusher, and Q = $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. Our cost function penalizes deviation from the reference trajectory and pushing actions that cause sliding in any direction.

4.2 Learning the Pushing Dynamics

After laying out the forward model in 4.4, with $p^{b\to c}_t$ as the only unknown term, we then train an LSTM [Dolphin, 2021] model to approximate this unknown quantity. We approximate the pushing dynamics using an LSTM instead of a first order Markov model to account for propagated error generated in the MPC rollouts.

4.2.1 Procuring Data

To train and test the LSTM model, we use MuJoCo to generate pushing data. We consider an obstacle free world and the friction coefficients between the block and pusher, and the block and ground are both 0.5.

We collect data by running three policies namely straight path policy, curved policy and a random Policy to push the block. We use Halton sampling [Halton, 1964] to sample uniformly over 2π for initial block orientation. The block is placed at the origin and the pusher

is placed exactly 4 cm behind the block. We also introduce noise to the block's start pose and orientation to learn a more general pushing dynamics.

The straight path pushing policy outputs a constant speed and sets the steering angle to zero.

$$\pi_u : f(t) = (u_s, 0) \tag{4.7}$$

where u_s is a fixed speed. The curved policy works similar to the straight path pushing policy with an added constant steering angle.

$$\pi : f(t) = (u_s, u_\phi) \tag{4.8}$$

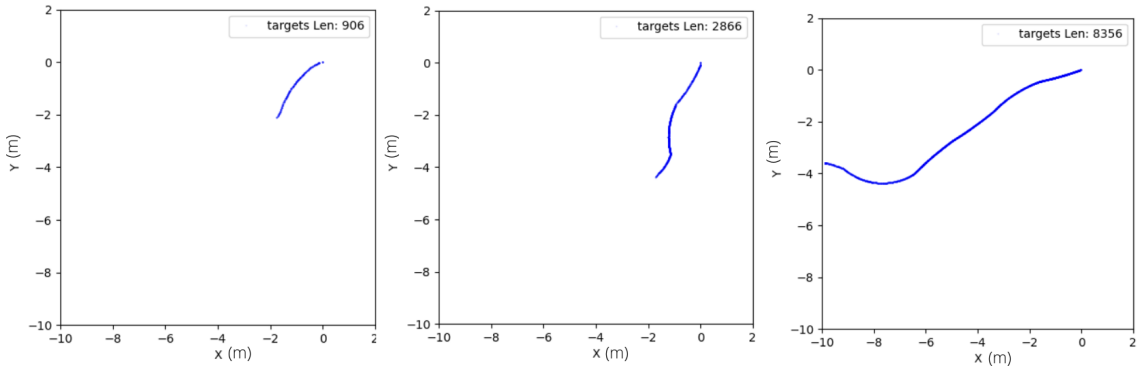
The straight and curved path policies record the most common and basic type of trajectories, however in order to learn more complex pushing trajectories we invoke a random policy as well. Our random policy chooses a random value for speed and steering angle between $(0.3, 1)$ m/s and $(-0.3, 0.3)$ rad respectively for a fixed time step.

Algorithm 1 Random Policy

```
step ← 200
run_time ← 10000
i ← step
while (Block is in contact with pusher) and (i <= run_time) do
  if modulo(i, step) = 0 then
    Generate Random action (u_s, u_φ), (u_s, u_φ ∈ U)
  end if
  Increment i
end while
```

We record the block and car states and the input speed and steering angle before and after the pushing action. We end data collection for a particular run when the block loses

contact with the pusher. We generate a total of 536 trajectories, of lengths ranging from 0.5 to 12 meters approximately. Example trajectories can be found in Fig 4.1. We shuffle trajectories in the data and use 80% and 20% of the data for training and testing respectively.



(a) Trajectory of length 906 data points.

(b) Trajectory of length 2866 data points.

(c) Trajectory of length 8356 data points.

Figure 4.1: Trajectories of varying lengths in the dataset.

4.2.2 Long Short term Memory Network (LSTM)

After recording training and testing data, we train a Long Short Term Memory (LSTM) [Dolphin, 2021] Neural network of sequence length 500 to approximate a function that maps a sequence of previous system states and current input to the future system state.

We want to approximate a joint probability distribution over the entire data given as

$$p(X) = p(x_1, x_2, x_3, ...)$$

However, since we can only collect a finite set of data samples, we write the joint probability distribution as

$$p(X) = p(x_1, x_2, x_3, ..., x_n) \tag{4.9}$$

where n is the total number of data points.

Using the Bayesian theorem of conditional probability

$$p(A, B) = p(A|B)p(B)$$

we deduce

$$p(X) = \prod_{t=1}^{T} p(x_t|x_{<t}) \tag{4.10}$$

Finally, to improve computation efficiency of 4.10 we apply the nth order Markov assumption to approximate $p(X)$ autoregressively [Rao, 2021], [Jurafsky and Martin, 2008] which results in

$$p(X) = \prod_{t=1}^{T} p(x_t|x_{(t-1)\rightarrow(t-n)}) \tag{4.11}$$

We use an LSTM model to approximate the joint probability distribution. The architecture of LSTM is inspired by logic gates in electric circuit's [Dolphin, 2021]. The key reason for choosing an LSTM to model a nth order Markov model is the LSTM's ability to retain memory. The LSTM comprises of 3 gates, the Forget, Input and Output gates (not shown

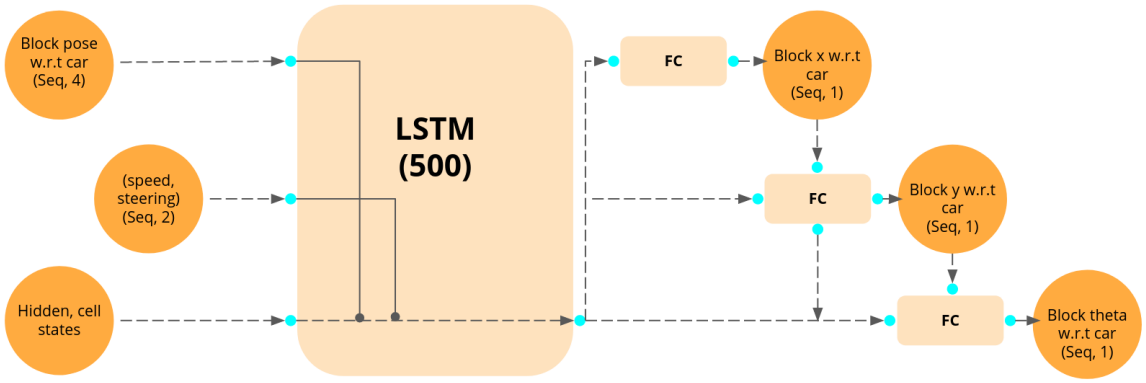


Figure 4.2: LSTM model architecture with sequence length 500 and added stochasticity.

in Fig 4.2). The Forget gate modifies the cell state and erases unwanted memory forwarded from the previous LSTM block. The Input gate adds new information to the cell state and

finally the Output gate modifies the hidden state.

Our LSTM model takes as inputs a total of 500 previous states starting from the current state of the system and maps the combined inputs of $(p_t^{b\to c}, p_{t-1}^{b\to c}, ..., p_{t-499}^{b\to c}, u_t)$ to $p_{t+1}^{b\to c}$. Pushing is a stochastic action [Stüber et al., 2020], which means given the same initial state and inputs, pushing actions can lead to different outcomes. Hence, we add stochasticity to our model during training by using predicted x displacement to predict y displacement and predict θ displacement using predicted x and y displacements of the block as shown in the model architecture in Fig 4.2. In addition, we use teacher forcing method to speed up the LSTM training and use smooth L1 loss as our loss function. In our case, the smooth L1 loss equation [Pytorch, 2019] is given by

$$l_n = \begin{cases} 0.5(x_n - y_n)^2 & \text{if } |x_n - y_n| < 1 \\ |x_n - y_n| - 0.5 & \text{otherwise} \end{cases}$$

where x_n is the model prediction and y_n the corresponding label.

4.3 Planning for trajectories using the friction cone and limit surface analytical model

Our MPC framework acts as a low-level planner to track trajectories. However, the non-holonomic constraint of the MuSHR race car and the sliding motion of the block makes it difficult to track every trajectory without losing the block. Thus, there is a need for a higher level planner to output a trackable trajectory without losing the block. We call these set of trajectories stable trajectories.

In this section, we develop a set of stable pushes using the concepts of friction cone and limit surface. Stable pushes are a set of inputs to the MuSHR race car in velocity space that do not cause any relative sliding motion of the block with respect to the pusher. Using stable pushes, we generate motion primitives which are replicated to form a graph for the planner. The planner then outputs a stable trajectory from the start state of the block to

it's goal state. Motion primitives are defined as motions that can be executed by the robot.

Our planner requires the following assumptions. We assume dry friction (Coulomb friction) for our friction cone and limit surface calculations, and that the friction coefficient is constant. We operate in the quasistatic regime, implying that the MuSHR race car pushes the block slow enough to neglect inertial forces and sliding motion only occurs due to friction forces between the block and the pusher.

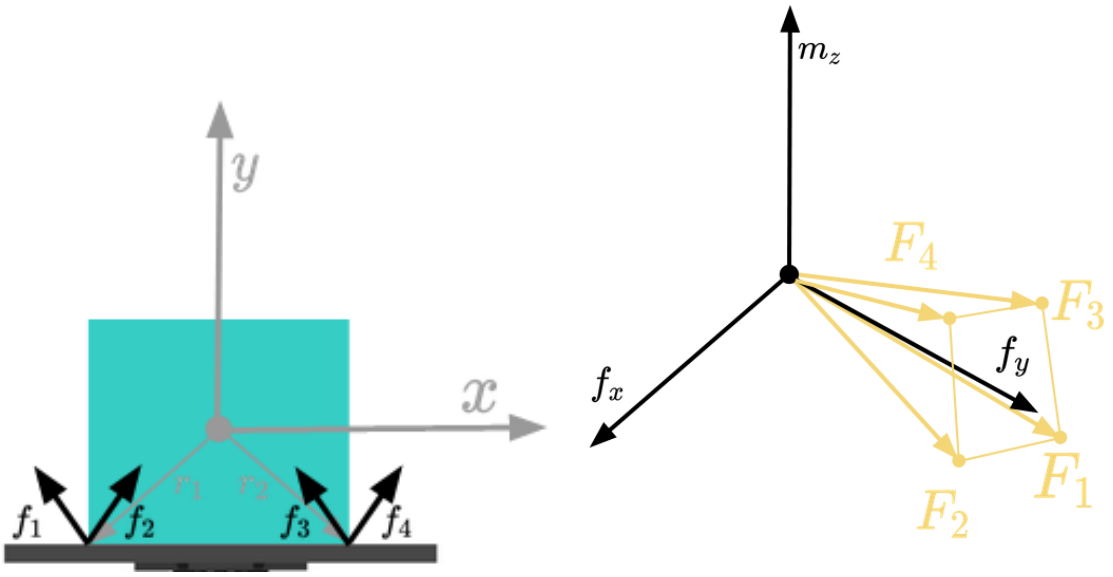
4.3.1 Friction cone and Friction wrench

The friction cone consists of the resultant of the normal and friction forces acting on the block. We use the concept of friction cone to model sliding of the block relative to the pusher. We consider two friction cones acting on two edges of the block as shown in Fig 4.3a, else there is a possibility of the block moving into the pusher. The maximum resultant forces of the normal and friction forces of the two friction cones along with their moments about the z axis is plotted as shown in Fig 4.3b develops the composite wrench cone model. The composite wrench cone represents friction forces that result in the block sliding with constant velocity.

4.3.2 Limit surface

Having developed the wrench cone model, we now want to find a set of friction forces that do not cause sliding motion of the block and convert them to velocity space. We achieve this by using the concept of limit surface introduced by [Goyal et al., 1991] under the assumption of a known pressure distribution.

Consider a pressure distribution of the normal force $p(r)$ that varies with the distance from the center of the block. Friction force acting on a infinitesimal chunk of the block's



(a) Two friction cones at the two edges of the block.

(b) Wrench cone model.

Figure 4.3: The resultant of friction and normal forces and the wrench cone model respectively.

surface is given by

$$f_t = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = -\mu \int_A \hat{v}(r) p(r) dA \tag{4.12}$$

where $\hat{v}(t)$ is the unit velocity vector of the infinitesimal chunk and $f(t)$ is the friction force of the entire block surface. Similarly, the moment in the z direction is given by

$$m_z = \int_A (\, r \times \hat{v}(r)\,) p(r) dA \tag{4.13}$$

Using equations 4.12, 4.13, a known pressure distribution and velocities, we can calculate the limit surface for the pusher-block system. However, it's challenging to find out the true pressure distribution of the block and repeating these calculations for slight changes in pressure distribution is computationally inefficient. To solve this problem, we make an

approximation about the shape of the limit surface. [Robert D. Howe, 1996] show that an ellipsoidal approximation of the limit surface acts as it's lower bound for any pressure distribution. Not only does this help improve computational efficiency but also increases the safety factor to avoid sliding. The ellipsoidal limit surface is expressed as shown below.

$$\frac{f_x^2}{(\mu f_n)^2} + \frac{f_y^2}{(\mu f_n)^2} + \frac{m_z^2}{(m_z)_{max}^2)} = 1 \tag{4.14}$$

It can be shown that for any force on the limit surface, it's normal is the constant velocity of sliding as shown in Fig 4.4. Thus, we can intersect the composite wrench cone of stable forces with the limit surface and get the stable velocities.

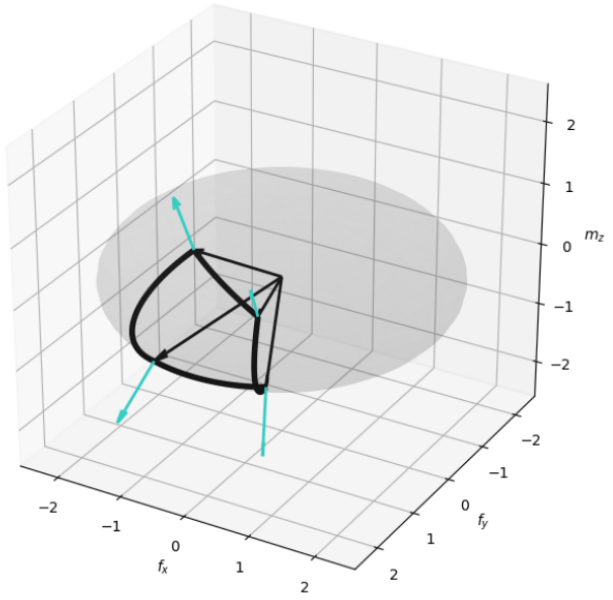


Figure 4.4: Intersection of limit surface with black vectors representing the wrench cone and the blue vectors their associated velocities.

4.3.3 Graph based planning

We can generate a set of stable motion primitives from the limits of the stable velocities for the block. Since we are within the limits of the friction forces that do not cause sliding

motion of the block, we can assume that the block is glued to the car and generate trajectories using the car dynamics as shown in 4.3.

The graph based planner [sbpl, 2007], [Likhachev, 2010] replicates the motion primitives in space, and uses the A* search algorithm with the heuristic function as euclidean distance.

We now have the planning and MPC frameworks powered by stable pushing and the LSTM model respectively. We also integrate stable pushing with the MPC framework. Here $u_\phi \in (-0.15, 0.15)$. These steering angle values are calculated from the limit surface from the method described in 4.3.2. In addition, p^{slide} is considered constant since we assume that stable pushing does not cause any sliding motion. This results in higher computational efficiency and enables us to carry out pushing experiments in the real world.

Chapter

RESULTS

Now that we've developed our planner and MPC frameworks, we tested our algorithms for pushing blocks, adaptability to different pushing experiments and robustness to initial sliding displacements. We display pushing results and insights for 3 set of experiments. Our baseline is a Gaussian Process (GP) [Rasmussen and Williams, 2006], [Yu et al., 2016] modelled MPC, a first order Markov model and considers the pushing dynamics a function of only the current state and current input. The GP is a non-parametric learning model and requires less than 3000 data points to train. In addition, the GP has also been used in prior works to approximate the pushing dynamics. Lastly, we tested our LSTM modelled MPC against a stable pushing MPC that assumes no sliding motion between the block and pusher and has a constrained set of steering angles provided by the friction cone and limit surface analytical model in 4.3.2. Stable pushing MPC is computationally efficient and can be used to conduct pushing experiments in the real world.

5.1 Metrics

We defined the following metrics to evaluate our pushing experiments.

1. Success

   The qualitative metric Success measures our algorithm's ability to push the block to it's goal position within a set threshold. Since our main focus is pushing the block without losing contact and not precision control, we chose a circular threshold of 3 cm.

2. Path tracking error (PTE)

   This metric is a quantitative measure of how well our MPC tracks the reference tra-

jectory. Larger deviance from the reference trajectory can lead to failure to reach the goal position.

3. Distance from center (DFC)

   DFC measures the average absolute magnitude of the sliding motion of the block along the pusher. Ideally, we want DFC $\rightarrow 0$, as sliding motion leads to loss of contact with the block.

4. Model Performance

   (a) Average multi-step trajectory prediction

   We measure the ability of our models to predict rollouts for a fixed length of 0.5 m in a multi-step prediction fashion. Accurate rollouts are key to MPC control.

   (b) Average single-step prediction error

   The average prediction error measures the ability of the models to predict the future state of the block w.r.t. the car given current car and block states and current input. This is a common metric used to evaluate performance of Machine learning models.

5.2 Experiment 1: Controller Performance

We tested the 3 MPCs on semi-circular trajectories as shown in Fig 5.1. Initial pushing experiments suggested that circular trajectories cause the pusher to lose the block due to sliding motion. Thus, these experiments served as a baseline to test our MPCs ability to stabilize the block at the center of the pusher while tracking trajectories. The trajectories are sampled from semicircles generated from a fixed radius at equal intervals between $(0, 2\pi)$ and the speed of the MuSHR race car is fixed at 0.5 m/s. We generated a total of 3 trajectories of radii 3, 5 and 7 meters.

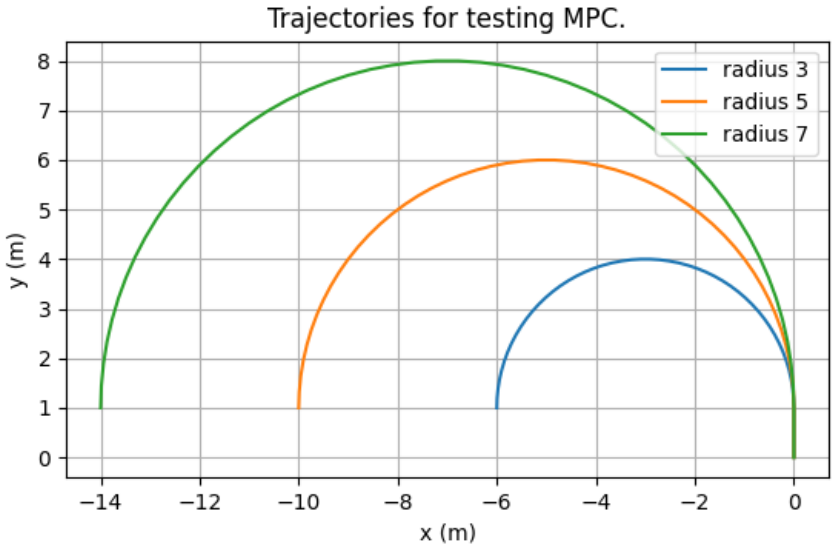


Figure 5.1: Circular trajectories to test the MPC performance.

5.2.1 Observations and Insights

We observed from table 5.1 that the metric values do not differ significantly for all the three MPC frameworks with the exception of the GP based MPC not succeeding in one experiment. In addition, the multi-step trajectory prediction of LSTM and GP models are identical for rollouts of length 0.5 m.

The key insights from this experiment are that we do not need very complicated models to learn the pushing dynamics. We can replace data hungry models like LSTM with a GP, and in our pushing regime and MPC framework, safely assume that the current state of the block is a function of the previous block and car states, proving our hypothesis wrong. However, for longer horizon multi-step predictions, the LSTM outperforms the GP as shown

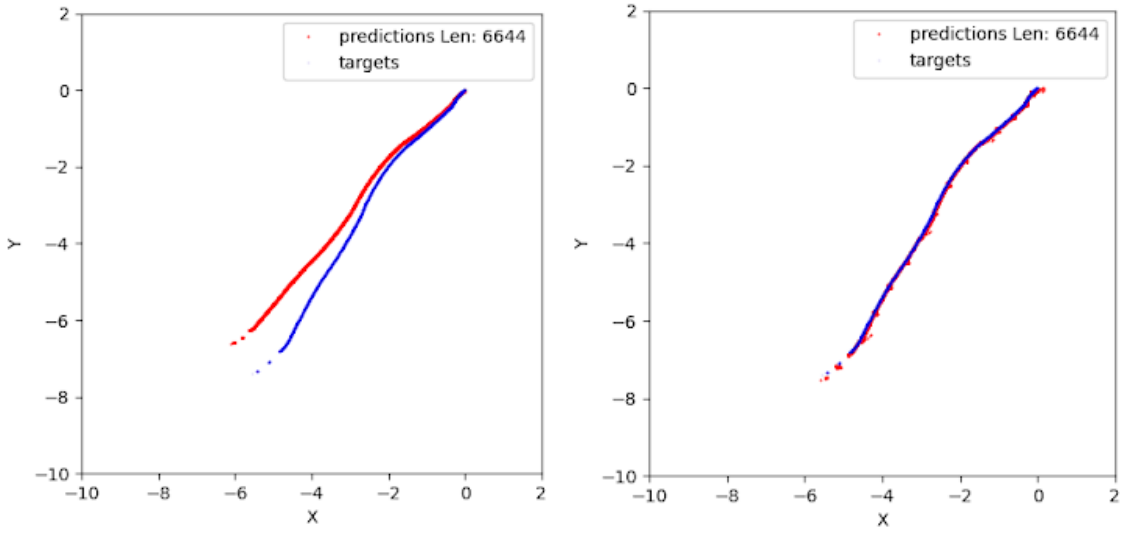
Metrics	LSTM (500)	GP	Stable pushing
Success	3/3	**2/3**	3/3
PTE (cm): (mean, sd)	(10.986, 3.980)	**(10.752, 3.728)**	(12.184, 3.995)
DFC (cm): (mean, sd)	**(4.278, 3.318)**	(4.649, 3.264)	(4.382, 3.178)
Average Multi-step error (cm): (mean, sd)	(0.366, 0.338)	**(0.288, 0.270)**	-
Average single-step error (cm): (mean)	0.21	**0.03**	-

Table 5.1: Average MPC performance on circular trajectories.

Metrics	LSTM (500)	GP
Average multi-step error for trajectory length $\geq$ 4 meters (cm)	**0.434**	17.603

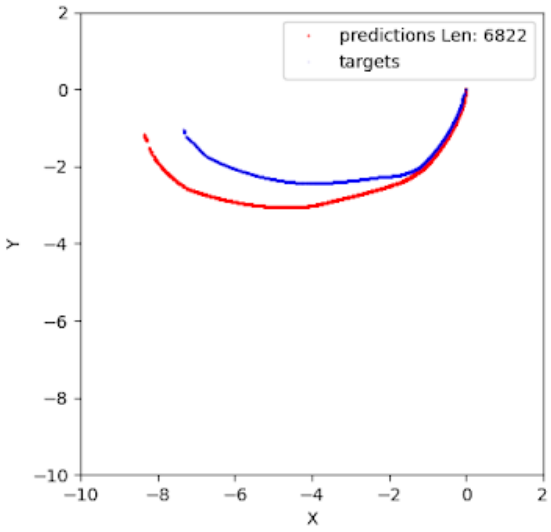
Table 5.2: Multi-step trajectory prediction for longer horizons.

in table 5.2 and Fig 5.2. In addition, the performance of data-driven methods is similar to stable pushing. Thus, an LSTM based MPC is a better alternative when operating with rollout lengths greater than 4 m. Additionally, our LSTM can also be used to find combinations of inputs that cause lower magnitudes of sliding.

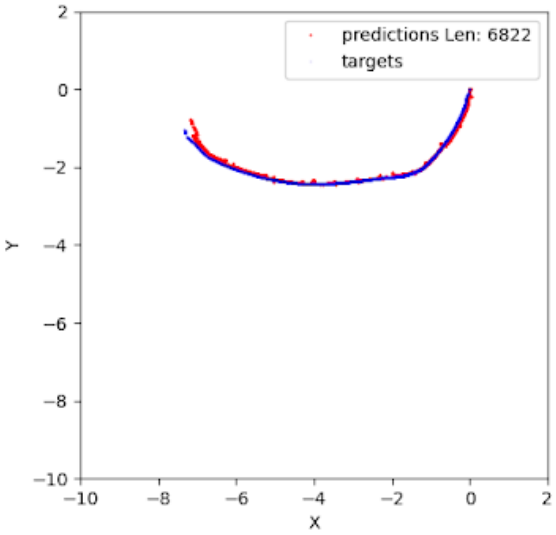


(a) GP multi-step predictions.

(b) LSTM mutli-step predictions.



(c) GP multi-step predictions.



(d) LSTM mutli-step predictions.

Figure 5.2: Multi-step trajectory prediction comparison between the GP and LSTM models for trajectory lengths of 6644 and 6822 data points.

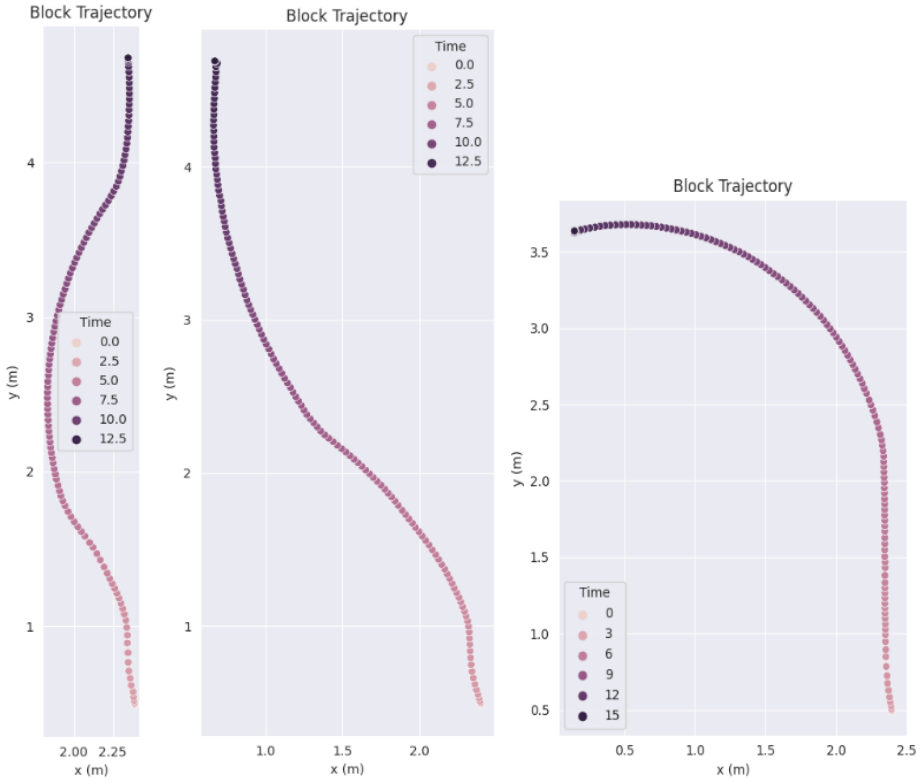
5.3 Experiment 2: Pushing experiments

Pushing experiments included integration of the planner and the MPC frameworks to push blocks. Unlike 5.2 where the MPC tracked semi-circular trajectories generated by fixing the turning radius, in these pushing experiments we first outputted a trajectory using our graph based planner between the block's start position and fixed goal position followed by the MPC tracking the trajectory.

We decided the goal positions of the block such that they lie within the mocap's field of vision. The generated trajectories for the experiments are shown in Fig 5.3. The goal position of the block in the straight path trajectory is straight ahead of it's start position, however due to overshoot of the straight motion primitives, the planner outputs a curved path. Our real world experiments were replicated to best match the simulation start and goal positions of the block. The infrared mocap markers were placed strategically to detect the center of the car and the block. Initial experiments with pushing along circular

Metrics	**Planner +**			
	LSTM (500)	**GP**	**Stable pushing**	**Stable pushing (real world)**
Success	3/3	**2/3**	3/3	3/3
PTE (cm): (mean, sd)	(7.465, 4.764)	(7.302, 6.523)	**(6.974, 6.262)**	(8.686, 7.631)
DFC (cm): (mean, sd)	(2.211, 3.2)	(2.379, 3.318)	**(1.916, 1.055)**	(0.788, 0.351)

Table 5.3: Average MPC performance on planner generated trajectories. Trajectories are shown in Fig 5.3



(a) Straight path.

(b) S turn path.

(c) Left turn path.

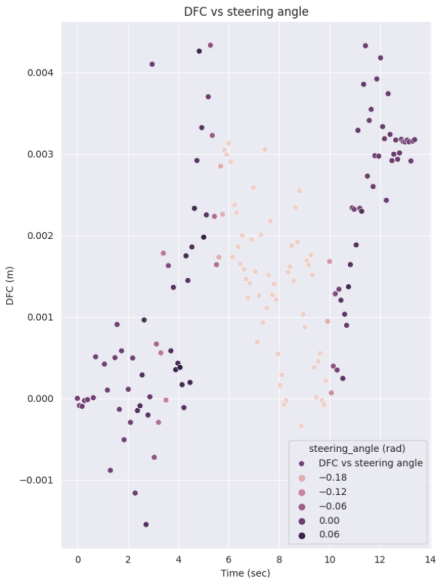
Figure 5.3: Planner outputted trajectories for pushing experiments in 5.3

trajectories suggested a continuous sliding motion and experiments with smaller turning radius lost the block quicker. These observations hinted that larger magnitudes of sliding is proportional to the steering angle. We confirm this observation by plotting the DFC metric against time.

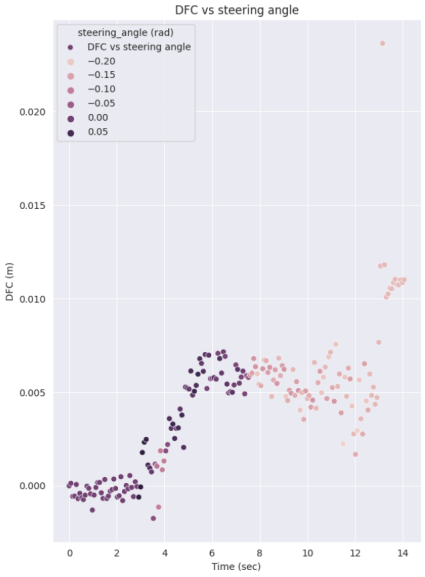
5.3.1 Observations and Insights

In these experiments, we observed that the stable pushing MPC performs better as indicated by our metrics compared to both the LSTM and GP based MPCs. Sliding motion is proportional to the steering angle of the MuSHR car as shown in Fig 5.4. Lastly, as shown by Fig 5.5, our block is stabilized at the center of the pusher with the respect to the sliding limits.

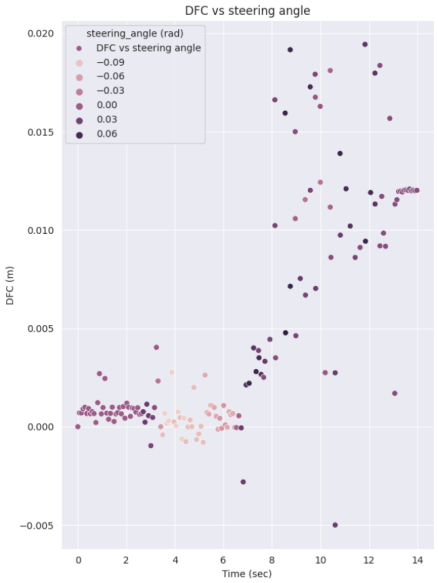
Our key insight from the experiments in 5.3 is that our planner's quasistatic assumption for pushing is reasonable displayed by the negligible sliding motion in figure 5.5.



(a) Straight path

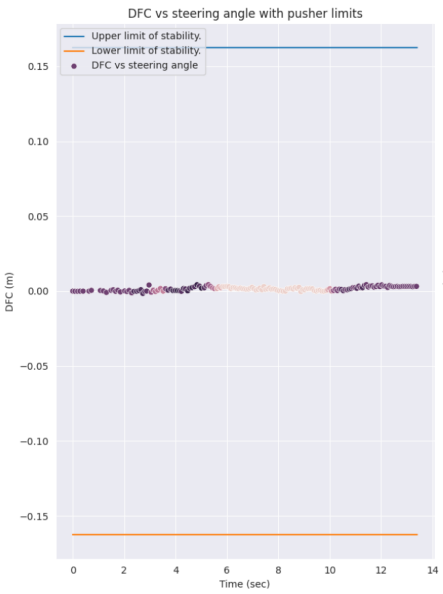


(b) S turn path



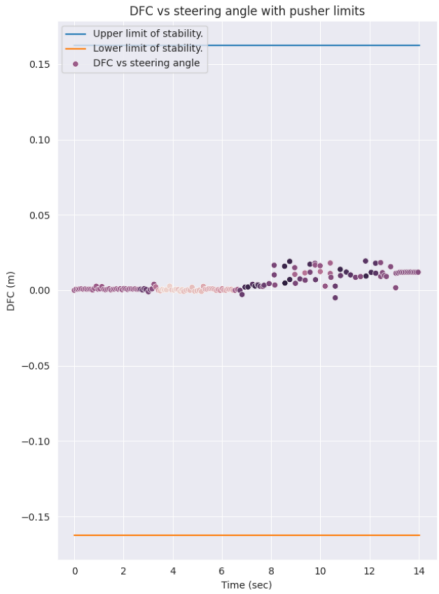
(c) Left turn path

Figure 5.4: Relationship between DFC metric and steering angle. The plots are color coded as per the steering angles with pink data points being a right turn and darker violet data points being left turns. Magnitudes of the steering angles are mentioned in the labels



(a) Straight path with pushing limits

(b) S turn path with pushing limits



(c) Left turn path with pushing limits

Figure 5.5: DFC vs steering angle with added pushing limits. At the pushing limit the block loses contact with the pusher.

5.4 Experiment 3: Robustness Performance

Uncertainty of physical parameters like friction coefficients, pressure distribution of the block, inertial forces etc. require our pushing experiments to be robust to slight variations of these parameters. Our experiments measured robustness through introducing an initial sliding displacement. We chose the following initial displacements for the three pushing trajectories considered in 5.3

$$\text{Displacements} = [-3, -1, 1, 3]\, cm.$$

where positive and negative displacements were an initial displacement of the block with respect to the center of the pusher in the right and left directions respectively. Experiments with an initial displacement of magnitude greater than 3 cm, resulted in losing the block for all the MPCs.

Metrics	**Planner +**		
	LSTM (500)	**GP**	**Stable pushing**
Success	12/12	**10/12**	12/12
PTE (cm): (mean, sd)	(7.794, 4.887)	**(7.036, 6.543)**	(8.909, 3.131)
DFC (cm): (mean, sd)	**(2.260, 3.198)**	(3.450, 2.907)	(2.304, 1.894)

Table 5.4: Average MPC performance on planner outputted trajectories with initial sliding displacements.

5.4.1 Observations and Insights

We observed that the stable pushing MPC didn't account for stabilizing the block to the center of the pusher resulting in larger DFC metric compared to the LSTM and GP based models. In addition, the GP lost contact with the block in two experiments with initial displacements of -3 cm and -1 cm.

The key insight from these set of experiments is that the ability of the LSTM to successfully stabilize the block while tracking the trajectory proves that it's more robust compared to the GP and stable pushing based MPCs.

Chapter

CONCLUSION

We first trained a Long Short Term Memory (LSTM) model with added stochasticity in an autoregressive fashion using a sequence length of 500 previous states to learn the pushing dynamics. We then compared our LSTM based MPC against a Gaussian Process (GP) and stable pushing based MPCs on pushing blocks along circular trajectories and observed that the LSTM based MPC is able to push the block successfully for all the experiments unlike the GP based MPC. However, both models approximated the pushing dynamics fairly similarly as shown in table 5.1, suggesting that the pushing dynamics can be approximated as a function of just the current state and current input action. Due to sliding motion and the MuSHR car's nonholonomic constraint, we developed a long horizon planner using the friction cone and limit surface analytical models under the assumption of quasistatic pushing. We found that the stable pushing MPC performs best compared to the LSTM and GP based MPCs on our metrics when tracking planner outputted trajectories. However, the LSTM based MPC adapts better to initial sliding displacements by stabilizing the block at the center of the pusher displayed in table 5.4. Thus, we conclude that the LSTM based MPC is more robust and a better choice for a general purpose pushing controller for non-stable pushing trajectories.

In addition, our planner integrated with the stable pushing MPC was successfully implemented in the real world with minor differences in the performance suggesting that our simulation environment is realistic. Through data collected from real world experiments, we showed in Fig 5.5 that the planner's quasistatic assumption is reasonable. Thus, it safe to plan for the car using the constrained set of stable pushes assuming the block is glued to the car.

6.1 Limitations

Our planner currently consists of only large motion primitives which lead to long and complex trajectories. This requires more space for performing pushing experiments in the real world and since we collect pose data using mocap, our experiments are bound within a fixed area given by the cameras' field of vision. This limit's our current set of pushing experiments to a very small subset of possible pushing experiments.

Also, the LSTM and GP based MPCs are computationally inefficient and cannot be run in real-time without the use of large GPUs which cannot be mounted on and powered by our small MuSHR race car.

Since we determine the success of an experiment as successfully pushing the block to the goal position without losing contact with the pusher, we do not incorporate for replanning if we lose the block.

6.2 Future Work

To reduce required real world space for pushing experiments, we need to incorporate smaller motion primitives within our stable motion primitves. Generating smaller motion primitives isn't challenging, however they require the MPC to track the trajectory more closely. We've observed that the block misses the goal position if the MPC doesn't track the trajectories well. This also implies that we need to design a more effective cost function for our MPC framework.

In addition, we can reconfigure the MuSHR car to a different side of the block to generate smaller trajectories. We intend to develop a planner that uses reconfiguration as a motion primitive.

Lastly, we need to make the LSTM model computationally efficient to run in real-time.

Some common methods include pruning smaller magnitude weights [Toole, 2021] or using a GPU to improve LSTM forward propagation run time. Also, a Transformer model [Vaswani et al.] is a better choice compared to an LSTM, since they are faster, proven to perform better at retaining information using self-attention and require less data to train.

BIBLIOGRAPHY

P. Agarwal, J.-C. Latombe, R. Motwani, and P. Raghavan. Nonholonomic path planning for pushing a disk among obstacles. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3124–3129, Albuquerque, NM, USA, 1997. IEEE. ISBN 978-0-7803-3612-4. doi: 10.1109/ROBOT.1997.606763. URL http://ieeexplore.ieee.org/document/606763/.

J. Busch, J. Pi, and T. Seidl. PushNet: Efficient and Adaptive Neural Message Passing. *arXiv:2003.02228 [cs, stat]*, Dec. 2020. doi: 10.3233/FAIA200199. URL http://arxiv.org/abs/2003.02228. arXiv: 2003.02228.

D. Cappelleri, J. Fink, B. Mukundakrishnan, V. Kumar, and J. Trinkle. Designing open-loop plans for planar micro-manipulation. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 637–642, Orlando, FL, USA, 2006. IEEE. ISBN 978-0-7803-9505-3. doi: 10.1109/ROBOT.2006.1641782. URL http://ieeexplore.ieee.org/document/1641782/.

M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A Compositional Object-Based Approach to Learning Physical Dynamics. *arXiv:1612.00341 [cs]*, Mar. 2017. URL http://arxiv.org/abs/1612.00341. arXiv: 1612.00341.

M. R. Dogar and S. S. Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2123–2130, Taipei, Oct. 2010. IEEE. ISBN 978-1-4244-6674-0. doi: 10.1109/IROS.2010.5652970. URL http://ieeexplore.ieee.org/document/5652970/.

R. Dolphin. LSTM Networks | A Detailed Explanation, Mar. 2021. URL https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9.

K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning Visual Predictive Models of Physics for Playing Billiards. *arXiv:1511.07404 [cs]*, Jan. 2016. URL http://arxiv.org/abs/1511.07404. arXiv: 1511.07404.

C. E. García, D. M. Prett, and M. Morari. Model predictive control: Theory and practice—A survey. *Automatica*, 25(3):335–348, May 1989. ISSN 0005-1098. doi: 10.1016/0005-1098(89)90002-2. URL https://www.sciencedirect.com/science/article/pii/0005109889900022.

J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

S. Goyal, A. Ruina, and J. Papadopoulos. Planar sliding with dry friction Part 1. Limit surface and moment function. *Wear*, 143(2):307–330, Mar. 1991. ISSN 0043-1648. doi: 10.1016/0043-1648(91)90104-3. URL https://www.sciencedirect.com/science/article/pii/0043164891901043.

J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, Dec. 1964. ISSN 0001-0782, 1557-7317. doi: 10.1145/355588.365104. URL https://dl.acm.org/doi/10.1145/355588.365104.

Jiaji Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377, Stockholm, Sweden, May 2016. IEEE. ISBN 978-1-4673-8026-3. doi: 10.1109/ICRA.2016.7487155. URL http://ieeexplore.ieee.org/document/7487155/.

D. Jurafsky and J. H. Martin. *Speech and language processing: An introduction to speech recognition, computational linguistics and natural language processing*. Prentice Hall, Upper Saddle River, NJ, 2008.

G. Lee, T. Lozano-Pérez, and L. P. Kaelbling. Hierarchical planning for multi-contact non-prehensile manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 264–271, Sept. 2015. doi: 10.1109/IROS.2015.7353384.

S. H. Lee and M. R. Cutkosky. Fixture Planning With Friction. *Journal of Engineering for Industry*, 113(3):320–327, Aug. 1991. ISSN 0022-0817. doi: 10.1115/1.2899703. URL https://doi.org/10.1115/1.2899703.

M. Likhachev. Search-based Planning with Motion Primitives. page 86, 2010.

K. Lynch. The mechanics of fine manipulation by pushing. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2269–2276, Nice, France, 1992. IEEE Comput. Soc. Press. ISBN 978-0-8186-2720-0. doi: 10.1109/ROBOT.1992.219921. URL http://ieeexplore.ieee.org/document/219921/.

K. Lynch. Estimating the friction parameters of pushed objects. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, volume 1, pages 186–193 vol.1, July 1993. doi: 10.1109/IROS.1993.583097.

K. M. Lynch and M. T. Mason. Stable Pushing: Mechanics, Controllability, and Planning. *The International Journal of Robotics Research*, 15(6):533–556, Dec. 1996. ISSN 0278-3649. doi: 10.1177/027836499601500602. URL https://doi.org/10.1177/027836499601500602.

M. Mason. Manipulator grasping and pushing operations, 1982.

M. T. Mason. Mechanics and Planning of Manipulator Pushing Operations. *The International Journal of Robotics Research*, 5(3):53–71, Sept. 1986. ISSN 0278-3649. doi: 10.1177/027836498600500303. URL https://doi.org/10.1177/027836498600500303.

M. T. Mason. Compliant sliding of a block along a wall. In V. Hayward and O. Khatib, editors, *Experimental Robotics I*, volume 139, pages 568–578. Springer-Verlag, Berlin/Heidelberg, 1990. ISBN 978-3-540-52182-2. doi: 10.1007/BFb0042542. URL http://link.springer.com/10.1007/BFb0042542.

Michael. Vehicle Dynamics: The Kinematic Bicycle Model, Sept. 2020. URL https://thef1clan.com/2020/09/21/vehicle-dynamics-the-kinematic-bicycle-model/.

K. Miyazawa, Y. Maeda, and T. Arai. Planning of graspless manipulation based on rapidly-exploring random trees. In *(ISATP 2005). The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005.*, pages 7–12, July 2005. doi: 10.1109/ISATP.2005.1511442.

B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *2012 IEEE International Conference on Robotics and Automation*, pages 4373–4378, May 2012. doi: 10.1109/ICRA.2012.6225042. ISSN: 1050-4729.

S. Narasimhan. Task level strategies for robots.

M. Peshkin and A. Sanderson. The motion of a pushed, sliding workpiece. *IEEE Journal on Robotics and Automation*, 4(6):569–598, Dec. 1988a. ISSN 2374-8710. doi: 10.1109/56.9297.

M. Peshkin and A. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal on Robotics and Automation*, 4(5):524–531, Oct. 1988b. ISSN 2374-8710. doi: 10.1109/56.20437.

Pytorch. SmoothL1Loss — PyTorch 1.9.0 documentation, 2019. URL https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html.

S. Rao. A course in Time Series Analysis. page 512, January 2021.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9. OCLC: ocm61285753.

J. Richalet, A. Rault, J. L. Testud, and J. Papon. Paper: Model predictive heuristic control. *Automatica (Journal of IFAC)*, 14(5):413–428, Sept. 1978. ISSN 0005-1098. doi: 10.1016/0005-1098(78)90001-8. URL https://doi.org/10.1016/0005-1098(78)90001-8.

M. R. C. Robert D. Howe. Practical Force-Motion Models for Sliding Manipulation, 1996. URL https://journals.sagepub.com/doi/abs/10.1177/027836499601500603?casa_token=m8O8pNnhVD4AAAAA:Yhzj9jzltilDQWCGyXAHOCtNJV3n_bfEPgACLo5e2lTPTqLt2e5sKNVsbPB_hwixc6L3CWkeiJ5XYA.

M. Rockett. A data-driven model predictive controller for quasistatic nonholonomic pushing, 2020.

sbpl. Home | Search-Based Planning Lab, 2007. URL http://www.sbpl.net/.

S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Chouhury, C. Mavrogiannis, and F. Sadeghi. MuSHR: A low-cost, open-source robotic racecar for education and research. *CoRR*, abs/1908.08031, 2019.

J. Stüber, M. Kopicki, and C. Zito. Feature-Based Transfer Learning for Robotic Push Manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5643–5650, May 2018. doi: 10.1109/ICRA.2018.8460989. ISSN: 2577-087X.

J. Stüber, C. Zito, and R. Stolkin. Let's Push Things Forward: A Survey on Robot Pushing. *Frontiers in Robotics and AI*, 7:8, Feb. 2020. ISSN 2296-9144. doi: 10.3389/frobt.2020.00008. URL http://arxiv.org/abs/1905.05138. arXiv: 1905.05138.

E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Vilamoura-Algarve, Portugal, Oct. 2012. IEEE. ISBN 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1. doi: 10.1109/IROS.2012.6386109. URL http://ieeexplore.ieee.org/document/6386109/.

J. Toole. Deep learning has a size problem, Apr. 2021. URL https://heartbeat.fritz.ai/deep-learning-has-a-size-problem-ea601304cd8.

E. Ugur. Goal emulation and planning in perceptual space using learned affordances - ScienceDirect, 2011. URL https://www.sciencedirect.com/science/article/pii/S0921889011000741?casa_token=qyHTdltBK3EAAAAA:BnEGvEMVTCBgDk77r5pzSo960pLMaltU63GnZw6ppsWndMdSOLBtpEY_AJJJmzSCZUGdJqb0vFE.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin. Attention is All you Need. page 11.

N. Watters, A. Tacchetti, T. Weber, R. Pascanu, P. Battaglia, and D. Zoran. Visual Interaction Networks. *arXiv:1706.01433 [cs]*, June 2017. URL http://arxiv.org/abs/1706.01433. arXiv: 1706.01433.

T. Yoshikawa and M. Kurisu. Indentification of the center of friction from pushing an object by a mobile robot. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pages 449–454 vol.2, Nov. 1991. doi: 10.1109/IROS.1991.174510.

K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez. More than a million ways to be pushed. A high-fidelity experimental dataset of planar pushing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 30–37, Daejeon, South Korea, Oct. 2016. IEEE. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7758091. URL http://ieeexplore.ieee.org/document/7758091/.

A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning. *arXiv:1803.09956 [cs, stat]*, Sept. 2018. URL http://arxiv.org/abs/1803.09956. arXiv: 1803.09956.

Appendix A

SOFTWARE TOOLS

The code for this thesis can be found in four parts.

1. MuJoCo Simulation environment in Python (https://github.com/prl-mushr/mushr_push_sim.git)

   Code to create a simulation environment for pushing experiments using OpenAI's mujoco_py package.

2. Model training (https://github.com/Alrick11/mushr_push_mstp.git)

   Contains Data collection and code for training the LSTM and GP models.

3. Planner (https://github.com/schmittlema/pysbpl.git)

   Contains code for the graph based planner.

4. Pushing (https://github.com/schmittlema/pushr.git)

   Integrates the planner and MPC to perform pushing experiments both in simulation and the real world. Also includes code for analysing bagfiles to evaluate and plot metrics.