Title
Developing Combinatorial Optimization and Data-driven Methods for Multi-modal Motion Planning

Author
Lin, Xuan

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Developing Combinatorial Optimization and Data-driven Methods for Multi-modal Motion Planning

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mechanical Engineering

by

Xuan Lin

2022

ABSTRACT OF THE DISSERTATION

Developing Combinatorial Optimization and Data-driven Methods for Multi-modal Motion Planning

by

Xuan Lin
Doctor of Philosophy in Mechanical Engineering
University of California, Los Angeles, 2022
Professor Dennis W. Hong, Chair

Legged robots require fast and reliable motion planners and controllers to satisfy real-time implementation requirements. In this dissertation, we investigate the model-based motion planning and control techniques for robotics problems involving contact, including multi-legged robot walking and vertical climbing, item manipulation inside a cluttered environment, and self-reconfigurable robot systems. Each of them can be formulated into a mixed-integer nonlinear (non-convex) program problem for optimization solvers to resolve.

In general, mixed-integer nonconvex programs are challenging to solve. In this dissertation, we adopted several approaches including the decoupling approach, coupled approaches such as ADMM, and data-driven approaches. In the end, we benchmark the performance of the proposed approaches on the bookshelf manipulation problem. Through comparison of various approaches, we show that the data-driven approach can potentially achieve a high success rate, fast solving speed, and good objective function value, given that the new problem is within the trained distribution. Planned trajectories are validated on the hardware showing the planner's capability of generating real-world feasible trajectories.

The dissertation of Xuan Lin is approved.

Lieven Vandenberghe

Veronica Santos

Tetsuya Iwasaki

Dennis W. Hong, Committee Chair

University of California, Los Angeles

2022

To my amazing labmates, robots, and family members.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author would like to say thank you to:

My advisor, Dr. Dennis Hong, who has been giving me unconditional support during the years of my Ph.D. in RoMeLa.

My amazing labmates, who has given me support about my work, and all kinds of things outside the lab. There are too many names, so do not put them here in case I leave out some.

My Ph.D. committee members, who helped me with my Ph.D. journey.

All professors who had given me great lectures about different subjects in which I am interested, especially outside the robotics and control fields.

My family members who keep on discussing with me about my plans.

And finally, I want to say thank you to myself. It has been a great challenge to overcome all the hurdles and reach this point. But you have been doing a great job. There are even bigger obstacles to appear. Be confident, you can take them down.

VITA

2013–2015 M.S., University of California Los Angeles

2009–2013 B.S., Harbin Institute of Technology

PUBLICATIONS

1) Xuan Lin, et al. "Multi-Limbed Robot Vertical Two Wall Climbing Based on Static Indeterminacy Modeling and Feasibility Region Analysis." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.
2) Xuan Lin, et al. "Optimization Based Motion Planning for Multi-Limbed Vertical Climbing Robots." 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019.
3) Y Shirai, X Lin, Y Tanaka, A Mehta, D Hong, "Risk-Aware Motion Planning for a Limbed Robot with Stochastic Gripping Forces Using Nonlinear Programming." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)/IEEE Robotics and Automation Letters 5 (4), 4994-5001
4) Xuan Lin, Gabriel I. Fernandez, Dennis W. Hong , "ReDUCE: Reformulation of Mixed Integer Programs using Data from Unsupervised Clusters for Learning Efficient Strategies" 2022 IEEE International Conference on Robotics and Automation (ICRA)
5) Shirai, Yuki, Xuan Lin, Ankur Mehta, and Dennis Hong. "LTO: Lazy Trajectory Optimization with Graph-Search Planning for High DOF Robots in Cluttered Environments." 2021 IEEE International Conference on Robotics and Automation (ICRA)

6) Xuan Lin, Min Sung Ahn, and Dennis W Hong, "Designing Multi-Stage Coupled Convex Programming with Data-Driven McCormick Envelope Relaxations for Motion Planning" 2021 IEEE International Conference on Robotics and Automation (ICRA)

7) Jingwen Zhang, Xuan Lin, and Dennis W Hong, "Transition Motion Planning for Multi-Limbed Vertical Climbing Robots Using Complementarity Constraints" 2021 IEEE International Conference on Robotics and Automation (ICRA)

8) Yusuke Tanaka, Yuki Shirai, Zachary Lacey, Xuan Lin, Jane Liu, Dennis Hong, "An Under-Actuated Whippletree Mechanism Gripper based on Multi-Objective Design Optimization with Auto-Tuned Weights" 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021.

9) Xuan Lin, Gabriel I. Fernandez, Dennis W. Hong, "Multi-Modal Multi-Agent Optimization for LIMMS, A Modular Robotics Approach to Delivery Automation." 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

10) Yuki Shirai, Xuan Lin, Alexander Schperberg, Yusuke Tanaka, Hayato Kato, Varit Vichathorn, and Dennis Hong, "Simultaneous Efficient Contact-Rich Grasping and Locomotion Optimization Enabling Free-Climbing for Multi-Limbed Robots." 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

11) Yusuke Tanaka, Xuan Lin, Yuki Shirai, Alexander Schperberg, Hayato Kato, Alexander Swerdlow, Naoya Kumagai, and Dennis Hong, "SCALER: A Tough Versatile Quadruped Free-Climber Robot." 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

CHAPTER 1

Introduction

1.1 Optimization and Learning based Motion Planning

Robots need to plan and control their motions to complete given tasks autonomously. During the process, Legged robots or manipulators will need to constantly interact with the environment such as to control the body posture, manipulate the objects, make connections between robot modules, etc. In practical applications, the problem usually comes with solving time limitations, but can easily scale up. Our current tools of discrete optimization do not generally meet those requirements even when the problem scale becomes slightly larger, e.g. robot model becomes more complex, or the number of agents becomes larger. In this thesis, we first develop optimization-based motion planning techniques to solve motion planning problems, mostly for legged robots, and implement the trajectories on the actual hardware. We then combine learning methods to speed up the optimization solving process for a few combinatorial optimization problems coming from practical robotics applications. This framework is developed in the hope that it may be scalable to more complicated combinatorial motion planning problems.

Common method to tackle motion planning problems include graph search methods [1], Lyapunov-function-based nonlinear control based methods [2, 3], sampling based methods [4], optimization based methods [5–12], and learning based methods [13, 14]. In addition, researchers exploit combinations of the aforementioned methods to tackle complex problems [15]. For example, many of the classical graph search and control based methods suffer when the di-

mension of the problem becomes high. This may be remedied by optimization based methods [10].

As a relatively new technique, optimization based methods are being used to plan motion for multi-robot systems [16], legged robots [5–12], self-driving cars [17], and so on. Typical optimization based approaches such as mixed-integer convex programs (MICPs) [7, 18], nonlinear or nonconvex programs (NLPs) [6, 9, 19] and mixed-integer NLPs (MINLPs) [20] offer powerful tools to formulate motion planning problems. Those formulations are handed to the solvers for solutions. However, each optimization scheme has its own drawbacks. NLPs tend to suffer from local optimal solutions. In practice, local optimal solutions can sometimes have bad properties, such as inconsistent behavior as they depend on initial guesses. Mixed-integer programs (MIPs) are a type of NP-hard problem. Branch-and-bound is usually used to solve MIPs [21]. MIP solvers seek global optimal solutions, therefore, having more consistent behavior than NLP solvers. For small-scale problems, these algorithms usually find optimal solutions within a reasonable time [7, 22]. On the contrary, MIPs can require impractically long solving times for problems with a large number of integer variables [23]. MINLPs incorporate both integer variables and nonlinear constraints, hence, very expressive. Unfortunately, we lack efficient algorithms to tackle MINLPs. Many practical problems require a solving speed of at most a few seconds. As a result, it is difficult to implement most of the optimization schemes online for larger-scale problems.

Learning based methods are typically used to identify features and discover heuristics that help to find solutions for a rather complicated systems. Typical learning methods includes supervised learning, unsupervised learning, reinforcement learning, or imitation learning. Given sufficient training data, supervised learning can be used to train a map from the problem feature to the robot walking gait [24], optimal cost-to-go [25, 26], mixed-integer strategies

[27,28], lyapunov functions for stabilizing controller [29], and so on. Unsupervised learning methods are relatively less explored, but can be used to identify modes from data and construct piece-wise convex functions. Typical usage can be seen in piece-wise affine (PWA) system identification [30–33], and is recently expended to solve nonlinear programming problems [34]. The difference of using clustering methods for system identification versus solving nonlinear optimization is that the former cares clusters in the parameter space while the latter cares clusters in the solution space. Reinforcement learning methods has been explored for locomotion [35, 36], manipulation [37] and has demonstrated impressive performance on real hardware. However, the data efficiency, convergence guarantee are still among many open and challenging problems along this line.

Recently, researchers explore learning methods to help the optimization solvers, mixed-integer optimization solvers in particular, to find solution faster [27, 28, 38–40]. These approaches help to implement integer programmings from middle to large scale onto robotic systems that have solving speed requirements.

1.2 Multi-model Legged Walking and Climbing Robots

Walking and vertical wall climbing robots are applicable in many situations, such as surveillance, search and rescue, and building maintenance. Since wheeled vehicles can move fast on flat surfaces, wheeled robots have been experimented with for wall climbing [41] [42]. Like many wheeled robots, non-legged wall-climbing robots are impeded by uneven surfaces, limiting their capabilities. Many animals found in nature demonstrate fast and agile climbing with their limbs. Legged animals can climb up highly unstructured environments as well as traverse on ground. They are also able to jump onto and grab structures

using only their hands, demonstrating highly mobile motions that non-legged robots cannot even attempt. Much of the research on climbing robots started by mimicking animals [43] [44], and then gradually the systems became more complex. The most recent advancement is [45] which presented a 35 kg robot with 4, 7-degree-of-freedom limbs, climbing on smooth surfaces with a gecko type gripper and rough surfaces with micro-spine gripper. When generating climbing motions, many researchers in the field resort to templates [46] [47]. Templates are used to study the dynamics of climbing. Its implementation is currently limited to light, low degree-of-freedom robots. The climbing motion for more complex, high degree-of-freedom robots is still quasi-static [45]. When climbing an environment not seen before, the robot needs to carefully plan its steps and torque based on the environment to find a trajectory to reach its objective. Thus, wall climbing becomes a motion planning problem. This line of work was started by [4], which presented an algorithm based on a more classical graph search method listed in the previous section.

1.3 Contributions

This dissertation proposes methods to formulate various problems including multi-limbed robot walking and climbing, item manipulation inside a cluttered environment, and self-reconfigurable robot system, into mixed-integer nonlinear (non-convex) programs (MINLPs). First, a model of the proposed robotic system is formulated including kinematics, dynamics, or compliance. Next, the model is resolved by optimization solvers. Different approaches such as decoupling, ADMM, and data-driven approaches are proposed and implemented. Finally, we benchmark those methods on a bookshelf organization problem to understand their performances such as capability to find a feasible solution, solving speed, and optimality. For each of the problems mentioned above, we implemented the planned trajectories on the actual robot hardware

to demonstrate real-world feasibility.

1.4 Outline

In chapter 3, the design of the robot being used throughout this series of research is presented. The state estimation and control methods for this robot are also presented. The robot is tested with various tasks such as open loop walking, weight lifting, and walking on uneven terrains for its strength, speed, and stability. There are two versions of this multi-legged robot: one six-legged older version (SiLVIA), and one four-legged newer version (SCALER). This dissertation will focus on SiLVIA. Some results will be demonstrated for SCALER. In chapter 4, a robot whole-body model considering joint compliance is presented. This model allows calculations for the joint and body center of mass deflections. In particular, it allows the calculations for contact forces when the robot is climbing between two walls. Hardware testing results for simple two-wall climbing are shown. In chapter 5, we formulate the mixed-integer programs to tackle the motion planning tasks for the multi-legged robot to climb up on uneven terrains and avoid obstacles. We also investigated planning transition gait from the ground to the wall using complementary constraints, and use data to speed up mixed-integer planner with envelope constraints. In 6, we presented a motion planner on a self-reconfigurable robot system named LIMMS, which is a larger-scale problem than robot climbing. We investigated ADMM as a coupled kinodynamic planning approach. In each chapter, the hardware demonstrations are included.

From chapter 7, we investigated data-driven methods to further improve the feasibility, optimality and solving speed of the optimization schemes. In chapter 8, we applied the data-driven methods to two problems involving contact. One is the bookshelf organization problem, which is an item manipula-

tion problem involving contact. The other one is a mixed-integer non-convex model-predictive control problem on SCALER. We showed the benchmark results using various proposed methods with the bookshelf problem. Finally, chapter 9 concludes the thesis and discusses the ongoing work and future works.

CHAPTER 2

Background

2.1 Model Based Motion Planning for Multi-limed Robots

Multi-limbed robots use discrete contact points to navigate through the environment. Compared to their wheeled counterparts, they are less limited by the type of terrain and are more mobile in uneven or discrete terrain such as stairs. However, since the trajectory of limbed robots is composed of discrete contact points as well as the order of limb sequence (gaits), continuous path planning techniques such as potential field-based methods [48] that are used for wheeled robots do not directly apply. The current literature typically use graph search methods [1], integer programming methods [18], linear complementary planning methods [49] or learning methods [50, 51] to treat the gait and contact point planning.

In addition to kinematics, it is also important to plan contact forces on each contact point as this is typically important for the stability of legged robots. To plan contact forces, a dynamic robot model is typically required. For bipedal or quadruped robots with trot gait, the intrinsic dynamics is unstable, and commonly used models include ZMP, capture point, and SLIP models. For hexapod robots, a dynamic model is usually not required since the intrinsic dynamics is stable for tripod gaits. On the other hand, if the limbed robots are subject to heavy load, non-negligible deformation can happen to the robot body and legs. In this case, the whole body compliance model is considered, as introduced in the next chapter.

2.1.1 Graph Based Planning

Graph-based motion planning operates on graph models of the environment or physics. These methods naturally fit the tasks of legged robot footstep planning as legs make discrete contacts to the environment. Typical algorithms include A^* [52], ARA^* [53], D^* [54], R^* [55], and so on. The footstep planning work done on ASIMO [56] first introduced A^* for bipedal robot footstep planning. Later work includes [1] which proposed A^* algorithm which allows partial feet contact implemented on ATLAS and Valkyrie humanoid robots. Using graph search based methods to plan dynamics is not straightforward, hence it is somewhat rare to see this line of research include robot COM dynamics as stated in [1]. One approach will be designing simple heuristics for body dynamics and introducing the cost function.

2.1.2 Sampling based Planning

Graph based planning methods work well for low-dimensional problems. However, as the method requires gridding the configuration space, as the dimension of the problem gets higher, the number of grids required for solving the problem grows exponentially. This so-called curse of dimensionality makes many grid-based methods unable to solve the problem with configuration space more than tens. Sampling based methods such as PRM [57], RRT [58] are invented to deal with static motion planning problems in relatively higher configuration space. This set of methods uses samples to speed up the exploration of space and connect the samples in a collision-free manner, resulting in a map (PRM) or a tree (RRT). Sampling based methods will work well even with a dimension of configuration space at tens. Problems such as 6-7 DoF manipulator motion planning are friendly to this method. When first proposed, they focus on finding a feasible trajectory. PRM* and RRT* methods [59] are later

proposed which keeps on finding more optimal trajectories. One drawback of these methods is that it relies on efficient samples to discover feasible solutions. If the configuration space contains narrow tunnels (this tends to happen when equality constraints appear), this chance of getting feasible samples dramatically decreases. These algorithms also suffer from planning problems with dynamics as the sampling efficiency also drops significantly. One approach to improve sampling efficiency relies on simulating the system forward [60].

2.1.3 MICP Gait and Whole body Planning

Another common approach to planning footstep positions is to use mixed-integer programs. Contact planning includes a series of on-off constraints which can naturally be formulated with binary variables and big-M formulation. If those binary variables are relaxed into continuous variables and the problem is convex after relaxation, the formulation is mixed-integer convex. There exist efficient off-the-shelf solvers to solve mixed-integer convex programs such as Gurobi, CPLEX, etc. This line of work is started by [25], where mixed-integer quadratic programming is used to plan the footsteps given collision-free contact regions followed by a nonlinear program for the upper body motion. The upper body motion with pure kinematics can easily be incorporated into MIQP formulation [7]. Full kinematics can be incorporated using the approach proposed by [61]. If the problem scale is small, the solving time for MIPs is relatively fast. However, the MIP worst solving time grows exponentially as the number of discrete variables gets larger. Moreover, the solving time is dramatically slower after introducing the nonlinear dynamic constraints which are typically approximated with McCormick envelopes [62]. For a problem with a reasonable scale, MIPs with envelopes can take hours to solve [23]. In this thesis, we try to introduce learning methods as a remedy.

2.1.4 NLP and LCS Planning

Nonlinear programs (NLPs) are the most natural approach to dealing with nonlinear dynamics constraints. NLPs solvers typically use sequential quadratic programs (SQPs) or interior point based methods which do not guarantee to find a feasible solution and will likely result in a local optimal if it does find the solution. On the other hand, NLPs typically can be solved faster than the MIP formulation of the same problem [61]. If contacts exist, NLPs can incorporate complementary constraints which may be termed linear complementary systems (LCS) if other constraints are linear. Complementary constraints enforce orthogonality between continuous variables, effectively making them discrete. However, complementary constraints are known to be numerically difficult for NLP solvers hence special treatments are typically required [63].

This line of work includes [19] which solves robot motion plans with full kinematics and center of mass dynamics. [64] which incorporates complementary constraints for contact planning, and [6] which plans contact time for each leg. A more recent work implements NLP formulations online [65]. This formulation is simplified to minimize the nonlinearities and added a regularization term through offline extracting heuristics from simulation data [66]. A non-simplified NLP formulation for walking is still too slow to solve online for MPC. In this situation, one can simplify the model and implement convex MPC [67].

2.1.5 Reinforcement Learning Based Planning

Reinforcement learning (RL) based methods have draw more attention from robotics researchers and have been implemented on real hardware for quadruped robots [35, 68], bipedal robots [36, 69, 70], and manipulation [37, 71, 72]. If the RL controller works on the actual hardware, it can demonstrate good perfor-

mance which other controllers would take effort to realize. However, RL-based controllers are usually nontrivial to implement on the hardware. For example, [35] specifically trained a neural network to represent the actuator dynamics and use that network to train policies in the simulator. [68] trained a teacher policy network and a student policy network. It is mentioned in this paper that training a rough-terrain locomotion policy directly via RL was not successful due to sparse reward. In [69] the RL policy network is trained to mimic reference trajectories which are collected offline using direct collocation methods. Moreover, RL-based methods usually require a large amount of data for training. For robotics problems, the most practical approach to collect those data would be to run the simulation. However, simulators introduce additional sim-to-real gaps as almost none of the current simulators represents precisely the parameters on the actual hardware.

2.1.6 Supervised Learning Based Planning

Reinforcement learning methods require the learning agents to actively explore the action space to collect good trajectories. On the other hand, data can be collected in advance offline, then a learning agent can be trained to operate online. Typical approaches include training to mimic certain parameters in the trajectory. For example, [24, 73] uses supervised learning to map feedback signals to the Bezier coefficients of the desired trajectory. Another approach is to train an agent to partially solve a combinatorial optimization problem. For example, [39] solves a mixed-integer program by training a network to propose candidate integer variables and solve multiple convex optimizations online.

2.1.7 Template based Dynamic Planning, ZMP, ICP, CWC

For planning with dynamic systems, the aforementioned methods for static motion planning tend to suffer. Effectively, a dynamic system has different obstacle constraints defined differently at any location in the configuration space. When dealing with this problem, researchers usually try to simplify the system dynamics into so-called templates which can hopefully be resolved with regular controller design techniques like LQR.

For the problem of legged robots locomotion, typical templates used are the rimless wheel, inverted pendulum, spring-loaded inverted pendulum (SLIP), and so on [74]. These simplified dynamic models help to capture the essential part of robot dynamics, design simple controllers, prove stability. The most famous example probably is the Raibert controller on the SLIP model [75]. The Raibert controller utilizes 2 control inputs: linear spring position and hip angle, to control a 2D hopping robot. Simple as it is, it has shown impressive dynamic performance on robots back in the 80s. Even nowadays, these heuristics are still used to plan the leg footstep positions [76,77].

Aside from templates, another approach to get simple controller heuristics is to investigate rigid body dynamics for the whole robot (ignore e.g. leg swing dynamics) and make assumptions. A good example that is also very successful is zero moment point (ZMP) [78]. ZMP uses rigid body dynamics assuming the flat ground, constant center of mass height, and foot does not collide with the ground. With those assumptions, the dynamic equations become linear permitting simple stability criterion.

ZMP can be used to plan dynamic motion for bipedal robots or quadruped robots. Hexapods usually have more than 3 contact points on the ground, hence do not need such a stability rule. Usually, it is sufficient to guarantee stability for hexapods if the center of mass is within the support polygon. If

the terrain is not flat, an extended support polygon can be used [4].

Instantaneous capture point (ICP) [79] which computes the position to make a step to recover the body posture based on the linear inverted pendulum model. Based on the principle to regulate the orbital energy to zero after making one step, criteria can be obtained indicating when to take a step (when the capture point is outside the convex hull of the foot support area), where to take a step (the base of support covers the capture point), and how many steps are required. With a linear inverted pendulum model plus a flywheel, the capture point will grow to a capture region. However, for quadruped robots and hexapods which are more stable than bipedal robots, capture point based methods can provide heuristics but are seldom used as principle control methods [80].

Another stability criterion more general than ZMP is the contact wrench cone (CWC) [81, 82]. In fact, the most general criterion for contact failure should be the frictional failure of the contact force field on the foot. The ZMP criterion effectively simplifies this assuming the robot walks on the flat ground and ignores slipping failure. In [83], the authors have shown that the failure of the contact force field is equivalent to the failure of the contact wrench cones on each of the vertex of the convex foot shape. This yields more general stability criteria that can work on the non-flat terrain as well as taking into account the sliding failure, but can still be relatively easy to compute [82]. For typical quadrupeds or hexapods with point foot contact, this criterion is reduced to simple friction cone failure criterion. However, for climbing robots that equip grippers, this failure criteria can still be used.

2.1.8 Shooting methods through iterative local approximation - DDP/iLQR

One set of methods that takes use of the advantage of close-form control solution for LQR problems is the differential dynamic programming (DDP) [84] and iterative LQR (iLQR) [85] methods. Those methods would make the quadratic approximation of the value functions and linear approximation of the dynamics given an initial trajectory. As a result, the LQR problem has a closed-form solution, which is then integrated forward to update the trajectory. The process is repeated until it converges. One limitation of the DDP-type methods is that it does not directly deal with the constraints, e.g. in the control signal domain. Some researches have been done to alleviate this limitation [86]. In addition, these methods are developed for smooth dynamics. To use them for legged robot motion planning, one will need to extend the algorithm to hybrid systems. Works along this line came out recently using e.g. saltation matrix [87].

2.1.9 Lyapunov function based methods

Lyapunov function based method is a typical method for controller design on nonlinear systems. The basic idea is to search for a Lyapunov function that is positive definite by itself but the derivative is negative definite, resembling the energy of the system. If those conditions can be satisfied, we not only find valid control inputs but also rigorously prove the stability of the controller as well as retrieve its region of attraction.

The challenge of Lyapunov function based methods is that since the algebraic form of the function is not explicitly defined, it is impossible to search over each candidate. One has to restrict the form of Lyapunov functions to a subset, typically quadratic forms. If the coefficients of the Lyapunov functions are predetermined, the searching problem is reduced to quadratic programs

(QPs) where the small-scale ones can be efficiently solved online for control inputs. These methods are named control Lyapunov functions (CLFs) [3, 88], control barrier functions (CBFs) [89]. These methods have been used on safety critical systems [90, 91], bipedal walking [92, 93], hopping [94], etc.

If one would like to search the control signal simultaneously with the Lyapunov functions, the optimization problems become nonlinear programs which are significantly more challenging. One approach is to use sum-of-squares programs (SOS) [95] and restrict the controllers and Lyapunov functions to polynomials.

Except for Lyapunov function based methods, other nonlinear control methods can also be used to do whole body control for walking robots with contact such as hybrid zero dynamics [96].

2.1.10 Combining different approaches

Previous sections provide various methods to resolve motion planning and control problems. Each of them has assets and drawbacks. To resolve more complicated problems, combinations of those methods are reasonable. For example, [97] combines LQR controller with randomized tree search (RRT) blending in information from the region of attraction analysis. [93] combines RRT* with control Lyapunov functions to search for generating feasible motions on uneven terrain. [98] proposes a hierarchical planning framework where a high-level DQN-based planner coordinates the low-level gradient-based controller to plan reactive manipulation. [99] developed a hierarchical planner where the high-level planner uses Random Possibility Graph that quickly explores potential actions to aid the low-level planner. [5] combines MICP for footstep planning and NLP for body dynamics planning to allow a humanoid robot to walk over uneven terrains in the DARPA robotics challenge. [10] com-

bines a high-level graph search method and low-level mixed-integer programming method to solve obstacle-free trajectories that improved the solving speed compared to a nominal MICP solver.

CHAPTER 3

Introduction to SiLVIA and SCALER

This chapter introduces the multi-legged robots that are used for walking and climbing tests throughout the research. There are two versions of the multi-legged robot. The older version is a six-legged robot, named SiLVIA: Six Legged Vehicle with Intelligent Articulation. The newer version is a four-legged robot, named SCALER: Spine Enhanced Climbing Autonomous Legged Exploration Robot. SiLVIA is used mostly for two-wall climbing. SCALER is used mainly for the research of climbing on a rock-climbing wall with spine-enhanced grippers and state estimation tests. The author led a team to design SiLVIA, and manufactured most of the Aluminum components. SCALER is mainly designed and manufactured by Yusuke Tanaka, although sharing the same code framework with SiLVIA.

Table 3.1: SiLVIA parameters

Parameter	Value
Degree of Freedom for Each Limb	3
Limb Coxa Length	57 [mm]
Limb Femur Length	195 [mm]
Limb Tibia Length	375 [mm]
Weight	10.3 [KG]
Motor Proportional Gain P	12
Motor Integral Gain I	0
Motor Derivative Gain D	0
Max Torque	25 [Nm]

Table 3.2: SCALER parameters

Parameter	Value	
	Walking	Climbing
Degree of freedom for each limb	3	7
Total weight	6.3 [kg]	8.8 [kg]
Body mass	2.5 [kg]	
One leg total mass	0.95 [kg]	1.57 [kg]
Motor pair max torque	10 [Nm]	

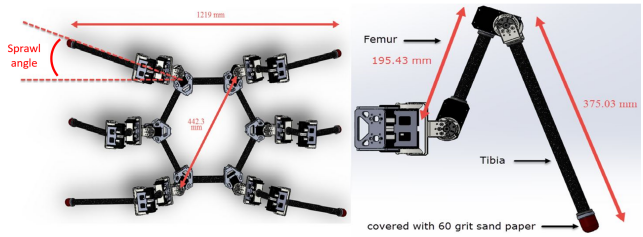


Figure 3.1: Hexapod robot body dimension and position

3.1 Design and Manufacturing

The robotic platform SiLVIA used in this study is a hexapod design with a central body frame and 6 limb assemblies, as shown in Fig. 3.1. The central body frame consists of aluminum brackets interconnected with carbon fiber tubes. Each limb has 3 degrees of freedom and consists of a coxa, an upper femur, and a lower tibia assembly. The tibia and femur assemblies are made with carbon fiber tubes and are connected by a dual motor assembly.

Thirty-six MX-106 motors have been used in pairs for actuation. The stall torque for each motor pair is approximately 25.0 Nm. The robot carries its own battery, computer, and IMU. It weighs 10.3 kg. The robot's end effectors are covered by 60 grit sand paper to enhance friction. The parameters of the robot are summarized in Table 3.1.

Another hardware platform used for climbing on a single rock-climbing wall is shown in Fig. 3.2. This platform has a quadruped design with each limb

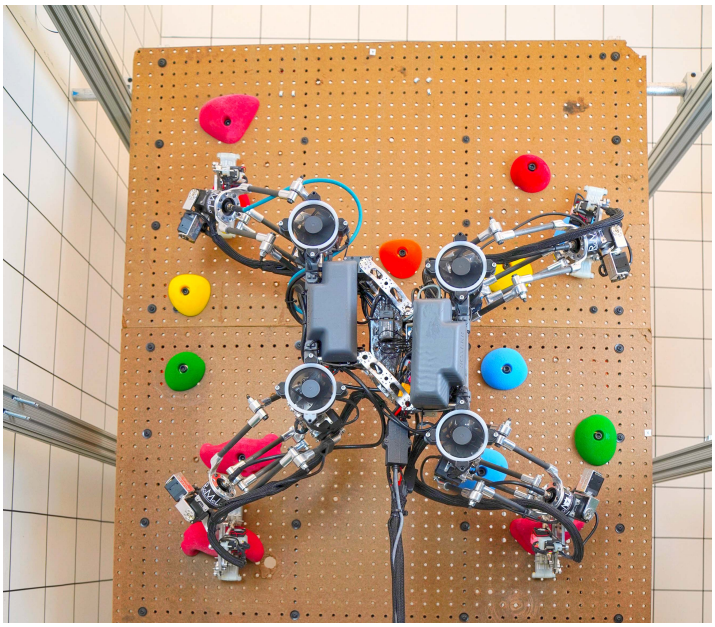


Figure 3.2: The quadruped robot SCALER climbing on the rock-climbing wall

a 5-bar-linkage to increase the leg strength. The strength of actuation is further improved with the paired Dynamixel XM430-350 motors which is light weight (160kg) but deliver more than 10Nm continuous torque. The SCALER robot body also has 1 degree of freedom which enlarges the workspace of each limb. When walking, each limb has 3 degree of freedom with actuators all placed nearby the shoulder to decrease the leg swinging momentum. When climbing, the wrist is equipped with additional 3 degree-of-freedom actuation and a gripper based on micro-spine design [100]. This type of gripper is very good at grasping onto rough surfaces. The wrist is also equipped with FT sensors allowing it to control the contact forces. SCALER weights 6.3kg for walking configuration, and 8.8kg for climbing configuration including grippers. It is able to carry over 3kg payload for climbing.

3.2 Software Construction

Both SiLVIA and SCALER are equipped with various sensors: force-torque sensors on the wrists (SCALER only), a body IMU sensor, Dynamixel actuators encoders, and sensors for the grippers (SCALER only), all of which are handled in a corresponding dedicated I/O thread.

The Dynamixel servos and GOAT gripper modules (SCALER only) are connected over RS-485 and to the CPU via a USB bus. Due to the large number of Dynamixel motors, the communication frequency with the encoders is capped to 150Hz, which is sufficient for the targeted tasks. All other sensors run at over 400Hz.

Joint space position control is either done on the Dynamixel actuator or on the CPU which runs our custom position controllers that generate position or velocity control inputs to the actuators. Though the communication frequency limits the control frequency, joint velocity control allows the robots to perform more aggressive motions.

3.2.1 State Estimation

State estimation is used for walking to provide full state feedback so that an MPC controller based on [67] can be used instead of explicit wrench references. Legged State Estimation(LSE) is implemented from [101] which takes IMU, F/T sensor, contact detection, and encoders along with kinematics, and provides pose and velocity estimations using an Extended Kalman filter. A diagram of software architecture is shown in Fig. 3.3. Body and foot trajectories can be generated either manually or by optimization-based planners [7] outside of the current SCALER software structure.

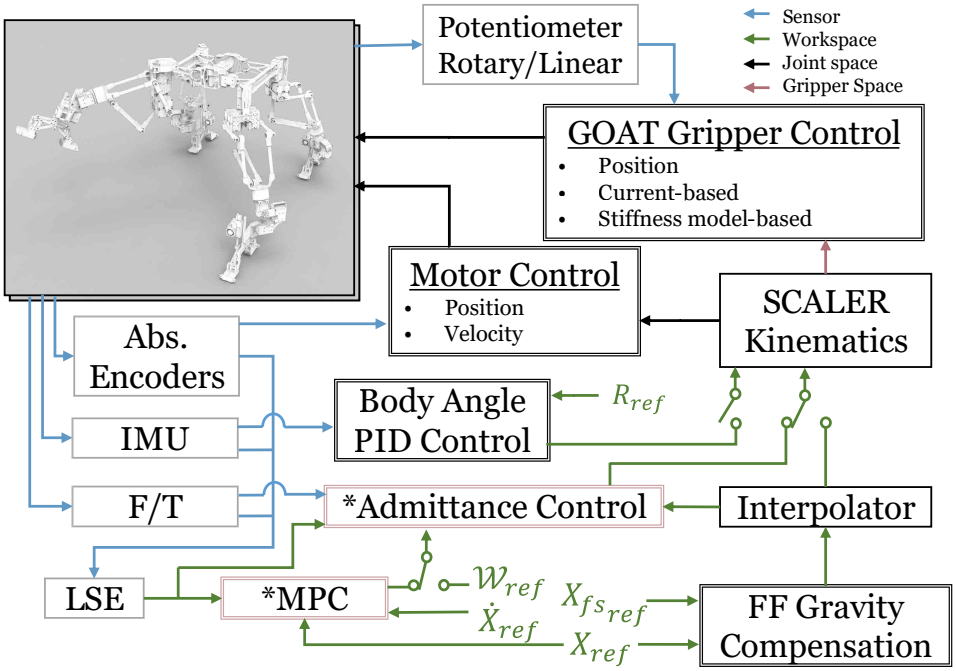


Figure 3.3: Software architecture for SCALER robot.

3.2.2 Control

Admittance control is implemented as operational space force control to track a contact reaction force or wrench trajectory. Specifically:

$$M_d\ddot{\mathbf{x}} + D_d\dot{\mathbf{x}} + K_d(\mathbf{x} - \mathbf{x}_0) = \mathbf{f}_{meas} - \mathbf{F}_{ref} \tag{3.1}$$

Where M_d, D_d and K_d are desired mass, damping and spring coefficients. $\mathbf{F}_{ref}$ is the wrench profile to be tracked. $\mathbf{x}$ is the position of the robot end-effector. The control output of (3.1) is $\ddot{\mathbf{x}}$ which is integrated allowing position or velocity control. When $K_d = 0$, (3.1) tracks the reference force profile $\mathbf{F}_{ref}$ which can be used to track contact force on the gripper. When $\mathbf{F}_{ref} = 0$, (3.1) tracks the reference position trajectory $\mathbf{x}_0(t)$ with compliance which mitigates the impact during walking. Due to gear backlash, the force control struggles to track rapidly changing force profiles (e.g., a square wave of more than 10 Hz). This control bandwidth suffices for low-speed tasks such as climbing that experi-

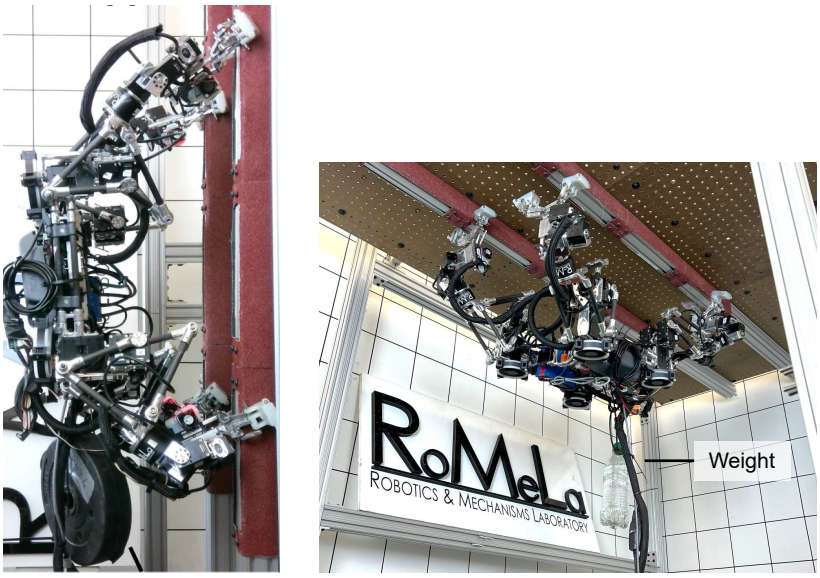
ence less frequent force reference changes. A technique to auto-tune the controller gains is proposed in [102].

3.3 Testings

In this section, we present results for open-loop tests where the robot demonstrated capabilities of robust locomotion without any feedback. These capabilities have been demonstrated before on the hexapod robot RHex [103] with a unique leg design. Despite its impressive terrain traverse capabilities, RHex's leg has only 1 degree of freedom hence incapable of tasks which requires more careful footstep planning.

Figure 3.6 shows a few moments of SiLVIA robot performance testing. We demonstrate that SiLVIA can walk over uneven terrains, walk up/downstairs (walking upstairs is more difficult) all with open loop gait. SiLVIA can also lift a 20kg weight which is 2.3 times its own mass. With suction cups attached to its feet, SiLVIA can attach itself to the white board very reliably and perform tasks. The main factors of such capabilities are that SiLVIA has 6 legs which gives extra stability and support of its body on uneven terrains. In addition, SiLVIA use dual Dynamixel 106 motors which has large gear ratio and enormous torque output given its weight. The strength and stability makes the climbing tasks described afterwards possible.

The testings on SCALER are mainly focused on fast walking, walking with payload and single wall climbing. Due to the linkage design, the robot is able to walk stably with a payload of more than 13kg as shown by Fig. 3.5. When walking at full speed, it is able to exceed $0.56 m/s$. Comparison tables of the walking speed and payload between different robots can be seen in [104]. This robot has decent performance even comparing to the famous quadruped robots such as ANYMAL and SPOT. In addition, this robot can climb onto a single



(a) 90 degrees climb with 3.4 kg payload.

(b) Upside down climbing.

Figure 3.4: SCALER climbing test.

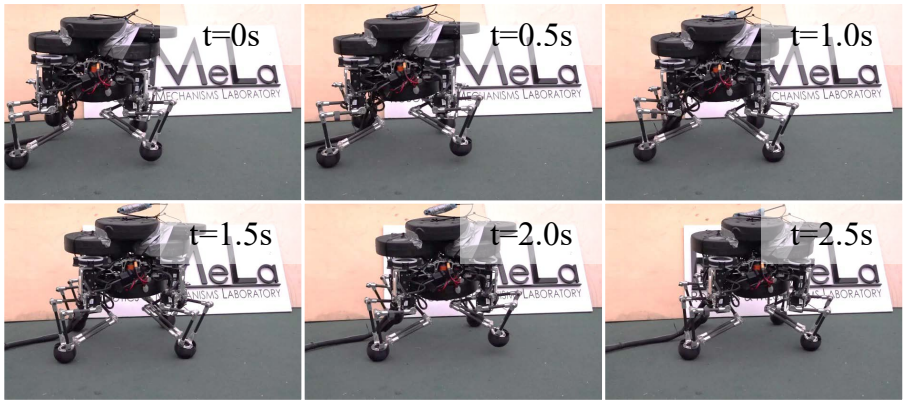


Figure 3.5: SCALER walking test.

rock-climbing wall perpendicular to the ground as shown by Fig. 3.2. It is also able to climb upside-down, or climb with a payload of more than 3kg shown by Fig. 3.4. The climbing speed is 0.42m/min. This speed is significantly faster than Lemur 3 which is a climbing robot of similar size and similar type of gripper [45].

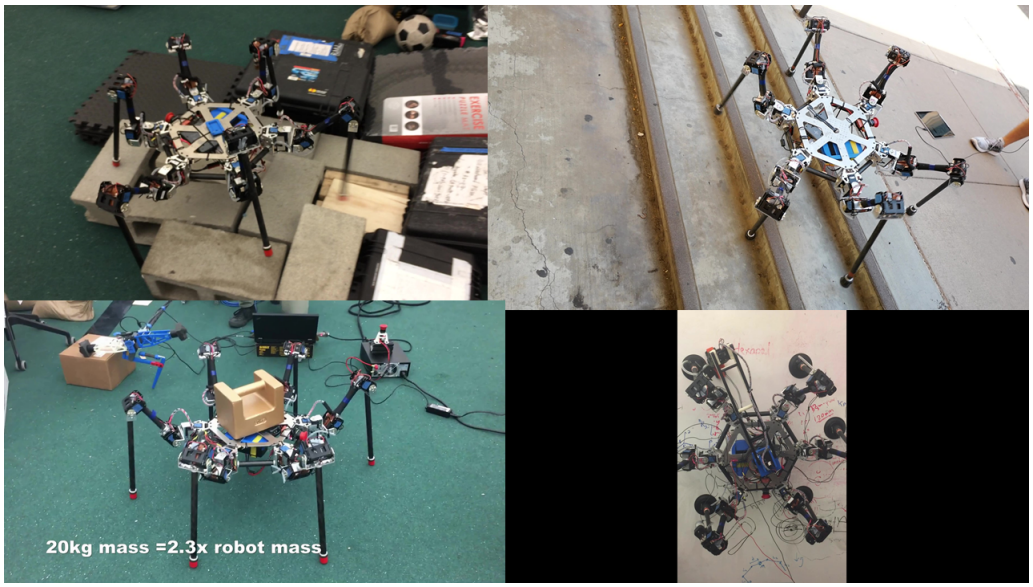


Figure 3.6: Validation of open loop performance. Left up: SiLVIA walk over uneven terrain using open loop gait. Right up: SiLVIA walk down stairs using open loop gait. Left down: SiLVIA lift a 20kg weight which is 2.3 times it own mass. Right down: SiLVIA equipped with suction cups is attaching itself on the white board.

3.4 Grippers

Both SiLVIA and SCALER are equipped with micro-spine grippers. The design of the grippers are similar to [105]. Previous works [105–110] has done comprehensive modeling and design works on spine-based grippers. However, there is one key difference between the way we use the spine gripper and the way previous works use the spine gripper. The main difference is that the approach [105–110] takes assumes the gripper has a preferred direction. The spines are angled to maximize the adhesion force. These spines are placed throughout the palm of the gripper [107], and the way the gripper approaches to the rock surface and grip is consistent with the spine preferred direction. Based on their model [106], minimizing the normal pushing force from the springs to the spines is preferred since extra normal force will decrease the spine adhesion force. On the other hand, all our grippers operates under large

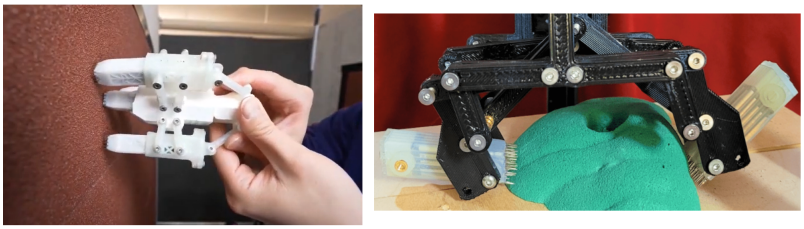


Figure 3.7: Grippers used for SiLVIA and SCALER. Left: SiLVIA gripper which has a 3 finger design. Right: SCALER gripper which has a 2 finger design.

extra normal force [100], shown by figure 3.7. The function of our gripper is leaning more towards increasing the coefficient of friction. According to the model [106], the amount of extra normal force exerted on the gripper will make the gripper completely non-functional. Therefore, modifications are required on the previously published models.

We identify three forces acting on the gripper. The gripping force f^g is defined as the force between the gripper and the surface of the environment *when no external force is acting on the gripper*. For magnet type of grippers, this force corresponds to the magnetic force. For our spine based gripper, even under minimal external load, the spines will insert into the microscopic gaps on the surface, generating a significant amount of shear force. We assume that f^g has pure shear components. f^e is the external load on the gripper generated by the robot limb. f^r is the sole reaction force between the surface of the environment and the gripper. Due to the spines, we assume that this reaction force is a frictional force and is subject to friction cone constraint. For many other grippers (e.g., suction cups, magnetic grippers), the ultimate gripping forces are also frictional forces. Thus, our force model is generalizable. The forces on the gripper should satisfy:

$$-f^r = f^g + f^e \tag{3.2}$$

A gripper may play two roles. First, it can increase the coefficient of friction

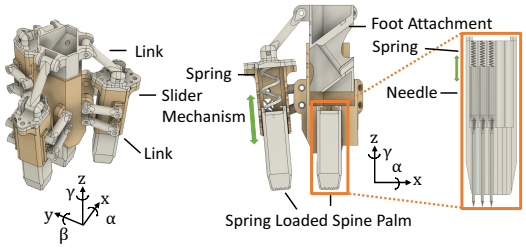


Figure 3.8: Mechanical design of the spine gripper

between the surface of the environment and the toe, relaxing the constraint on $\boldsymbol{f}^r$. It may also provide $\boldsymbol{f}^g$ to increase the reaction force. Equation 3.2 indicates that the gripping force $\boldsymbol{f}^g$ can increase the reaction force if $\boldsymbol{f}^e$ is fixed. This decomposition allows the planner to take into consideration gripping forces explicitly.

In this work, we assume that gripping forces by spine grippers is a function of the gripper orientation and the coefficient of friction. This is because with a microscopic view, how the spine touches the surface has a significant influence on the gripping force [105, 106]. Hence, the state $\boldsymbol{s}$ is a four-dimensional vector with $\boldsymbol{s} = [\alpha, \beta, \gamma, \lambda]^\top$ where α, β, γ are the rotation angles along x, y, z axis that are defined in 3.8, respectively. λ is the coefficient of friction of the surface. Although the gripper generates the shear and the normal force on the environment, the normal force by our grippers is relatively small. Thus, we assume that the grippers generate only shear adhesion.

Here, we assume that the shear force follows Gaussian distribution. Given a data set $S = \{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_n\}$ with the measured shear forces $\boldsymbol{y}^g = \left[y_1^g, \ldots, y_n^g\right]^\top$, the shear force $\boldsymbol{f}^g$ by a gripper can therefore be modeled as:

$$\boldsymbol{f}^g(\boldsymbol{s}) \sim \mathcal{GP}(\boldsymbol{\mu}^g(\boldsymbol{s}), \boldsymbol{\kappa}^g(\boldsymbol{s}, \boldsymbol{s}_*)) \tag{3.3}$$

where, $\boldsymbol{f}^g = \left[f_1^g, \ldots, f_n^g\right]^\top$, n is the number of samples from a GP.

$\boldsymbol{\mu}^g(\boldsymbol{s}) = \left[\mu_1^g(\boldsymbol{s}), \ldots, \mu_n^g(\boldsymbol{s})\right]^\top$ is the mean and $[\boldsymbol{\kappa}^g]_{i,j} = \kappa^g\left(\boldsymbol{s}_i, \boldsymbol{s}_j\right)$ is the covariance

matrix, where $\kappa^g(\cdot,\cdot)$ is a positive definite kernel. In this work, we employ the squared exponential kernel as follows:

$$\kappa^g\left(\boldsymbol{s}_i, \boldsymbol{s}_j\right) = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{\left|\boldsymbol{s}_i - \boldsymbol{s}_j\right|^2}{\ell^2}\right) \tag{3.4}$$

where σ_f^2 represents the amplitude parameter and l defines the smoothness of the function $\boldsymbol{f}^g$.

Here, let $\boldsymbol{D} = [\boldsymbol{s}_1, \cdots, \boldsymbol{s}_n]^\top$ be the matrix of the inputs. In order to predict the mean and variance matrix at $\boldsymbol{D}_*$, we obtain the predictive mean and variance of the shear force by assuming that it is jointly Gaussian as follows:

$$\hat{\boldsymbol{f}}^g = \mathbb{E}[\boldsymbol{f}^g(\boldsymbol{D}_*)] = \boldsymbol{\kappa}_*^\top \left(\boldsymbol{K}_D + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}_g \tag{3.5}$$

$$\hat{\Sigma}^g = \mathbb{V}[\boldsymbol{f}^g(\boldsymbol{D}_*)] = \boldsymbol{\kappa}_{**} - \boldsymbol{\kappa}_*^\top \left(\boldsymbol{K}_D + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\kappa}_* \tag{3.6}$$

where $\boldsymbol{\kappa}_* = \boldsymbol{\kappa}_g(D_*, D)$, $\boldsymbol{K}_D = \boldsymbol{\kappa}_g(D, D)$, $\boldsymbol{\kappa}_{**} = \boldsymbol{\kappa}_g(D_*, D_*)$, and σ_n^2 is the variance of the Gaussian observation noise with zero mean.

CHAPTER 4

Compliance Model of Multi-legged Robots

For climbing robot motion planning, a whole body model is required to reason the contact forces. This is especially true for the two-wall climbing case [111]. For two-wall climbing, the robot requires a contact force between its feet and the wall to keep itself from falling down, essentially squeeze itself between walls. Such a “squeezing" force comes from compliance as a result of P-controlled motors. One key feature of this contact force is that it is statically indeterminate with 3 or more contact points to the environment, meaning one cannot solve them without a proper compliance model for the robot [112]. This section derives the stiffness matrix of one of the robot's limbs in order to calculate the contact force. We then show how the whole body stiffness is obtained from individual limb stiffness.

4.1 VJM for a Limb Stiffness

We derive the limb's stiffness matrix in Cartesian space using VJM as done by [113]. The stiffness matrix describes the spring-like behavior of the robot's limb, assuming the shoulder is fixed with a force exerted on the end effector and each joint angle is regulated by position control. The mathematical definition of the 3D stiffness matrix is given by:

$$\underline{f} = \mathbf{K}\delta\underline{X} \tag{4.1}$$

where $\delta\underline{X}$ is the tiny shift in position at the end effector due to limb deformation, and $\underline{f}$ is the reaction forces on the end effector. Note that the definition

is based on the fact that there is a fixed-end constraint on the limb shoulder, which is consistent with the mechanical design (Fig. 3.1).

For position controlled motors, if the motor position is regulated with only a Proportional (P) gain, the joint behaves like a torsional spring, whose spring constant is proportional to motor's P gain. For the P gain used for climbing, the robot structures are considered rigid, and the majority of the limb compliance can be lumped into the joint compliance.

Given a robot limb that has N degrees of freedom with point contact only (no torque exerted on end effector), the standard Jacobian matrix $\mathbf{J}$ is a $3 \times N$ matrix. If the deformation of the robot limb is small enough, one can assume that the deformation is linear elastic. Thus the relationship between linear deflection on the end effector and the rotational deflection (similar to the rotation angle of a torsional spring) on each joint is:

$$\delta \underline{X} = \mathbf{J} \delta \underline{\theta} \tag{4.2}$$

where $\delta \underline{\theta}$ is the rotational deflection on each joint.

The relationship between reaction forces on the end effector and the resultant torque on each motor is:

$$\underline{\tau} = \mathbf{J}^T \underline{f} \tag{4.3}$$

If the P-controlled motors are modeled as torsional springs with spring coefficient k_is, the motor rotational deflection angles are:

$$\delta\theta_i = \frac{\tau_i}{k_i}, \;\; i = 1...N \tag{4.4}$$

or

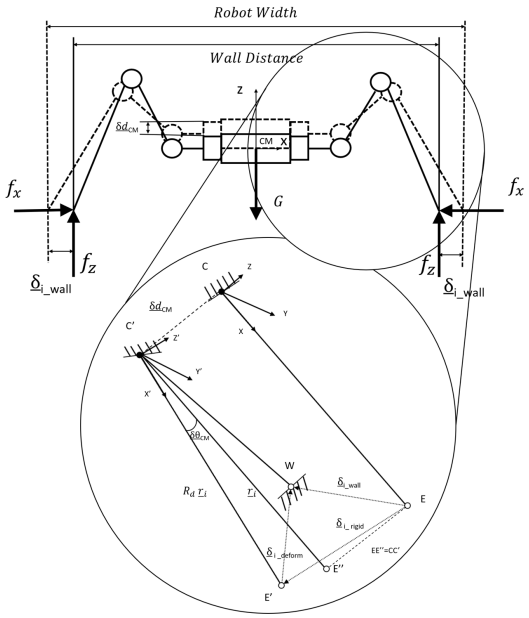


Figure 4.1: Diagram of limb deformation. The upper half shows the forces and deformations of the robot bracing between walls. The dashed line shows *state 1*, while the solid line shows *state 2*. The enlarged view shows the deformation of the robot limb, where $\overrightarrow{CE}$ represents *state 1*, $\overrightarrow{C'W}$ represents *state 2*, and $\overrightarrow{C'E'}$ represents *state 3*.

$$\delta\underline{\theta} = \mathbf{k}^{-1}\underline{\tau} \tag{4.5}$$

where:

$$\mathbf{k} = diag(k_i),\ \ i = 1...N \tag{4.6}$$

Therefore, from equation 4.2, plugging in equation 4.5 and 4.3, and then comparing it with equation 4.1, we have:

$$\mathbf{K} = (\mathbf{J}\mathbf{k}^{-1}\mathbf{J}^T)^{-1} \tag{4.7}$$

Note that the stiffness matrix from equation 4.7 will be symmetric and positive definite.

4.2 Whole Body Stiffness

In this subsection, the whole body stiffness matrix is assembled.

When a multi-limbed robot is climbing between 2 walls quasi-statically, each pose may be analyzed in two states. *State 1*: a human is holding the robot body while the its limbs stretch out to its commanded position with no wall contact. *State 2*: the wall is pushed in to its position and the human releases the robot's body. The difference between the commanded end effector position and the wall's position deforms the robot's limbs, and the body's center of mass has a deflection, *e.g.* the sag-down due to gravity. This causes the coordinate system attached to the body's center of mass to shift and rotate by a small amount, from XYZ to $X'Y'Z'$. This physical process is depicted in Fig. 4.1, where $\overrightarrow{CE}$ represents *state 1*, $\overrightarrow{C'W}$ represents *state 2*.

From standard elasticity theory, the movement from *state 1* to *state 2* contains two components: one due to rigid body movement (translation and rotation) and the other due to deformation. For this reason, the wall movement does not equal to the amount of deformation for limbs. To depict the correct deformation vector, we need to get rid of the rigid body movement portion. To do so, *state 3* is introduced, which starts from *state 2* but removes the wall while keeping the body fixed, depicted by $\overrightarrow{C'E'}$. *State 3* is a transition state which releases all the deformation, so that the movement from *state 1* to *state 3* only contains rigid body motion. Let the translational part of it be from $\overrightarrow{CE}$ to $\overrightarrow{C'E''}$, and the rotational part be from $\overrightarrow{C'E''}$ to $\overrightarrow{C'E'}$. Denote body's center of mass deflection by $\underline{\delta}_{CM} = [\underline{\delta d}_{CM}, \underline{\delta\theta}_{CM}]^T$, in which $\overrightarrow{CC'} = \overrightarrow{EE''} = \underline{\delta d}_{CM}$ is the small displacement and $\underline{\delta\theta}_{CM}$ is the small rotation. The wall-imposed deflection on limb i is denoted by $\underline{\delta}_{i\_wall}$, which is a known input. From *state 1*, if we first move the limb in parallel along $\underline{\delta d}_{CM}$ to get $\overrightarrow{C'E''}$, then rotate it to get $\overrightarrow{C'E'}$, we reach *state 3*. Then the wall deforms E' by $\underline{\delta}_{i\_deform}$ to reach W (*state*

2). Therefore $\underline{\delta}_{i\_deform}$ is the correct deformation vector for limb i, resulting in:

$$\underline{f}_i = \mathbf{K}_i \underline{\delta}_{i\_deform}, \;\; i = 1...N \tag{4.8}$$

Note this equation assumes the limb is subject to a fixed-end constraint at its shoulder, which is consistent with the way stiffness matrices are defined.

From Fig. 4.1:

$$\underline{\delta}_{i\_deform} = \underline{\delta}_{i\_wall} - \underline{\delta}_{i\_rigid}, \;\; i = 1...N \tag{4.9}$$

Wall motion $\underline{\delta}_{i\_wall}$ is a known input. To get $\underline{\delta}_{i\_deform}$, we seek an expression for $\underline{\delta}_{i\_rigid}$. Let $\mathbf{R}_d$ be the rotation matrix from frame XYZ to frame $X'Y'Z'$. Then the end effector displacement due to body rotation is $\overrightarrow{E''E'} = \mathbf{R}_d \underline{r}_i - \underline{r}_i$, where $\underline{r}_i = [x_i, y_i, z_i]^T = \overrightarrow{C'E''}$. Then we have:

$$\underline{\delta}_{i\_rigid} = \underline{\delta d}_{CM} + \mathbf{R}_d \underline{r}_i - \underline{r}_i, \;\; i = 1...N \tag{4.10}$$

When the rotation angles in $\underline{\delta\theta}_{CM}$ are infinitesimal, it can be shown [114] that $\mathbf{R}_d$ may be represented to the first order as:

$$\mathbf{R}_d = \begin{bmatrix} 1 & -\delta\theta_{CM_z} & \delta\theta_{CM_y} \\ \delta\theta_{CM_z} & 1 & -\delta\theta_{CM_x} \\ -\delta\theta_{CM_y} & \delta\theta_{CM_x} & 1 \end{bmatrix} \tag{4.11}$$

Where $\underline{\delta\theta}_{CM} = [\delta\theta_{CM_x}, \delta\theta_{CM_y}, \delta\theta_{CM_z}]^T$. Note this matrix is first order unitary.

Plugging equations 4.11 and 4.10 into equation 4.9:

$$\underline{\delta}_{i\_deform} = \underline{\delta}_{i\_wall} - [\mathbf{I} \;\; \mathbf{P}_i^T] \underline{\delta}_{CM}, \;\; i = 1...N \tag{4.12}$$

where:

$$\mathbf{P}_i = \begin{bmatrix} 0 & -z_i & y_i \\ z_i & 0 & -x_i \\ -y_i & x_i & 0 \end{bmatrix} \tag{4.13}$$

And by equation 4.8 the reaction force on the end effector is:

$$\underline{f}_i = \mathbf{K}_i(\underline{\delta}_{i\_wall} - [\mathbf{I} \;\; \mathbf{P}_i^T]\underline{\delta}_{CM}), \;\; i = 1...N \tag{4.14}$$

The static equilibrium equations are:

$$\sum_{i=1}^{N} \underline{f}_i + \underline{F}_{tot} = 0 \tag{4.15}$$

$$\sum_{i=1}^{N} (\underline{r}_i \times \underline{f}_i) + \underline{M}_{tot} = 0 \tag{4.16}$$

Where $\underline{F}_{tot}$ is the total load force and $\underline{M}_{tot}$ is the total load torque.

Plug equation 4.14 into equation 4.15 and 4.16 to get

$$\mathbf{A}\underline{\delta}_{CM} = \begin{bmatrix} F_{tot} \\ M_{tot} \end{bmatrix} + \sum_{i=1}^{N} \begin{bmatrix} \mathbf{K}_i \underline{\delta}_{i\_wall} \\ \mathbf{P}_i \mathbf{K}_i \underline{\delta}_{i\_wall} \end{bmatrix} \tag{4.17}$$

Where:

$$\mathbf{A} = \sum_{i=1}^{N} \begin{bmatrix} \mathbf{K}_i & \mathbf{K}_i \mathbf{P}_i^T \\ \mathbf{P}_i \mathbf{K}_i & \mathbf{P}_i \mathbf{K}_i \mathbf{P}_i^T \end{bmatrix} \tag{4.18}$$

is the *whole body stiffness matrix*. Equation 4.17 relates the body's center of mass deflection to its loading, plus a deformation input describing the effect of the imposed wall deflection.

As a result, the complete stiffness model can be summarized by equation (4.7), (4.13), (4.14), (4.17), and (4.18).

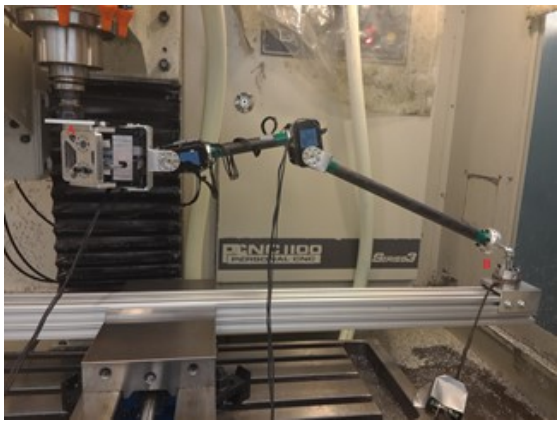


Figure 4.2: Test to retrieve stiffness matrices for SiLVIA leg on a CNC machine

Compared to previous works, especially [115] and [116], VJM, instead of MSA, is adopted to model the limb stiffness. Mobile robots tend to operate in uncertain environment and do not require high precision control. VJM method, although less accurate than MSA, models much faster, expediting its implementation. Additionally, the deformation input $\underline{\delta}_{i\_deform}$ is introduced as the difference between commanded end effector position and wall position. The robot's limb can actively change $\underline{\delta}_{i\_deform}$ by changing its commanded position to avoid failure.

4.3 Experiment for Stiffness Test

In order to test the theoretical results, experiments are established to measure the stiffness matrix of the robot limb. The experiment is done on a 3 axis CNC machine.

Figure 4.2 illustrates our setup. The robot limb is fixed at point A through the machine gripper, while hinged at point B by a ball joint. This ball joint ensures that no moment can be transferred from the fixture, thus simulate a pure friction contact. The ball joint is connected to the vise, which can move relative to the gripper at 3 axes. The z axis is along vertical direction, with positive up-

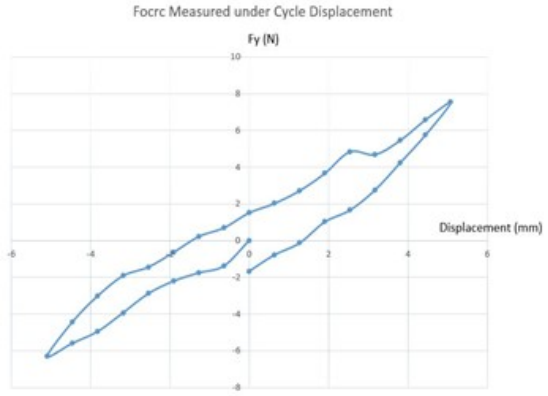


Figure 4.3: Curve of cycle loading demonstrating the gear backlash of SiLVIA motors

wards, while the x axis is along horizontal direction, with positive to the right. The amount of displacement is controlled by the machine. If we displace the robot limb along one axis, we get reaction force at end effector, which is measured by a ATI mini45 6 axes force/torque sensor. By doing single axis loading test, we can retrieve 2 components in the stiffness matrix. Therefore, through 2 axes loading test, we are able to build the 2 by 2 stiffness matrix. When the single axis test is conducted, point A and B are shift relative to each other up to 2 inch, with a 0.025-inch step, thus 8 data points are measured.

First, a cycle loading test is conducted to test the impact of backlash. Starting from zero load condition, the limb is loaded 2 inch along -Z direction first, and gradually unload and load 2 inch along +Z direction. The reaction force is measured at 0.025-inch step. One set of data is depicted in Figure 4.3, measured at $\theta_1 = 70°$ and $\theta_2 = 50°$. It is noticed that when unloaded, the 0 reaction force point is shifted. The experiment is repeated several times, and this phenomenon reappears with the same amount of loading, thus the author thinks it's due to the motor backlash (instead of plastic deformation). We can tell from Figure 4.3 that for the robot of our size, the zero reaction force point can shift 2.5mm due to backlash. This issue needs to be taken into consideration when the robot is climbing.

TABLE I. EXPERIMENT DATA

	r (mm)	K(N/m)	λ_{min}	λ_{max}
1	158	$\begin{bmatrix} 0.5 & -1.7 \\ -1.7 & 11.0 \end{bmatrix}$	0.2316	11.2684
2	176	$\begin{bmatrix} 1.0 & -2.0 \\ -2.0 & 8.5 \end{bmatrix}$	0.5	9
3	222	$\begin{bmatrix} 1.0 & -2.0 \\ -2.0 & 5.0 \end{bmatrix}$	0.1716	5.8284
4	236	$\begin{bmatrix} 1.3 & -1.8 \\ -1.8 & 3.0 \end{bmatrix}$	0.1594	4.1406
5	242	$\begin{bmatrix} 1.2 & -1.5 \\ -1.5 & 2.2 \end{bmatrix}$	0.1189	3.2811
6	276	$\begin{bmatrix} 2.5 & -2.7 \\ -2.7 & 3 \end{bmatrix}$	0.0385	5.4615
7	303	$\begin{bmatrix} 2.0 & -2.0 \\ -2.0 & 2.1 \end{bmatrix}$	0.0494	4.0506
8	320	$\begin{bmatrix} 1.7 & -2.0 \\ -2.0 & 3.0 \end{bmatrix}$	0.247	4.453
9	437	$\begin{bmatrix} 5.0 & -1.1 \\ -1.1 & 0.9 \end{bmatrix}$	0.6235	5.2765
10	458	$\begin{bmatrix} 5.5 & -1.4 \\ -1.4 & 0.5 \end{bmatrix}$	0.1347	5.8653
11	481	$\begin{bmatrix} 15.0 & -2.3 \\ -2.3 & 0.4 \end{bmatrix}$	0.0462	15.3538
12	484	$\begin{bmatrix} 18.0 & -5.0 \\ -5.0 & 1.6 \end{bmatrix}$	0.1958	19.4042

Figure 4.4: Result of stiffness testing: stiffness matrices and its eigenvalues

Since the motor has a backlash, the joints are preloaded to get rid of it when we measure stiffness matrices. The preload is subtracted out from the data points, and the slopes are averaged to compute stiffness coefficients, i.e.

$$k = \frac{1}{n} \sum_{i=1}^{n} \frac{F - F_{preload}}{\delta - \delta_{preload}} \tag{4.19}$$

We load the robot limb at 12 different configurations, the result is listed in table I. The result eigenvalue curves are plotted in Figure 13. All testes are repeated at least twice to make sure the data are repeatable. Stiffness matrix K is put as:

$$K = \begin{bmatrix} K_{xx} & K_{xy} \\ K_{yx} & K_{yy} \end{bmatrix} \tag{4.20}$$

We can tell from the experiment result that the stiffness coefficients do depend considerably on limb configurations. For example, $K_x x$ in configuration 11 is 30 times as much as it is in configuration 1. Although the idea behind this

model is motivated by structure compliance, it will be valuable to gauge what portion of compliance ellipsoid is actually contributed by structure compliance instead of joint compliance. By assuming that the all structure elements are perfectly rigid, and model the joint compliance into torsional springs with stiffness coefficients K_1 and K_2, one can derive analytical expression for compliance matrix induced only by joint compliance. From 2 sets of experiment data, K_1 and K_2 can be identified, and pure joint compliance matrices are computed to compare to experiment data. With this method, we roughly gauge the structure compliance to comprise to 30% of the total compliance.

CHAPTER 5

Motion Planning Algorithm for Walking and Climbing

Having set up the robot model, we formulate the optimization problem for vertical two-wall climbing in this section, and solve it with mixed-integer convex solvers.

5.1 A Two Step Decoupled Planner

5.1.1 Vertical Climbing Problem Formulation

5.1.1.1 Safety Factor for Climbing

One difference in wall climbing from walking is that wall climbing is a high-risk task. Falling down from a climb is likely to not only damage the robot and its environment but also injure people. When in a non-controlled environment, several uncertainties may cause climbing to fail unexpectedly. For instance, the friction coefficient can never be measured precisely, or there may be unexpected external load *e.g.* , wind. In our analysis wall-climbing tasks are typically static postures since the process happens slowly. However, there still exist velocities which may cause the end effector to disengage or over-torque. For this reason the authors propose the notion of safety factors for wall-climbing motions. In order to motion plan, it not only needs to satisfy the nominal constraint but also needs to satisfy the safety factor constraint.

Similar to finite element analysis, the safety factor is generated by analyzing each posture of the motion and calculating the ratio of the current index over the critical failure index. In this paper, we investigate a robot climbing between

two walls with frictional contact, and the two failure modes are insufficient friction (slip) and motor over-torque. Therefore, there are two safety factors to consider. Imagine being able to gradually reduce μ from the nominal value to the critical value μ_c, when the robot is about to slip. This provides us with a notion of the safety factor with respect to the coefficient of friction, S_μ, defined in equation (7). Similarly, if we imagine lowering the motor torque limit τ_{max} from its nominal value to a critical value τ_c, which is right before the motor over-torques. We can define another safety factor with respect to the motor's max torque, S_τ, defined by equation (8).

$$S_\mu = \mu/\mu_c \tag{5.1}$$

$$S_\tau = \tau_{max}/\tau_c \tag{5.2}$$

These notions are first introduced in our previous paper [111], where it can be retrieved graphically from feasibility region analysis. In this paper, we formulate a convex optimization problem to plan for the amount of pushing forces that satisfy the safety factor constraints for each planned robot pose.

5.1.1.2 Complete Formulation of the Planning Problem

In Fig. 5.10, we present here the complete mathematical formulation of motion planning problem for M-rounds climbing between walls with friction, where part of the decision variables, Γ_p, are

$$\Gamma_p = \{\underline{\mathbf{p}}_i[j],\ \underline{\mathbf{p}}_{COM}[j],\ \underline{\Theta}_b[j] \mid i = 1,\ldots,N,\ j = 1,\ldots,M\} \tag{5.3}$$

and the other part of decision variables, Γ_f, are

$$\Gamma_f = \{\, \underline{\delta}_{COM}[j],\ \underline{f}_i[j],\ \underline{\delta}_{i\_wall}[j],\ \mathbf{K}_i[j] \mid i = 1,\ldots,N,\ j = 1,\ldots,M \,\} \tag{5.4}$$

for each round j, it plans body's center of mass (COM) position $\underline{\mathbf{p}}_{COM}[j]$, body orientation $\Theta_b[j]$, body deflection $\underline{\delta}_{COM}[j]$ and the i^{th} limb's toe positions $\underline{\mathbf{p}}_i[j]$, the limb stiffness matrices $\mathbf{K}_i[j]$, the contact forces $\underline{f}_i[j]$, and limb deflections $\underline{\delta}_{i\_wall}[j]$, where i is the limb index.

Constraint A and B limit the range of travel between rounds. In *constraint* C, we approximate the limb workspace by a ball. $\underline{v}$ is the vector from robot body's COM to the first joint of the limb. *Constraint* D ensures the toe lies on a feasible contact region on the wall. In this paper, we assume perfect knowledge of the wall geometry, *i.e.* its mathematical expression is available to the planner. *Constraint* E represents inverse kinematics. *Constraint* F is equation (4.7), the limb stiffness matrix based on VJM. *Constraint* G is equation (4.17) (4.18) (4.13), the whole body stiffness model. *Constraint* H is equation (4.14) which relates limb contact force with its deflection. *Constraint* H, I, J, and K ensure the safety factor constraint in Section 5.1.1.1 is satisfied.

Given the results from [6], the complete problem may be solvable with a single NLP solver. Instead of doing that, we chose to separate the problem into two parts that solve an MICP/NLP problem first and then solve a series of standard convex optimization problems, as demonstrated in the next section.

5.1.2 Problem Solving with Optimization

Several papers e.g. [6, 19, 65] have demonstrated the power of NLP solvers being able to solve various nonlinear motion planning problems. However, NLP solvers can easily get trapped by local minima if the problem is complicated. We noticed that in the optimization problem *constraint* F can be numerically

hard for optimization solvers, especially due to potential singularity issues. Therefore, we chose to naturally decouple the problem into two sections at *constraint* F, as indicated on the left of Fig. 5.10. The first section is composed of *constraint* A, B, C, D. This is similar to a standard walking motion planning problem and can be solved by an MICP or NLP solver given the vast existing literature. The second section including *constraint* G, H, I, J, K. Given all toe positions and body posture, this part is a standard force distribution problem [117], and can be formulated into a standard convex optimization problem that can be easily solved. In this setup, the *constraint* F along with *constraint* E are evaluated algebraically after solving the first section of the problem and are not fed into any optimization solver. By doing so, we sacrifice some optimality, which can be seen in the following form:

$$\underset{\Gamma_p\ \Gamma_f}{\textbf{minimize}} \quad f(\Gamma_p, \Gamma_f)$$

subject to, for each round $j = 1, \ldots, M$ **and for each limb** $i = 1, \ldots, N$ **(**$N = 6$ **for hexapods)**

$$\text{Posture Planner} \begin{cases} [\Delta\underline{\mathbf{p}}_{min}, \Delta\underline{\Theta}_{min}] \le [\Delta\underline{\mathbf{p}}_{COM}, \Delta\underline{\Theta}_b] \le [\Delta\underline{\mathbf{p}}_{max}, \Delta\underline{\Theta}_{max}] & (\textit{constraint } A: \text{ body COM and orientation stepsize}) \\ \Delta\underline{\mathbf{P}}_{min} \le \Delta\underline{\mathbf{p}}_i \le \Delta\underline{\mathbf{P}}_{max} & (\textit{constraint } B: \text{ limb toe stepsize}) \\ \|\underline{\mathbf{p}}_i - \underline{\mathbf{p}}_{COM} - \mathbf{R}\underline{v}\|_2 \le \Delta_{FK} & (\textit{constraint } C: \text{ limb reachability}) \\ \underline{\mathbf{p}}_i \in \{\text{feasible contact regions}\} & (\textit{constraint } D: \text{ toe position assignment}) \end{cases}$$

$$\text{IK, VJM} \begin{cases} \underline{\theta}_i = f_{IK}(\underline{\mathbf{p}}_i) & (\textit{constraint } E: \text{ limb inverse kinematics}) \\ \mathbf{K}_i = (\mathbf{J}(\underline{\theta}_i)\mathbf{k}^{-1}\mathbf{J}(\underline{\theta}_i)^T)^{-1} & (\textit{constraint } F: \text{ limb stiffness from VJM}) \end{cases}$$

$$\text{Force Planner} \begin{cases} \mathbf{A}\underline{\delta}_{COM} = \begin{bmatrix} F_{tot} \\ M_{tot} \end{bmatrix} + \sum_{i=1}^{N} \begin{bmatrix} \mathbf{K}_i \\ \mathbf{P}_i\mathbf{K}_i \end{bmatrix} \underline{\delta}_{i\_wall} & (\textit{constraint } G: \text{ limb deflection and body sagdown}) \\ \underline{f}_i = \mathbf{K}_i(\underline{\delta}_{i\_wall} - [\mathbf{I}\ \mathbf{P}_i^T]\underline{\delta}_{COM}) & (\textit{constraint } H: \text{ limb deflection and contact forces}) \\ \underline{\tau}_i = \mathbf{J}(\underline{\theta}_i)^T \underline{f}_i & (\textit{constraint } I: \text{ Jacobian}) \\ S_\mu \ge 1 & (\textit{constraint } J: \text{ frictional safety factor}) \\ S_\tau \ge 1 & (\textit{constraint } K: \text{ motor torque safety factor}) \end{cases}$$

Figure 5.1: Complete optimization formulation for vertical climbing problem

$$\begin{aligned} \underset{\Gamma_p\ \Gamma_f}{\text{minimize}} \quad & f(\Gamma_p, \Gamma_f) \\ \text{subject to} \quad & h_1(\Gamma_p) \le 0 \\ & h_2(\Gamma_p, \Gamma_f) \le 0 \end{aligned}$$

The constraint $h_1 \leq 0$ denotes the part that is to be solved in the first part of the optimization problem while $h_2 \leq 0$ in the second part of the problem. Compared to an NLP solver that takes care of both sets of constraints simultaneously, in our 2-step setup constraint $h_1 \leq 0$ is solved independent of $h_2 \leq 0$. This means an optimal solution to $h_1 \leq 0$ may render $h_2 \leq 0$ non-optimal. However, we choose to solve this problem in such way since it has several advantages:

1. **Interpretability** The two problems have clear and distinct physical interpretations. The first part focuses on solving a series of postures, while the second part optimizes for how much force the robot needs to exert on the wall. If at one round the solver fails, it is clear why the planner fails, and part of the feasible solutions may still be used. Whereas for an NLP solver, if it returns *infeasible*, it tends to return results that can't be utilized and with no interpretable information.

2. **Adaptability** The first part of the problem is identical to the motion planning problem for legged walking. Thus, it easily connects to the vast literature of walking robot motion planning. Only the second part depends on end effectors for climbing, which can be easily reformulated if the end-effector is swapped.

3. **Speed** Decoupling the problem into two parts turns a bulk part of the problem into convex optimization problems, which can be solved efficiently. This can be justified by Table 5.1.

In the next two sections, we introduce detailed formulations for each part of the problem.

5.1.3 Optimization for Climbing Posture

Given a goal configuration during wall climbing, a pre-defined number of postures M to reach it should be computed under the constraints of step size, kinematics, and toe contact points within feasible contact regions. To simplify the task of assigning toe contact points, we divided feasible contact regions into several pre-computed convex constraints represented by $\mathbf{A}_r\mathbf{p}_i \leq \mathbf{b}_r$ with perfect knowledge of structured wall geometry, where r is the index of feasible contact region. The IRIS algorithm [5] used for typical walking robot motion planning problem, is also able to compute these regions with perception. The entire optimization problem for climbing posture is formulated as follows:

$$
\begin{aligned}
\underset{\Gamma_p,\, \boldsymbol{H}}{\text{minimize}} \quad & (\underline{\mathbf{q}}[M] - \underline{\mathbf{q}}_g)^T \underline{\mathbf{W}}_g (\underline{\mathbf{q}}[M] - \underline{\mathbf{q}}_g) + \\
& \sum_{j=1}^{M-1} (J_{COM} + J_{ROT} + J_S) \\
\text{subject to} \quad & \text{for } j = 1, \ldots, M, \\
& \Delta\underline{\mathbf{p}}_{min} \leq \|\underline{\mathbf{p}}_{COM}[j] - \underline{\mathbf{p}}_{COM}[j-1]\|_2 \leq \Delta\underline{\mathbf{p}}_{max} \\
& \Delta\underline{\mathbf{P}}_{min} \leq \|\underline{\mathbf{p}}_i[j] - \underline{\mathbf{p}}_i[j-1]\|_2 \leq \Delta\underline{\mathbf{P}}_{max} \\
& \Delta\underline{\Theta}_{min} \leq \underline{\Theta}_b[j] - \underline{\Theta}_b[j-1] \leq \Delta\underline{\Theta}_{max} \\
& \|\underline{\mathbf{p}}_i[j] - \underline{\mathbf{p}}_{COM}[j] - \mathbf{R}\underline{v}\|_2 \leq \Delta_{FK} \\
& H_{r,i}[j] \;\Rightarrow\; \mathbf{A}_r\mathbf{p}_i \leq \mathbf{b}_r \\
& \sum_{r=1}^{R} H_{r,i}[j] = 1 \quad H_{r,i} \in 0, 1
\end{aligned}
$$

where $\Delta\underline{\mathbf{p}}_{min}$, $\Delta\underline{\mathbf{p}}_{max}$, $\Delta\underline{\mathbf{P}}_{min}$, $\Delta\underline{\mathbf{P}}_{max}$, $\Delta\underline{\Theta}_{min}$, $\Delta\underline{\Theta}_{max} \in \mathbb{R}^3$ are bounds for the toe, body's COM and orientation step sizes. $\Delta_{FK} \in \mathbb{R}$ is the radius of the limb workspace ball. For each round, $\boldsymbol{H} \in \{0,1\}^{R\times 6}$ is taking on integer values to assign toes to feasible contact regions where R is the number of feasible contact

regions. The conditional constraint about $H_{r,i}$ is represented using a standard big-M formulation.

In terms of cost function, the first term is introducing the distance of the last round from goal configuration where $\underline{\mathbf{q}}[M] = [\underline{\mathbf{p}}_1[M], \ldots, \underline{\mathbf{p}}_6[M]]$ while $\underline{\mathbf{q}}_g$ is the goal configuration. The shifting amounts J_{COM}, J_{ROT} and J_S of body's COM, orientation and toe positions are added to avoid turning or climbing too far in one single step, as the second term of cost function:

$$\begin{cases} J_{COM} = \Delta\underline{\mathbf{p}}_{COM}^T \underline{\mathbf{W}}_{COM} \Delta\underline{\mathbf{p}}_{COM} \\ J_S = \sum_{i=1}^{6} \Delta\underline{\mathbf{p}}_i^T \underline{\mathbf{W}}_s \Delta\underline{\mathbf{p}}_i \\ J_{ROT} = \Delta\underline{\Theta}_b^T \underline{\mathbf{W}}_{ROT} \Delta\underline{\Theta}_b \end{cases}$$

where $\underline{\mathbf{W}}_{COM}$, $\underline{\mathbf{W}}_s$, $\underline{\mathbf{W}}_{ROT}$ and previous $\underline{\mathbf{W}}_g$ are weights used to tune the optimizer. Except rotation matrix $\mathbf{R}$ computed from $\underline{\Theta}_b[j]$, which has a nonlinear constraint, other parts forms an MICP, since they are either linear or quadratic (convex). To address the nonlinearity, linear approximation of $\mathbf{R}$ for $\underline{\Theta}_b[j] = [\alpha, \beta, \gamma]$ is used as follows:

$$\mathbf{R}(\underline{\Theta}_b[j]) = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} \tag{5.5}$$

This approximation is valid for applications involving small rotations which is reasonable for wall-climbing applications that do not require large rotations. The error is under 3% while angles are smaller than 10° using the Frobenius norm of a matrix to compare the similarity between linearly-approximated and exact rotation matrix.

When large changes in the body orientation are expected, linearization of the rotation matrix becomes invalid. In this case, NLP can be used. We formu-

lated our NLP according to [6] without considering the dynamic model. Moreover, NLP can easily handle non-convex terrains, *e.g.* , round tubes, which extends the application of our work.

5.1.4 Optimization for Pushing Force

After the posture planner is finished, the inverse kinematics, *constraint* E, and the stiffness matrix, *constraint* F, may be evaluated directly. Since those constraints are complicated and may have numerical stability issues due to the inverse of the matrix, we avoid directly placing it in a gradient based solver. The second part of algorithm tackles the problem of how much force each limb needs to exert on the wall. We use a pre-defined gait to go from one planned posture to the next one. The robot lifts one leg and puts it on the wall, pushes the body upwards, then lifts another leg, and repeats. We pick 12 critical instants between two postures for the force planner to investigate: 6 instants after the robot lifts one leg and 6 instants after the robot pushes its body up. The planner is formulated into a series of standard convex optimization problem so that it can be solved efficiently.

In Section 5.1.1.1 the notion of climbing motion safety factors, S_μ and S_τ, are proposed. Since the two safety factors are inversely related: pushing harder against the wall will increase S_μ but decrease S_τ and vice versa. We would like to guarantee that the safety factors are above a value larger than 1 while having a weight to tune the pushing force. This can be formulated as:

$$
\begin{aligned}
\underset{\tau_i \ \underline{f}_i}{\text{maximize}} \quad & S_\tau + wS_\mu \\
\text{subject to} \quad & |\tau_i| \leq \tau_{max}/S_\tau \\
& \underline{n}_i^T \underline{f}_i \geq 0 \\
& \|\underline{f}_i - (\underline{n}_i^T \underline{f}_i)\underline{n}_i\|_2 \leq (\mu/S_\mu)(\underline{n}_i^T \underline{f}_i) \\
& S_\mu \geq 1 \ , \ S_\tau \geq 1
\end{aligned}
$$

where $\underline{n}_i$ is the wall normal vector at toe i, and w is the weight to trade-off between the two safety factors. If we define $S_{\tau\_inv} = 1/S_\tau$, we can write the torque constraints into a linear form. Although the friction cone constraint itself is convex, adding in frictional safety factor S_μ as an optimization variable makes it non-convex. Thus we set a constant S_μ, which shrinks the friction cone. We put normal reaction forces $\underline{n}_i^T \underline{f}_i$ into the objective function. By tuning the amount of normal reaction force, the effective friction cone constraint can be made looser or tighter. Stated formally:

$$
\begin{aligned}
\underset{\underline{\delta}_{COM} \ \underline{f}_i \ \underline{\delta}_{i\_wall} \ \tau_i \ S_{\tau\_inv}}{\text{minimize}} \quad & S_{\tau\_inv} - w \sum_{i=1}^{N} \underline{n}_i^T \underline{f}_i \\
\text{subject to} \quad & \mathbf{A}\underline{\delta}_{COM} = \begin{bmatrix} F_{tot} \\ M_{tot} \end{bmatrix} + \sum_{i=1}^{N} \begin{bmatrix} \mathbf{K}_i \\ \mathbf{P}_i \mathbf{K}_i \end{bmatrix} \underline{\delta}_{i\_wall} \\
& \underline{f}_i = \mathbf{K}_i(\underline{\delta}_{i\_wall} - [\mathbf{I} \ \ \mathbf{P}_i^T]\underline{\delta}_{COM}) \\
& \underline{\tau}_i = \mathbf{J}(\underline{\theta}_i)^T \underline{f}_i \\
& 0 \leq S_{\tau\_inv} \leq 1 \\
& |\tau_i| \leq S_{\tau\_inv}\tau_{max} \\
& \underline{n}_i^T \underline{f}_i \geq 0 \\
& \|\underline{f}_i - (\underline{n}_i^T \underline{f}_i)\underline{n}_i\|_2 \leq \mu(\underline{n}_i^T \underline{f}_i)/S_\mu
\end{aligned}
$$

Increasing w makes the normal reaction force higher. This loosens the friction cone bound, since the required shear force (f_z in Fig. 4.1) is static determinate (half of the gravity G). Decreasing w increases S_τ, while tightening the friction cone bound. This problem is convex and can be solved quickly with a global optimal guarantee. According to our trial and error hardware testing, a lower bound of $S_\mu = 1.8$ provides sufficient safety against the coefficient of friction.

5.1.5 Results

We present here three scenarios that we investigated using our planner: climbing over steps on the walls, climbing on the walls while avoiding obstacles, and climbing on non-parallel walls. All results are validated on actual hardware with properly tuned weights. In each experiment, the walls are covered by rubber pads and the robot toes are covered by anti-slip tape, which gives a frictional coefficient μ around 1. A body posture regulator based on IMU orientation feedback and PID control is utilized for the robot body to track the planned orientation. No other feedback is used. Due to the stable but slow one-leg gait, the climbing speeds in all cases are around 20 cm/min.

5.1.5.1 Climbing over steps on the walls

In this scenario, we let the robot climb between two walls at a distance of 1230mm but with a 40mm thick by 200mm high step on both walls. The wall has multiple feasible contact surfaces, but the robot doesn't need to rotate for this task. Therefore, MICP is used for planning the robot posture with the body orientation kept flat. However, the robot does need to adapt to the tightening of the walls' distance between the walls. On the steps, the robot doesn't push as far out to prevent over-torque. Fig. 5.2 shows the planned series of postures,

visualized in MATLAB, as well as associated hardware testing scenarios.

To verify the contact force optimization results, the setup of the robot climbing onto the steps is simulated in V-rep. Fig. 5.3 plots the planned with the simulated torque curves and the critical friction coefficient μ_c as defined by equation (5.2) with their failure boundaries for 3 consecutive legs. The torque plotted is the maximal torque among three motors of one leg. When a certain leg is lifted and in the air, the planned torque is set to zero because lifting is achieved by the controller instead of planner. Simulated torques for the lifted leg is negligibly small compared to the torques when it is on the wall so that we can tell lifting phase from these curves. And the frictional factor is also zero with no friction generated for the lifted leg. In Fig. 5.3 (a), the maximal torques of right middle (RM) and right back (RB) legs decrease after right front (RF) leg finished lifting phase (between the shaded interval). With one leg in the air, other legs need to achieve larger contact force to avoid slipping. Once the lifted leg reaches its goal position, contact force would be re-distributed to the 6 legs on the wall. As we can tell, due to the complexity of contact, μ_c tends to exceed the planned values, which is the reason we weighted more on friction than torque. The non-smoothness of the data is in part due to the toe rubbing on the wall (caused by the physics simulator) and to the overshoot of PID body posture controller. Some points of the curves are above the boundaries, but the robot will not slip or over-torque if this is not continuous.

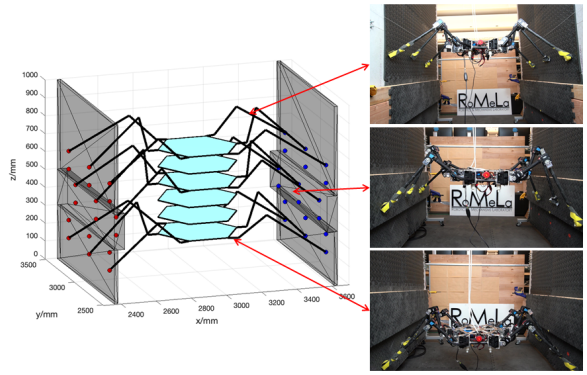


Figure 5.2: Visualization and hardware testing of planning results for climbing over steps on the walls.

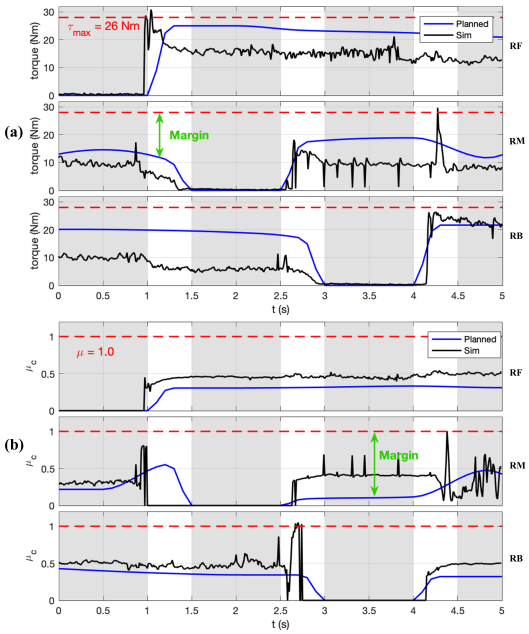


Figure 5.3: Diagrams of the planned and V-rep (Bullet 2.83 engine) simulated results for the required motor torque τ_c (maximum of 3 motors, diagram (a)) and coefficient of friction μ_c (diagram (b)) for a single leg. The plotted data is for right front (RF) leg, right middle (RM) leg, and right back (RB) leg. The shaded regions are when the robot lifts a certain leg and put it on the next position, and white regions are when the robot pushes its body up. The plot shows failure points (red dashed line), planned curves (blue line), and simulation results (black line). The arrow (green) indicates the margin due to planned safety factor. The results demonstrate a general correspondence of planned and V-rep simulated results.

5.1.5.2 Climbing on the walls while avoiding obstacles

This scenario focuses on planning the climbing direction and orienting the body to avoid an obstacle between the walls, obstructing a direct path. The planned results are visualized in Fig. 5.4. The robot doesn't need to rotate its body more than 20 degrees to complete the task; thus, MICP with a linearized body rotation matrix is applied. Due to the obstacle, the feasible contact region shrinks. We manually divide each wall into three convex regions: the upper, middle, and lower rectangle. For each round, the toe position is optimized within one of the three regions selected by the MICP planner. Currently, the robot does not have any vision sensor. In the future, this division can be provided by a perception system, and the complete process will be automated. The hardware test is shown in Fig. 5.5.

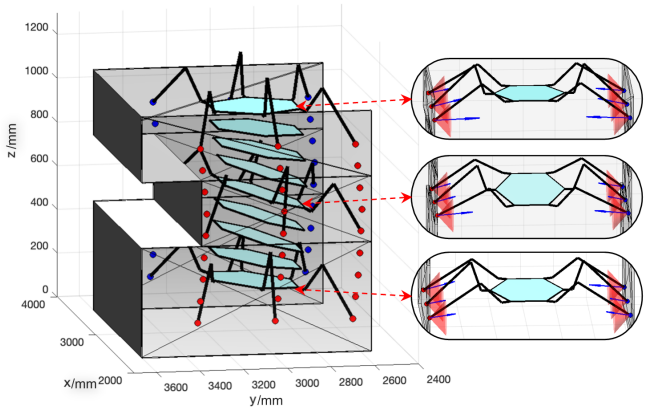


Figure 5.4: A series of trajectories generated by the MICP motion planner for the six-legged robot to climb up between two walls while avoiding an obstacle. Red and blue dots show the planned toe positions, and the hexagons show the body orientation. Three postures are detailed on the right with planned contact forces (blue arrows) along with the nominal friction cones (red).

Figure 5.5: Hardware test for climbing and avoiding an obstacle. The robot starts underneath the obstacle, as shown in the top figure, and then it angles its body and climbs forward and up simultaneously, as shown in the middle and bottom figures.

The MICP plans 8 rounds for this problem. Within each round, the robot lifts each leg once and pushes up the body for 6 times. Hence, the serial convex optimizer plans the force 12 times for each round and 96 times in total. Some statistics for this planner is shown in Table 5.1 (taken on an Intel Core i7-8750H machine). Since the bulk of the problem is convex, the total solution speed is decently fast. We point out here that a single NLP solver will need to deal with the same amount of variables and constraints; thus, a similar or slower speed is expected (refer to Table I in [6]).

5.1.5.3 Climbing on non-parallel walls

An interesting variation of the two-wall climbing problem is when the walls are no longer parallel. Ideally, we want the robot to take use of two walls at an arbitrary angle and climb up. This section demonstrates that our planner

Table 5.1: Specs for climbing and avoiding obstacle

	Solver	Variables	Constraints	T-Solve *	Total T-Solve
Posture Planner (MICP - 8 rounds)	Gurobi	480 (192 continuous 144 binary)	1002	420 ms	1380 ms
Force Planner	Gurobi	5856 (61 x 96)	10368 (108 x 96)	960 ms (10 ms x 96)	

* Include problem set-up time

can be used to plan the climbing motion when two flat walls are at a horizontal angle α. We pick $\alpha = 20$ degrees and implement the planning results on the hardware, shown in Fig. 5.6. Additionally, we are interested in retrieving a feasible climbing region regarding the given horizontal angle α and the wall coefficient of friction μ. We fix the distance between the two middle legs, and solve this problem by running the planner at discrete grid points for α and μ, and label each point feasible/infeasible. Fig. 5.7 shows the result when the robot is at its own weight without payload (10.3kg). In the plot, the shaded region shows where the robot succeeds, and the rest can be divided into where the robot fails to provide enough force, or fails kinematically (*i.e.* , some toes cannot reach the walls). This result demonstrates that our planner can be extended to broader two-wall cases and possibly to climbing up poles or trees where $\alpha = 180$ degrees.

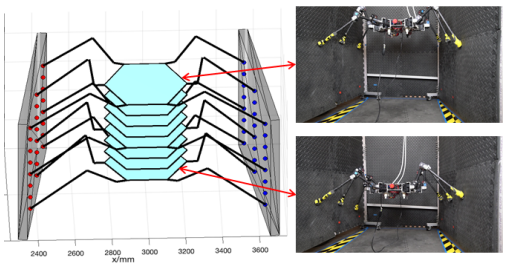


Figure 5.6: Visualization and hardware test of planning results for climbing on non-parallel walls with $\alpha = 20$ degrees.

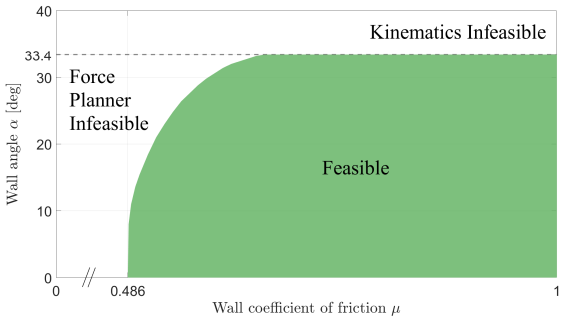


Figure 5.7: Feasible region for climbing on non-parallel walls.

5.1.6 Discussion

By decoupling the problem into two parts, we sacrifice some optimality. However, this makes the solver easier to tune. The feasibility region (as shown in [111]) for this problem is pretty narrow. Therefore, it is critical to find the proper weights in the force planner. Since our force planner is convex and interpretable, the difficulty of tuning the weights is attenuated.

To enable the posture planner to plan according to contact force, we can fuse constraints G through K into the posture planner. This will hopefully enhance the optimality of the solution returned. Additionally, we plan to develop a perception system that maps the wall and retrieves its geometry information. This combined with the planner may automate the complete climbing process and ensure the robot can track the motion plan. Given the solving speed of our algorithm, it could be implemented online to constantly re-plan. We also investigated the problem of climbing spirally up inside a tube with an MICP/NLP solver as the posture planner, which requires the robot to rotate its body by large angles. This will be published in future papers. Other future works include extending the safety factor design principle to other types of grippers *e.g.* , gecko type or microspine, comparing and combining MICP and NLP, *etc*. Dynamics could be added into the planner, if dynamic climbing is desired.

However, we show that if the planner resolves more than rigid body dynamics, *e.g.* , body compliance, a good option is to decouple the algorithm according to the physics, and resolve them hierarchically.

5.2 A Coupled Planner with Data-driven Envelope Relaxation

In the previous section, we presented a decoupled 2-stage planner with kinematics planning as the first stage and force planning as the second stage. While this planner is fast, one issue is that the first stage kinematics planner can make the second stage force planner infeasible since it has no knowledge of the existence of the second stage. One example is that if the kinematics planner put the contact force inside an infeasible area (e.g. coefficient of friction equals to zero), then the force planner would not be able to find a solution. In this section, we add coupling between the 2-stage planners. Specifically, we approximate bilinear torque constraints as McCormick envelopes and add them to the first stage planner.

5.2.1 Problem Setup

Similar to the last section, a multi-limbed robot is assumed to make N point contacts (i.e. pure contact force, no contact moment) with the environment. We denote the contact points with index i where $i = 1,...,N$. We model the environment with polygon meshes (e.g. a triangular mesh is depicted in Fig. 5.8).

Additionally, we assume that mesh i's vertices (corresponding to the limb i in contact with the mesh) are denoted by $\mathbf{v}_{iu}$, where u is the indices of the vertices, $u = 1,...,U$ and U is the number of vertices for mesh i. The normal direction $\mathbf{n}_i$ can be retrieved via a perception system. If we define $\mathbf{p}_i^w$ to be the position of toe i within mesh i with respect to the world frame:

$$\mathbf{p}_i^w = \sum_u p_{iu}^w \mathbf{v}_{iu},\ \sum_j p_{iu}^w = 1,\ p_{iu}^w \in [0,1] \tag{5.6}$$

If we define $\mathbf{p}_i^b$ to be the position of toe i with respect to the origin of the body coordinate system, we have:

$$\mathbf{p}_i^w = \mathbf{p}_{COM} + \mathbf{p}_i^b \tag{5.7}$$

where $\mathbf{p}_{COM}$ is the position of the center of mass (COM) with respect to the origin of the global coordinate system.

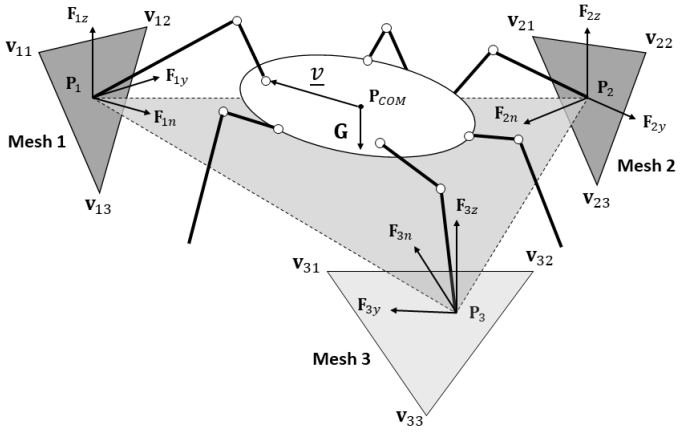


Figure 5.8: A robot making 3 contacts with the environment, subject to gravity. Contact planes are represented by triangular meshes with vertices $\mathbf{v}_{sj}(i,j = 1,2,3)$. $\mathbf{F}_{sn}$, $\mathbf{F}_{sy}$, $\mathbf{F}_{sz}$ are constant vectors along the normal and two pre-defined shear directions. The three contact points form a contact triangle.

Let the contact force on limb i be denoted by $\mathbf{f}_i$. Since $\mathbf{n}_i$ is known, we can define two mutually perpendicular shear directions $\mathbf{y}_i$ and $\mathbf{z}_i$, which can be pre-defined for every mesh. The three directional vectors form a local mesh coordinate frame, and the contact force can be represented as $\mathbf{f}_i = \mathbf{f}_{in} + \mathbf{f}_{iy} + \mathbf{f}_{iz}$ in the mesh frame, where $\mathbf{f}_{in}$ is the normal force, and $\mathbf{f}_{iy}$ and $\mathbf{f}_{iz}$ are the shear force components. Define $\mathbf{F}_{in}$, $\mathbf{F}_{iy}$, $\mathbf{F}_{iz}$ as constant vectors along the normal and two

shear directions on mesh i, we can express the dimensional force components using the non-dimensional force components f_{in}, f_{iy}, f_{iz} as $\mathbf{f}_{ik} = f_{ik}\mathbf{F}_{ik}$, $k = n, y, z$. The total contact force is:

$$\mathbf{f}_i = \sum_{k=n,y,z} f_{ik}\mathbf{F}_{ik} \tag{5.8}$$

Similarly, defining $\mathbf{P}_x$, $\mathbf{P}_y$, $\mathbf{P}_z$ as constant vectors along the 3 axes of the world frame, $\mathbf{p}_i^b$ can be expressed as:

$$\mathbf{p}_i^b = \sum_{j=x,y,z} p_{ij}^b \mathbf{P}_j \tag{5.9}$$

Both position and force characteristic quantities are chosen such that $p_{ij}^b \in [-1, 1]$, $f_{ik} \in [-1, 1]$.

Also note that equation (5.8) and (5.9) non-dimensionalized the force and position variables $\mathbf{f}_i$ and $\mathbf{p}_i^b$ into quantities f_{ik} and p_{ij}^b. This is suggested before utilizing certain mathematical operations to avoid unit mismatch as seen in [118].

Having set up the contact positions and forces, whole body constraints can be imposed. The robot motion is assumed to be quasi-static, thus the robot is always subject to the static equilibrium constraint:

$$\sum_{i=1}^{N} \mathbf{f}_i + \mathbf{F} = 0 \tag{5.10}$$

$$\sum_{i=1}^{N} \mathbf{p}_i^b \times \mathbf{f}_i + \mathbf{M} = 0 \tag{5.11}$$

where $\mathbf{F}$ and $\mathbf{M}$ are known external forces (gravity $\mathbf{G}$ in this work) and moments. COM is also assumed to always be at the geometric center, irrespective

of the robot limb motion.

Equation (5.11) introduces a bilinear term $\mathbf{p}_i^b \times \mathbf{f}_i$. Utilizing (5.8) and (5.9), the bilinear term can further be expressed as:

$$\mathbf{p}_i^b \times \mathbf{f}_i = \sum_{j=x,y,z} \sum_{k=n,y,z} p_{ij}^b f_{ik} \mathbf{P}_j \times \mathbf{F}_{ik} \tag{5.12}$$

To isolate the bilinear terms, let the moment variables be:

$$m_{ijk} = p_{ij}^b f_{ik} \tag{5.13}$$

where m_{ijk}'s are the moment components. Plugging (5.13) into (5.12), and further back into (5.11) results in:

$$\sum_{i=1}^{N} \sum_{j=x,y,z} \sum_{k=n,y,z} m_{ijk} \mathbf{P}_j \times \mathbf{F}_{ik} + \mathbf{M} = 0 \tag{5.14}$$

In the predominant case where legged robots are not equipped with grippers on its toes, the point contacts with the environment are pure frictional contacts. When the robot places its toes, the contact forces are subject to friction cone constraints. Given (5.8) and defining μ as the friction coefficient, the constraint can be written as:

$$\sqrt{f_{iy}^2 + f_{iz}^2} < \mu f_{in} \tag{5.15}$$

To plan a complete quasi-static motion to the specified goal, the motion planner generates a series of “key frames"—body COM positions, body orientations, and footstep positions—to the goal position. The number of key frames is pre-specified as M, and each key frame posture needs to satisfy kinematics

constraints. Between two consecutive key frames, step size constraints are enforced. Similar to [7], the following constraints are used:

$$
\begin{aligned}
&\text{for } r = 1, \ldots, M, \\
&\Delta\underline{\mathbf{p}}_{min} \leq \|\underline{\mathbf{p}}_{COM}[r] - \underline{\mathbf{p}}_{COM}[r-1]\|_2 \leq \Delta\underline{\mathbf{p}}_{max} \\
&\Delta\underline{\mathbf{P}}_{min} \leq \|\underline{\mathbf{p}}_i[r] - \underline{\mathbf{p}}_i[r-1]\|_2 \leq \Delta\underline{\mathbf{P}}_{max} \\
&\Delta\underline{\Theta}_{min} \leq \|\underline{\Theta}_b[r] - \underline{\Theta}_b[r-1]\|_2 \leq \Delta\underline{\Theta}_{max} \\
&\|\underline{\mathbf{p}}_i[r] - \underline{\mathbf{p}}_{COM}[r] - \mathbf{R}[r]\underline{v}\|_2 \leq \Delta_{FK}
\end{aligned}
\tag{5.16}
$$

where $\Theta_b[r]$ is body orientation, $\Delta\underline{\mathbf{p}}_{min}$, $\Delta\underline{\mathbf{p}}_{max}$, $\Delta\underline{\mathbf{P}}_{min}$, $\Delta\underline{\mathbf{P}}_{max}$, $\Delta\underline{\Theta}_{min}$, and $\Delta\underline{\Theta}_{max}$ are bounds for the toe, body COM, and orientation step sizes. The limb workspace is simplified into a sphere, with $\Delta_{FK} \in \mathbb{R}$ its radius. $\underline{v}$ is the constant shoulder vector from body COM to the first joint of the limb (depicted in Fig. 5.8). $\mathbf{R}[r]$ is the rotation matrix as a function of $\underline{\Theta}_b[r]$. We assume the body rotation angles are small (< 15 degrees), thus the rotation matrix can be linearized in terms of $\underline{\Theta}_b[r] = [\alpha, \beta, \gamma]$ [7].

In summary, there are kinematics constraints (5.16) (convex), (5.6) (5.7) (5.9) (linear), static equilibrium constraints (5.8) (5.10) (5.14) (linear), with additional constraints (5.13) (bilinear), and friction cone constraints (5.15) (convex). The objective function minimizes the distance of the planned final configuration to the goal configuration and penalizes the step size in similar fashion as [5][7]. Let us denote the complete problem by P. This problem is parametrized by θ which is the terrain geometry $\mathbf{v}_{iu}$. Fig. 5.8 illustrates the scene and the notations that are used in this paper. For simplicity, the decision variables are grouped into two sets -

kinematics variables $\Gamma_p = \{\ \underline{\mathbf{p}}_i[r],\ p_{ij}[r],\ \underline{\mathbf{p}}_{COM}[r],\ \underline{\Theta}_b[r]\ \}$ and force variables $\Gamma_f = \{\ \mathbf{f}_i[r],\ f_{ik}[r],\ m_{ijk}[r]\ \}$.

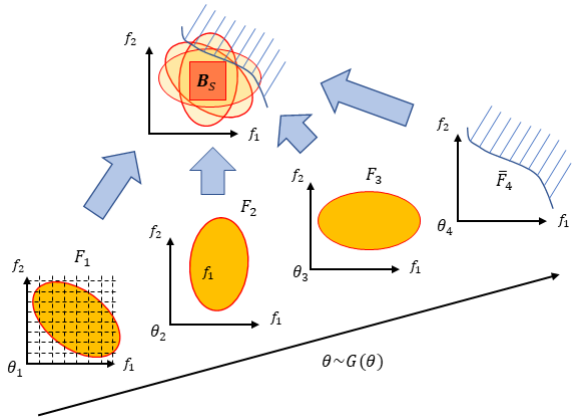


Figure 5.9: The proposed approach to find $\mathbf{B}_{\mathcal{F}}$ inside the intersection of solution set F_i's and exclude infeasible solution set $\overline{F}_i$'s. While we can finely grid the variable space and accurately approximate F_i's in each grid (shown in F_1 plot), the proposed approach results in less integer variables.

A standard approach to convert bilinear constraints into MICPs is to grid the variable space (shown in Fig. 5.9 with F_1) and use McCormick envelopes to approximate the nonlinear constraint inside each grid. This approach accurately describes solutions for any problem parameter θ, but introduces hundreds of binary variables into the optimization problem [61], hence solving speed is slow. We solve problem P by introducing a 2-stage convex optimization process $P_1 \rightarrow P_2$. During the first stage P_1, we approximate the bilinear constraints in P_2 using McCormick envelopes and solve for both sets of bilinear variables p_{ij} and f_{ik}. In the second stage P_2, we choose to keep one set of bilinear variables in the solution of P_1 and project the other set onto the bilinear surface. If one set of bilinear variables are given, the bilinear constraint (5.13) in P_2 is linearized. Because of the McCormick envelope in P_1, the 2-stage planner is *coupled*, with the approximation of the nonlinear constraint in P_2 embedded in P_1. This process is shown in Fig. 5.10.

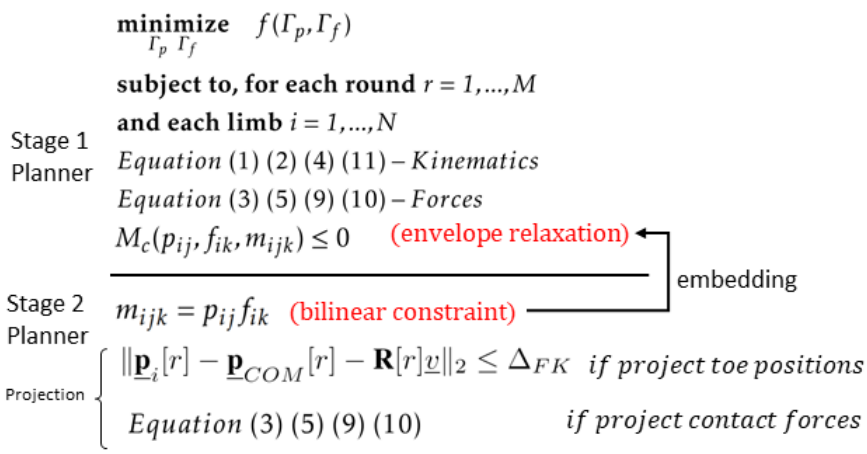


Figure 5.10: Optimization formulation of the coupled position and force planning problem

We propose to learn the best envelope to a certain type of problem with unsupervised learning, such that the envelope formulation captures the heuristics behind the given type of input. Since trajectory optimization solves a series of postures with identical mathematical formulation, it is sufficient to learn the constraints on a single posture. We generate data by solving the problem with $M = 1$ using the accurate MICP formulation [61], and collect data for bilinear variables. We then fit McCormick envelopes around the regions where data clusters, excluding the infeasible regions. Through this approach, the relaxation can be made tighter with a fixed number of envelopes.

5.2.2 Envelope Learning Algorithm

As solutions of similar types of problems could be densely populated in certain regions, learning methods, especially unsupervised learning, could be used to identify those regions to form a “tighter" McCormick envelope. This eliminates regions in the variables’ space where infeasible solutions are expected.

Our algorithm seeks the overlapping regions for bilinear variables $(\mathbf{p}, \mathbf{f})$. The position variable $\mathbf{p}$ to be learned is $\mathbf{p}^b$. We omit the superscript for simplic-

ity. Suppose we define problem $P(\theta_i)$ where θ_i is the parameter drawn from a distribution $G(\theta)$ and represents the terrain shape in our work. If $P(\theta_i)$ is feasible, we define the *solution set* $X(\theta_i) = \{\mathbf{x}|\mathbf{x} = (\mathbf{p}, \mathbf{f}, \mathbf{m}, \mathbf{y})$ feasible for $P(\theta_i)\}$. $\mathbf{p}, \mathbf{f}, \mathbf{m}$ are the variables that satisfy the bilinear constraint $m_d = p_d f_d$ in each dimension $d = 1, ..., D$, while $\mathbf{y}$'s are other optimization variables. If $P(\theta_i)$ is infeasible, we define the *infeasible solution set* $\overline{X}(\theta_i)$ by removing the bilinear constraint from $P(\theta_i)$ that generates $\overline{P}(\theta_i)$, and $\overline{X}(\theta_i) = \{\mathbf{x}|\mathbf{x} = (\mathbf{p}, \mathbf{f}, \mathbf{m}, \mathbf{y})$ feasible for $\overline{P}(\theta_i)\}$. A McCormick envelope relaxation of $m_d = p_d f_d$ [119] can be defined tightly over a pair of lower/upper bounds $[\mathbf{p}_d^L, \mathbf{p}_d^U]$ and $[\mathbf{f}_d^L, \mathbf{f}_d^U]$ that describes a rectangular region $\mathbf{B}$ over $[\mathbf{p}^T, \mathbf{f}^T]^T$. McCormick envelopes $M_c(\mathbf{B})$ satisfy the property such that $[\mathbf{p}^T, \mathbf{f}^T]^T \in \mathbf{B}$ and $m_d = p_d f_d$, $\forall d$ implies $M_c(\mathbf{B}) \leq 0$. Furthermore, $[\mathbf{p}^T, \mathbf{f}^T]^T \notin \mathbf{B} \Rightarrow M_c(\mathbf{B}) \not\leq 0$. Define S_i as the projection of $X(\theta_i)$ onto $\mathbf{p} \times \mathbf{f}$ ("$\times$" is Cartesian product) subspace giving all feasible $[\mathbf{p}^T, \mathbf{f}^T]^T$ for problem $P(\theta_i)$. Then, $\forall (\mathbf{p}, \mathbf{f}) \in S_i$, $\exists$ $(\mathbf{m}, \mathbf{y})$ such that $(\mathbf{p}, \mathbf{f}, \mathbf{m}, \mathbf{y})$ is feasible for $P(\theta_i)$. We also define the projection of *infeasible solution set* of $[\mathbf{p}^T, \mathbf{f}^T]^T$ as $\overline{S}_i$ by projecting $\overline{X}(\theta_i)$ onto $\mathbf{p} \times \mathbf{f}$ subspace.

Assume that for any feasible $P(\theta_i)$ that generates S_i, there exists an overlapping region: $\mathcal{S} = \cap S_i \neq \emptyset$. We can find a rectangular region $\mathbf{B}_\mathcal{S}$ inside $\mathcal{S}$ to construct a McCormick envelope relaxation $M_c(\mathbf{B}_\mathcal{S}) \leq 0$, and define the 2-step optimization process $P_1 \rightarrow P_2$. P_1 is defined by replacing the biliner constraints in $P(\theta_i)$ with the McCormick relaxation, and gives a solution $(\widetilde{\mathbf{p}}, \widetilde{\mathbf{f}}, \widetilde{\mathbf{m}}, \widetilde{\mathbf{y}})$. P_2 has identical constraints as P, but uses $(\widetilde{\mathbf{f}}, \widetilde{\mathbf{y}})$ (or $(\widetilde{\mathbf{p}}, \widetilde{\mathbf{y}})$) from P_1 to solve for $(\mathbf{p}^*, \mathbf{m}^*)$ (or $(\mathbf{f}^*, \mathbf{m}^*)$), as the bilinear constraints are linearized.

Theorem. *If there exist a McCormick envelope $M_c(\boldsymbol{B}_\mathcal{S}) \leq 0$ that satisfies the following 3 conditions:*

1. *$\forall$ θ_i that makes $P(\theta_i)$ feasible and generates S_i, $\boldsymbol{B}_\mathcal{S} \cap S_i \neq \emptyset$.*
2. *$\forall$ θ_i that makes $P(\theta_i)$ infeasible, $\boldsymbol{B}_\mathcal{S} \cap \overline{S}_i = \emptyset$.*

3. *(Complete Projectability)* $\forall \boldsymbol{f} \in \boldsymbol{B}_{\mathcal{S}}$, $\forall \boldsymbol{y}$, *(or* $\forall \boldsymbol{p} \in \boldsymbol{B}_{\mathcal{S}}$, $\forall \boldsymbol{y}$*) such that* $(\boldsymbol{f},\boldsymbol{y})$ *(or* $(\boldsymbol{p},\boldsymbol{y})$*) is feasible for* P_1, $(\boldsymbol{f},\boldsymbol{y})$ *(or* $(\boldsymbol{p},\boldsymbol{y})$*) is also feasible for* P.

Then the 2-step process $P_1 \to P_2$ *defined above satisfy:*

1. P *is feasible* $\Rightarrow P_1 \to P_2$ *is feasible. In addition, any feasible solution for* $P_1 \to P_2$ *is also feasible for* P.
2. P *is infeasible* $\Rightarrow P_1$ *is infeasible.*

Note that depending on how P_2 projects the solution of P_1, the required projectability condition (3) is different. If we keep $(\mathbf{f},\mathbf{y})$ and project $(\mathbf{p},\mathbf{m})$, then we need $(\mathbf{f},\mathbf{y})$ to be feasible for P, and vise versa.

Proof.
1) Suppose P is feasible with solution subspace S_i. Since $\mathbf{B}_{\mathcal{S}} \cap S_i \neq \emptyset$, $\exists [\mathbf{p}_i^T, \mathbf{f}_i^T]^T \in \mathbf{B}_{\mathcal{S}} \cap S_i$. $[\mathbf{p}_i^T, \mathbf{f}_i^T]^T \in S_i \Rightarrow \exists (\mathbf{m}_i, \mathbf{y}_i)$ such that $\mathbf{x} = (\mathbf{p}_i, \mathbf{f}_i, \mathbf{m}_i, \mathbf{y}_i)$ is feasible for P (satisfying each constraint except $M_c(\mathbf{p}_i, \mathbf{f}_i, \mathbf{m}_i) \leq 0$). In particular, $(\mathbf{p}_i, \mathbf{f}_i, \mathbf{m}_i)$ satisfies the bilinear constraint $m_d = p_d f_d$, $\forall d$. This together with $[\mathbf{p}_i^T, \mathbf{f}_i^T]^T \in \mathbf{B}_{\mathcal{S}}$ implies $M_c(\mathbf{p}_i, \mathbf{f}_i, \mathbf{m}_i) \leq 0$. Thus P is feasible $\Rightarrow P_1$ is feasible.
Now we show that P_1 is feasible $\Rightarrow P_2$ is feasible, and the solution of $P_1 \to P_2$ is feasible for P. Any feasible solution $(\mathbf{p},\mathbf{f},\mathbf{m},\mathbf{y})$ for P_1 satisfies $M_c(\mathbf{p},\mathbf{f},\mathbf{m}) \leq 0 \Rightarrow (\mathbf{p},\mathbf{f}) \in \mathbf{B}_{\mathcal{S}}$. By condition (3), $(\mathbf{f},\mathbf{y})$ (or $(\mathbf{p},\mathbf{y})$) is feasible for P, which means $\exists (\widetilde{\mathbf{p}}, \widetilde{\mathbf{m}})$ (or $(\widetilde{\mathbf{f}}, \widetilde{\mathbf{m}})$) such that $\widetilde{\mathbf{x}} = (\widetilde{\mathbf{p}}, \mathbf{f}, \widetilde{\mathbf{m}}, \mathbf{y})$ (or $(\mathbf{p}, \widetilde{\mathbf{f}}, \widetilde{\mathbf{m}}, \mathbf{y})$) satisfies P. Since P_2 has identical constraints as P, $\widetilde{\mathbf{x}}$ is feasible for P_2. As P_2 is feasible and has identical constraints as P, any solution for P_2 is feasible for P.
2) By condition (2), $\mathbf{B}_{\mathcal{S}} \cap \overline{S}_i = \emptyset$, $\forall [\mathbf{p}^T, \mathbf{f}^T]^T \in \overline{S}_i$ (satisfies all but the bilinear constraints), $[\mathbf{p}^T, \mathbf{f}^T]^T \notin \mathbf{B}_{\mathcal{S}}$, thus $M_c(\mathbf{p},\mathbf{f},\forall \mathbf{m}) \not\leq 0 \Rightarrow P_1$ is infeasible. □

Remark. *It may look like result 2) is too strong. Normally, a relaxation is feasible doesn't guarantee that the original problem is feasible. However, in this case it does.*

*Basically, 2) shifts the region of relaxation on ($\boldsymbol{p}, \boldsymbol{f}$) completely away from the **danger zone** - region that potentially makes infeasible problems feasible. The relaxation will admit new $\boldsymbol{m}$ points. However, no matter what $\boldsymbol{m}$ is, since ($\boldsymbol{p}, \boldsymbol{f}$) is infeasible for the original problem, ($\boldsymbol{p}, \boldsymbol{f}, \boldsymbol{m}, \boldsymbol{y}$) wll not be feasible. A good example is that, for some problem, $\boldsymbol{m} = \boldsymbol{pf}$ does not constraint anything. This means whenever θ_i is infeasible for P, $\overline{P}$ is still infeasible. In this case, the envelope can be all the domain, as other constraints for P are already making the problem infeasible.*

The idea behind finding $\mathbf{B}_{\mathcal{S}}$ is shown in Fig. 5.9. Condition (3) guarantees projectability for any point in $\mathbf{B}_{\mathcal{S}}$. Strictly satisfying it guarantees P_1 will not give solutions that cause P_2 to be infeasible. In practice, verifying condition (3) is very difficult. We put additional safety factors to make constraints in P_1 even tighter to reduce the the relaxation (at the risk of P_1 not a complete relaxation of P). As an simple extension of this paper, one can also learn an overlapping region of y_i's and formulate as an additional constraint into P_1. This guarantees that P_1 does not give any "strange" y_i that makes P_2 infeasible. Another approach is to set an optimal criteria for P_1 and guarantee that the optimal solution of $(\mathbf{f}, \mathbf{y})$ is always projectable to P_2 [120]. If the projection still fails with the above efforts, the problem is over-relaxed, suggesting that the envelope should be divided into multiple smaller pieces. Interesting future work remains where an objective function that guarantees projectability at optimal points could potentially exist. In addition, if we do not confine the formulation to convex ones, we can use nonlinear optimization (NLPs) in P_2 to do the projection. In this case, we only keep y_i from P_1 and project $(\mathbf{p}, \mathbf{y})$ simultaneously. Properly initialized NLPs can have fast speed with a solvable rate close to 100% [61]. Finally, even if $\cap S_i = \emptyset$, we can still identify multiple mutually exclusive $\mathbf{B}_{\mathcal{S}}$'s that give multiple envelopes. An example is shown in the next section.

We present two data-based approaches to identify $\mathbf{B}_{\mathcal{S}}$, one based on clustering to directly fit envelopes and the other one based on evolutionary algo-

Algorithm 1 DataGeneration

Input Number of samples N

Initialize feasible solution set S and infeasible solution set $\overline{S}$
while $i < N$ **do**
  Sample $\theta_i \sim G(\theta)$
  Solve $P(\theta_i)$ with high precision MICP
  if $(\mathbf{p}_i, \mathbf{f}_i)$ is a feasible solution for $P(\theta_i)$ **then**
    Add $(\mathbf{p}_i, \mathbf{f}_i)$ to S
  else
    Remove bilinear constraint (5.13) to produce problem $\overline{P}(\theta_i)$
    Solve $\overline{P}(\theta_i)$
    if $(\overline{\mathbf{p}}_i, \overline{\mathbf{f}}_i)$ is a feasible solution for $\overline{P}(\theta_i)$ **then**
      Add $(\overline{\mathbf{p}}_i, \overline{\mathbf{f}}_i)$ to $\overline{S}$
  Increase i
return $S, \overline{S}$

rithms.

5.2.2.1 Clustering Approach

Based on the high-accuracy MICP formulation [61], we can generate feasible solutions or prove infeasibility for problems $P(\theta_i)$ sampled from $G(\theta)$. We recognize that if we sample the same amount of feasible solutions inside each S_i, the overlapping region $\mathcal{S}$ will receive more samples, indicating a clustering approach may identify $\mathcal{S}$. We also need to draw samples from the *infeasible solution set* $\overline{S}_i$ and make the envelope exclude those points. The data generation algorithm is shown in Algorithm 1, where we generate a random terrain from a pre-defined distribution and collect the subsequent optimization's solutions depending on their feasibilities.

Having collected the data in S and $\overline{S}$, we use Algorithm 2 to fit the envelopes. With a specified number of envelopes, the algorithm first performs clustering with Gaussian Mixture Models (GMMs) to identify the center of clusters. Then, an optimization problem named Boundary_fit is solved to fit the largest rectangular region B around the centers excluding any points in $\overline{S}_i$.

Algorithm 2 EnvelopeFitting

Input Threshold probability p_{th}, Number of clusters n, DataGeneration's S, $\overline{S}$

Initialize dictionary $\mathbf{B}_{\mathcal{S}}$
for $k = 1, ..., n$ **do**
    Get cluster means (p_k, f_k) = GMM(S)
    B_k = Boundary_fit((p_k, f_k), $\overline{S}$)
    S_k = points in S that are in B_k
    S_k^o = GMM_filter($p > p_{th}$)
    $\mathbf{B}_{\mathcal{S}\_k}$ = get_exterior_boundary(S_k^o)
    Add $\mathbf{B}_{\mathcal{S}\_k}$ to $\mathbf{B}_{\mathcal{S}}$
return $\mathbf{B}_{\mathcal{S}}$

Multiple formulations can be used to achieve this. We used a formulation based on mixed-integer programming, but other options exist [25]. We then collect all points in B but remove those whose probability of belonging to the cluster is less than the threshold p_{th}. The exterior boundary formed by the remaining points gives $\mathbf{B}_{\mathcal{S}}$.

5.2.2.2 Evolutionary Approach

Contrasting to the fitting approach, a sampling-based evolutionary approach could, by nature, find bounds that are even tighter. Using a sufficient number of random terrain (input) and feasibility (output) pairs, a genetic algorithm (GA) can be tailored to solve a bilevel optimization that is indicative of the original problem at hand. While mostly following the conventional GA approach, we choose our chromosomes to be the lower and upper bounds of the envelope, while uniform crossover is done per lower/upper bound pair as opposed to per gene value. We design the fitness function to be representative of the bilevel optimization that occurs between the two stages of the original problem. To achieve this, we define two key metrics that help in finding better envelopes. We define a to be the percentage of original infeasible values becoming feasible, and b to be the percentage of original feasible values becoming infeasible.

At each solution's fitness calculation, a random pair of terrain parameter and ground truth feasibility z_g are selected from a pre-generated dataset built using the data generator. Then, the individual and the terrain parameter are used to find the feasibility z_1 in P_1. If z_g is infeasible while z_1 is feasible, a increases, whereas if z_g is feasible but z_1 is infeasible, b increases. Per generation, each individual is tested against K number of terrains from the training set and the average a and b are used in the calculation of the fitness function. While we try to minimize a, we keep b below a certain threshold δ. To ensure that the number of mutations also decreases over the generations and that the variance of the population's fitness decreases, the mutation rate is set to be a function of the generation number decreasing over time.

5.2.3 Training results

To validate our proposed algorithm, we choose the random distribution θ to provide two different kinds of terrains—ground and wall—whose $(\mathbf{p}, \mathbf{f})$ are expected to show distinct distributions. For training, we provide one terrain mesh to each leg, while varying terrain position, orientation, and friction coefficient μ randomly. For ground data, we uniformly sample the angle of normal vectors within the 30° region around the straight-up direction, while varying μ between [0.1, 0.8]. The wall data are collected to plan trajectories for the robot to climb up between two walls [111] with pure frictional contact. We vary the angles within the 30° region around the nominal direction as shown in the top right of Fig. 5.12, and vary μ between [0.1, 1.2]. We use Algorithm 1 to gather a set S of 500 feasible points and 500 infeasible points for envelope fitting, and another set $\overline{S}$ of the same number for validation. We set the number of envelopes to be 1. Both ground and wall envelopes are fit with 3 legs (two right, one left). Envelopes are also fitted with 5 legs on the wall. For validation, we separate the success rate into two categories showing respectively if

our convex optimization can identify feasible solutions correctly and identify infeasible solutions correctly. We show the results of both stages. If $\mathbf{B}_{\mathcal{S}} \in S_i$, P_1 should remain feasible if P is feasible. If condition (2) in our theorem holds, P_1 should be infeasible if P is infeasible. If condition (3) holds, the feasible solutions for P_1 should be projectable to make P_2 feasible, thus success rate for feasible solutions should not drop from P_1 to P_2. We use both convex and NLP methods mentioned in the previous section for P_2. For convex method, we use $\mathbf{p}^b$ from P_1 to solve $\mathbf{f}$ in P_2. The NLP method solves $\mathbf{p}^b$ and $\mathbf{f}$ together.

Expert human heuristics are used to create envelope parameters as baselines. Bounds for $\mathbf{p}^b$ are measured outer boundaries of each leg's workspace, while that for $\mathbf{f}$ are from our understanding of force profiles. Normal force bounds are $[0.0, 0.9]/[0.0, 0.5]$ for climbing/walking, as large normal forces are expected to prevent slipping. Shear force bounds are $[-0.3, 0.3]/[-0.15, 0.15]$ for climbing/walking. For wall climbing the vertical shear force boundeds are $[0.0, 0.4]$ as they need to counteract the gravity. We perform hand-tuning to optimize the performance. The results are in Table 5.2.

The results show that P_1 tends to match P well for all test cases when using clustering, as both success rates are close to 100%. For convex projection, about 20%∼30% of P_1's solutions cannot be projected for 3 leg case. This is due to a violation of condition (3) indicating a single envelope may over-relax the original problem. For the 5 leg test, both stages perform well. Intuitively, since the problem dimension is higher, $\mathbf{p}$ has more room to adjust at P_2. NLP projections perform well, with the rate of projection close to 100% and solving speed no more than a few hundred milliseconds.

Aside from training a single cluster, we tried to fit envelopes with combined ground and wall data. By giving $n = 2$ in Algorithm 2, two mutually exclusive envelopes representing the ground and the wall are identified. This results in an MICP formulation with one binary variable $z \in \{0, 1\}$ per posture that

Table 5.2: Validation results for trained envelopes

Problem		P_1		$P_1 \rightarrow P_2$	$P_1 \rightarrow P_2$ (NLP)
		correct feasible	correct infeasible	correct feasible*	correct feasible
3 leg ground	Cluster	98.16%	77.78%	83.54%	98.16%
	GA	70.08%	91.24%	37.40%	70.08%
	Heuristic	99.77%	50.00%	84.91%	99.77%
3 leg wall	Cluster	93.62%	80.00%	63.32%	93.62%
	GA	77.85%	97.72%	67.07%	77.85%
	Heuristic	78.72%	51.33%	60.75%	78.72%
5 leg wall	Cluster	95.27%	79.31%	90.41%	93.49%
	GA	68.21%	98.68%	62.70%	68.21%
	Heuristic	92.90%	72.41%	88.10%	92.25%

* Correct infeasible results for P_2 are identical to P_1 thus omitted.

switches between 2 modes as the robot traverses from one type of terrain to the other.

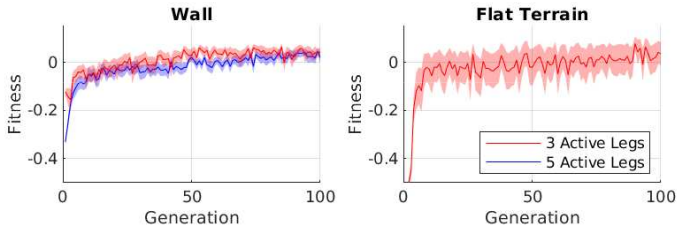


Figure 5.11: The average fitness of the population is shown with its variance. Certain individuals maximize the fitness early on in the generation.

Contrary to the clustering approach, the GA approach shows lopsided results. This could be an artifact of suboptimal initial solutions and a lack of generations as seen by the variance in Fig. 5.11. However, the extremely high infeasibility detection suggests that possibly a combined approach between a structured clustering and a conventional learning based approach could achieve higher accuracy.

The results suggest that our methodological approach based on data immediately provides comparable results to heuristics that require expert knowledge.

5.2.4 Planning results

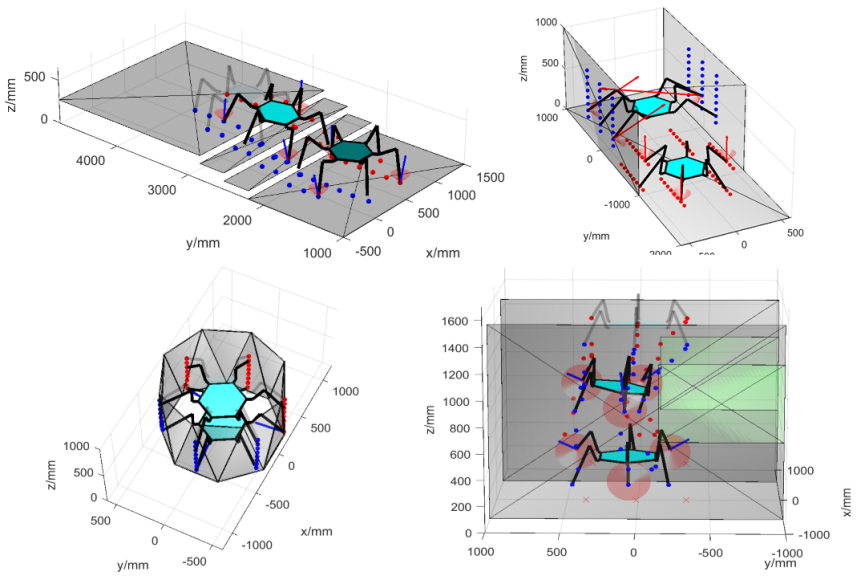


Figure 5.12: Motion plans generated by coupled position and force planner. Top left: the robot climbs stairs. Top right: the robot switches from walking to climbing. Bottom left: the robot climbs up inside a tube. Bottom right: the robot climbs up between two walls and avoid patches of low friction materials.

We use the learned envelopes in our 2-stage planner, and plan the motion for a 24 DoF hexapod. Trajectories to traverse in multiple terrains are found, including walking on flat ground with stairs, climbing between two flat vertical walls of varying friction, and climbing inside a tube. The coupling achieved in the planner is compared against the decoupled approaches [7]. The planner is verified on hardware. All results are included in the accompanying video [121].

5.2.4.1 Walking on Ground

The proposed planner generates trajectories for walking on a flat ground with stairs (Fig. 5.12 top left). This shows the feasibility of the approach on a simple environment because the terrains are effectively flat.

5.2.4.2 Climbing between Walls

This problem is first studied by [111], which uses a stiffness based approach allowing the robot to brace between two walls and climb. A 2-stage decoupled approach is then used [7] to plan the climbing motion, where it plans the position **p**'s without any knowledge of force. To test the proposed coupled planner, we plan the trajectory for the robot to climb on the walls with patches of zero friction $\mu = 0$ (green regions in Fig. 5.12, bottom right) and $\mu = 1$ in other areas. P_1 will have to place the toes outside the region to avoid P_2 being infeasible. For this, decoupled planner [7] would fail. We also plan a climbing motion inside a tube consisting of meshes at different angles (Fig. 5.12, bottom left), to test the adaptability to irregular walls. In both cases, the planer solves feasible motion plan on hardware.

5.2.4.3 Automatic switching motion plan from ground to wall

We demonstrate that with the learned 2 mode formulation, trajectories with contact forces that automatically switch between walking and climbing can be generated as seen in Fig. 5.12 top right. Beyond finding a feasible trajectory, the planner finds $z = 0/z = 1$ when the robot is on the ground/wall, indicating that it interprets the current motion as walking/climbing. This result shows the interpretability of the proposed planner. If we divide the variable space into smaller pieces and assign an integer variable for each piece, the information found by the planner can quickly get submerged by the large number of possible combinations of integer variables. Instead, our planner gives interpretable information that humans can process.

Table 5.3: Solving time for vertical two flat wall climbing

Test Name	Rounds M	Variables	Constraint	Solve Time [s]*
2-stage Convex	8	4128	1176	0.16
MICP without Envelope	4	16608 (276 binary)	3360	91
MICP with Envelope	4	16608 (276 binary)	4008	33
MICP without Envelope	8	33216 (552 binary)	6720	> 1,000
MICP with Envelope	8	33216 (552 binary)	8016	152

* Data taken on an Intel i5-6260U 1.80GHz machine with Gurobi [122]

5.2.4.4 Solving Time

We benchmark the solving time on the problem of climbing between two flat walls with uniform μ. Table 5.3 row 1 shows the solving time for the proposed two-stage convex planner. The convex solver generates trajectories in hundreds of milliseconds. Comparison with similar works using NLPs suggests a possible speed up of around 100 times [6]. This justifies the motivation behind using a multi-staged convex algorithm. The designed envelope can also be used to speed up MICP. We compare the accurate MICP formulation with and without our learned envelope as an additional constraint for the problem of climbing between two flat walls. A typical MICP solver deals with integer variables through a branch and bound algorithm [123] that expands nodes on each binary variable. Since [61] does not utilize the problem-specific knowledge, the solver wastes time expanding nodes inside regions that we already know are infeasible. Our approach reduces the solving time, and has minimum impact on the solutions (Table 5.3, row 2-4).

5.2.5 Discussion

A 2-stage motion planner is designed based on convex optimization, with the inter-stage coupling formulated as McCormick envelopes learned from data. We performed learning through clustering and GA approaches and validated against labeled data. The results show that a smaller number of integer vari-

ables and envelopes tailored to the type of problem can reduce solve time and help interpret the outcome. We also demonstrated the planner on the hardware.

Finer relaxations with smaller envelopes could be possible if one envelope is insufficient. Since we require exploring the solution set, which could be efficient on hardware, training on hardware is of interest. While we focused on a specific problem, this work may be generalizable for more complicated problems with multiple stages, and a larger class of nonlinear constraints.

5.3 Transition Planning

As we have demonstrated SiLVIA's capability of walking and climbing between two walls using motion planning, a transition trajectory planner between wall and ground is required. The role of this planner is to connect the ground and wall trajectories such that full mobility over space can be automatically realized. This work can be done through a pure open loop trajectory shown by Fig. 5.14. A more comprehensive and autonomous work using linear complementary constraints is done in [11], where in addition to the traditional tripod gait (3-3 gait), non-traditional gaits such as 2-2-2, 3-1-2-2 can also be realized. We briefly summarize the results here. For more details, please refere to [11]. We apply the same idea of [64] exploiting the complementarity condition between the contact distance and the contact force to our motion planning algorithm. As shown in Fig. 5.13, the contact force can be generated only when the contact distance is zero. Therefore, either the contact distance $(\mathbf{p}^c_{i,j})_z$ or the normal contact force $(\mathbf{f}^c_{i,j})_z$ will be zero which can be express as:

$$(\mathbf{f}^c_{i,j})_z \geq 0, \quad (\mathbf{p}^c_{i,j})_z \geq 0 \tag{5.17a}$$

$$(\mathbf{f}^c_{i,j})_z (\mathbf{p}^c_{i,j})_z = 0 \tag{5.17b}$$

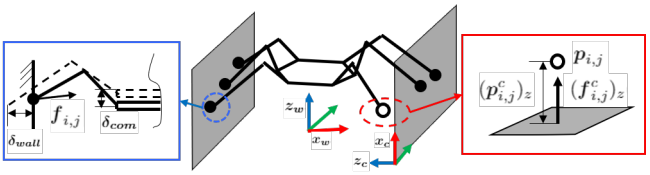


Figure 5.13: Illustration of the world frame $\{w\}$ and the local contact frame $\{c\}$. Solid dots indicate toes on the wall while the hollow one indicates the swing leg. Inside the red box is the complementarity condition between contact distance $(\mathbf{p}^c_{i,j})_z$ and the normal contact force $(\mathbf{f}^c_{i,j})_z$. Inside the blue box is the limb compliance for indirect force control of SiLVIA.

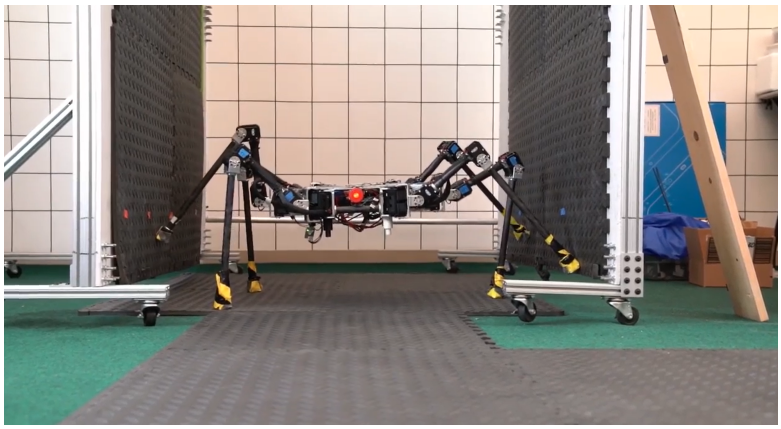


Figure 5.14: SiLVIA walking to climbing transition

For hardware demonstration, we let the robot stand between two parallel walls at a distance of 1230 mm. Initially, the robot stands on the ground with the body height as 210 mm. The desired configuration on the wall is the starting point of the climbing trajectory from our previous work [7]. The walls are covered by rubber pads and the robot toes are covered by anti-slip tapes, which gives a frictional coefficient μ around 1. With the nominal values for safety factors ($S_\tau = 1.8$, $S_\mu = 1.1$). The planner is able to generate more traditional tripod (3-3) or amble (2-2-2) contact sequence from ground to wall. Some of the more investigations are shown in Fig. 5.15, where we put two bricks next to the left wall on the ground. Intuitively the robot would be able to step on the brick for the intermediate round. To take advantage of the brick, we decrease the kinematics range for the legs on the left side until the leg is unable to reach

the desired positions on the wall in one step. The 3-3-3 contact sequence is generated which puts left front leg (LF) and left rear leg (LR) on the brick and right middle leg (RM) on the right wall, then moves right front leg (RF), right rear leg (RR) and left middle leg (LM) on the wall, finally moves LF, LR and RM again to the desired positions on the wall, as shown in Fig. 5.15. However, in the hardware experiment, the robot tips over when it is trying to lift 3 legs based on the supporting from 2 legs on the brick and 1 leg on the wall. The two legs on the brick (LF and LR) slip causing the failure. The reason is that the friction coefficient between the brick and the toe is much smaller than the one between the toe and the wall. We add one more round to enrich the possible contact sequence that the planner can search over. From the bottom row of Fig. 5.15. We can see that for the second round, the robot would not lift 3 legs together as what the robot would do in the 3-3-3 contact sequence. Instead, it is divided into two rounds. The robot would lift the leg LM to create 4 contact points and then lift the right two legs (RF and RR) to avoid too large horizontal forces required for the two legs on the brick. The robot successfully overcomes the transition phase with steps by performing the 3-1-2-2 contact sequence.

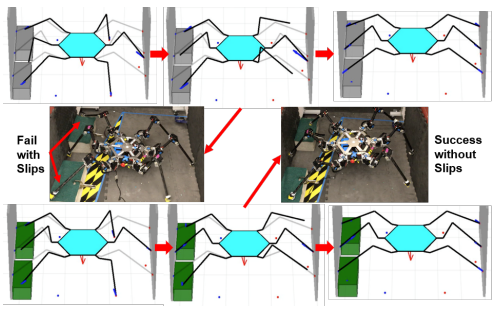


Figure 5.15: Visualization and hardware demonstration of planned transition motion. The upper row is the 3-3-3 contact sequence for the parallel wall with steps while the bottom row is the 3-1-2-2 contact sequence. The middle row is the screenshots of hardware experiments when implementing these two contact sequences. (Green represents a surface with smaller friction coefficient)

CHAPTER 6

Motion Planning Algorithm for Multi-modal Multi-agent Self-reconfigurable Robot System

After the successful implementation of the optimization-based motion planner on the multi-legged walking and wall-climbing robots, a natural next step is to test the planner on problems that are larger in scale and requires faster solving speed. In this chapter, we demonstrate using optimization-based motion planning to generate kinematics and dynamic trajectories for a modular reconfigurable robot system LIMMS (Latching Intelligent Modular Mobility System) for package delivery. This is a good example of solving motion planning problems on a larger scale. We set up the discrete logic constraints and continuous kinematics and dynamics constraints, use the Alternating Direction Method of Multipliers (ADMM) to solve the system, show the planning results, and discuss the issues that prevent those algorithms from real-time implementations. The materials in this section mainly come from [124]..

LIMMS was introduced in the recent work [125] as a modular approach to last-mile delivery, shown in Fig. 6.1. LIMMS system is composed of individual modules that resemble a 6 degree-of-freedom (DoF) arm but with wheels and a latching mechanism at both ends. As such either side of LIMMS can act as the base depending on the need. In the case of last-mile delivery, within the delivery truck, it is assumed that the surface, as well as boxes, have anchor points LIMMS can attach the latch to. By attaching one end to the walls within the vehicle LIMMS can carry and move boxes by latching its free end to it like other manipulators. To actually transport the box to the recipient's door, four LIMMS can attach to a box and use it as its body to walk itself there. Finally, to

return LIMMS can self-balance and wheel itself back. LIMMS introduces challenging motion planning problems, each LIMMS has multiple DoF, and this requires kinematic constraints which were rarely done [126] [127]. Moreover, each LIMMS has different approaches to locomoting itself: as a wheeled robot, as a leg, or as an arm. The delivery package can also be transported in different methods: to be manipulated or be moved as the body of a quadruped robot. This introduces complicated mixing of discrete decisions and continuous constraints.

6.1 Background on Modular Re-configurable Robots

Various approaches have been explored to resolve the reconfiguration planning problem, such as graph search methods [126], reinforcement learning [128], or optimization-based approaches [129]. We adopt optimization-based methods for LIMMS. Among the optimization-based methods, mixed-integer programs (MIP) are useful for discrete decision-making within multi-agent systems [16, 130]. However, the LIMMS motion planning problem includes nonlinear constraints such as kinematics. Therefore, the problem becomes a mixed-integer nonlinear (non-convex) program (MINLP). MINLPs are known to be computationally difficult when the problem scale is large. Aside from directly applying commercialized solvers, e.g., SCIP, there are generally two approaches that transform the MINLP problems into MIPs using linear approximations of the nonlinear constraints, or into nonlinear programs (NLPs) using complementary constraints to replace binary variables. Unfortunately, as the problem scales up, nonlinear constraints yield a large amount of piece-wise linear approximations making the problem very slow [23], and complementary constraints tend to cause infeasibility without a good initial guess [34].

In this paper, we implemented the alternating direction method of multi-

pliers (ADMM) to solve the MINLP problem, inspired by [131, 132]. ADMM decomposes the problem into two sub-problems: an MIP problem and an NLP problem. By decomposition, the problem scale of both MIP and NLP is reduced and becomes more tractable. The logic constraints about the connections of LIMMS are formulated into MIP while the nonlinear kinematics constraints are formulated into NLP. We establish general rules for such a system to operate under and use ADMM to explore possible approaches to resolve the given scenario, showing the feasibility of our proposed method as well as the richness of the system.

6.2 System Description

Hardware. To gain an intuition of the constraints in the optimization formulation, the physical limitations and design of the first LIMMS hardware prototype are detailed in this section and depicted in Fig. 6.1. LIMMS is a symmetric 6 DoF robot. At full length, a single LIMMS unit stretches to about 0.75 m and weighs roughly 4.14 kg including batteries. From preliminary tests in simulation with a target package payload of 2 kg, we determined off-the-shelf Dynamixel motors from ROBOTIS were sufficient for the prototype. With our custom gearbox, it reaches a peak velocity of 2 rad/s and a peak torque of 31 Nm. For interested readers previous recent work in [125] goes further in-depth on the overall design.

One of the crucial aspects of LIMMS is its latching mechanism, which allows it to attach either end to anchor points or itself. Latching will be used frequently since it is how LIMMS transitions to different modes and grab objects for manipulation. The current prototype proposed in [133] consists of a radially symmetric multi-blade design. Fig. 6.1 (bottom right) depicts the latch prototype. By the geometry of the blade and the anchor point hole pattern, the

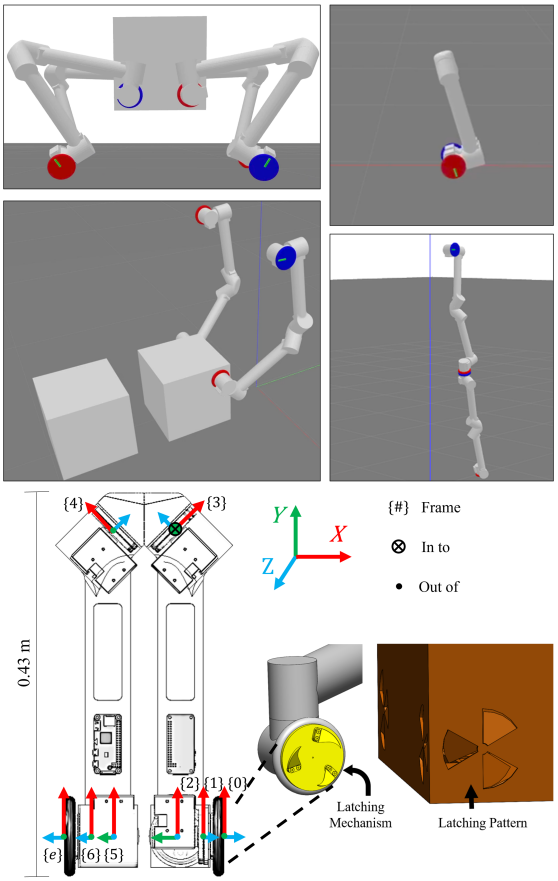


Figure 6.1: *Top* shows the various operational modes for LIMMS which will be considered for the optimization formulation. *Upper Left*: 4 LIMMS attached to a box in quadruped mode. *Upper Right*: 1 LIMMS self-balancing to move. This is called free mode. *Lower Left*: 2 LIMMS anchored to surfaces in manipulator mode. *Lower Right*: LIMMS attached to each other. This can be used in both quadruped and manipulator mode. *Bottom* shows the joint axis and the latching mechnanism to be integrated at the end effector.

latch mechanically self-aligns when it rotates.

The mechanism is designed to maximize robustness to misalignment in position and orientation about its center axis, see [133].

This has a dual effect of easing control effort and allowing for the latches to pull the box or itself into the desired position and orientation without being fully positioned. If LIMMS is kinematically constrained and cannot fully reach the target position and orientation, there is slack created by the latch's mechan-

ical design. In this sense when latching to an anchor point, the alignment does not need to strictly satisfy its constraint.

Modes of Operation. Through latching, LIMMS can enter many different modes to complete its tasks. For our proposed planner we only consider three different modes as these will be sufficient for delivering packages. However, LIMMS has the potential for many other modes, some of which are described in [125]. Fig. 6.1 depicts the 3 modes considered in our optimization formulation: 4 LIMMS attached to a box are in quadruped mode, LIMMS attached to walls or surfaces are in manipulator mode, a single LIMMS which wheels around or move like a snake is in free mode. The last sub-figure in the top half, lower right shows two LIMMS attached to the end of each other. This can be viewed as a sub-skill accessible to all modes.

6.3 Problem Formulation

In this section, we demonstrate the optimization formulation for LIMMS motion planning. The objective function of the optimization problem is to minimize the distance from the box center position to the target box position, i.e. deliver the box to the goal. The constraints consist of two parts: logic, which is formulated into constraints using integer variables, and kinematics or dynamics, which is formulated into linear or nonlinear equations of motion. As a result, the problem is an MINLP, which will be separated into an MIP and NLP to be then solved with ADMM. Assume there are B boxes and L LIMMS. Each box has $S_B = 4$ anchor points at the center of each face. The optimization is run from $t = 1,...,T$ time steps. We use upper case letters to indicate constants such as the number of boxes B, the number of anchor points S_B, and use the lower case letters to index the quantities such as $b = 1,...,B$, $s_b = 1,...,S_B$. i is used to index the binary variables. Upper case letters are also used as variable

	Var	Dim	Description
Continuous	$z_{B,i}$	$[B,T]$	Mode for box b at time t. $i = \{1:$ stable object, $2:$ free object, $3:$ manipulated, $4:$ quadruped$\}$
	$z_{L,i}$	$[L,T]$	Mode for limb l at time t. $i = \{1:$ free, $2:$ arm, $3:$ add arm $4:$ leg, $5:$ add leg$\}$
	$z_{S,i}$	$[B,S_B,L,T]$	Mode of connection for anchor point s_b on box b to limb l at time t $i = \{1:$ empty, $2:$ to arm, $3:$ to leg$\}$
	$z_{W,i}$	$[S_w,L,T]$	Mode of connection for anchor point s_w on wall to limb l at time t $i = \{1:$ empty, $2:$ to arm$\}$
	$z_{Ac,i}$	$[L_{pr},L_{po},T]$	Mode for connection s.t. limb l_{po} connects as an additional limb to l_{pr} $i = \{1:$ not connected, $2:$ connected$\}$
	$z_{Lc,i}$	$[L_{pr},L_{po},T]$	Mode for connection s.t. limb l_{po} connects as an additional leg to l_{pr} $i = \{1:$ not connected, $2:$ connected$\}$
	$\mathbf{p}_B$	$[B,T]$	Position of center of box b at time t
	$\mathbf{R}_B$	$[B,T]$	Orientation of box b
	$\mathbf{c}_B$	$[B,C,T]$	Position of corner c for box b at time t
	$\mathbf{p}_L$	$[J,L,T]$	Position of joint j of limb l at time t
	$\mathbf{R}_L$	$[J,L,T]$	Rotation matrix of joint j of limb l
	$\mathbf{f}_L$	$[B,S,L,T]$	Contact force at anchor point s of box b from limb l at time t
	$\mathbf{a}$	$[L_1,L_2,T]$	Normal vector of separating plane for l_1 and l_2
	b	$[L_1,L_2,T]$	Offset of separating plane for l_1 and l_2

Table 6.1: Table of optimization variables

names. For example, $z_{B,i}$ are binary variables associated with boxes. We state the assumptions made for this formulation:

1. The box does not rotate and the momentum is assumed to be balanced. This simplifies the dynamics constraints. In practice, this minimizes the damage to the contents in boxes during shipping.

2. Multi-body dynamics of LIMMS are not enforced. This is to simplify the constraints.

All binary and continuous variables are summarized in Table 6.1 except those that pertain to enforcing collision avoidance with the environment for simplicity (i.e. binary variables $\delta_{Ba,i}$, $\delta_{Bg,i}$, $\delta_{La,i}$, $\delta_{Lg,i}$, and the corresponding

$\lambda \in [0,1]$ variables for convex combination). Note *pr* is short for previous, and *po* is short for post.

6.3.1 Integral Logic Constraints

Mode for Boxes. We define 4 modes for each box represented by 4 binary variables: $z_{Bi}[b,t], i = 1,...,4$ for the mode of box b at t. Mode 1 is stable object mode, where the box is supported by the ground. Mode 2 is free object mode where the box is in the air subject to gravity. Mode 3 is manipulated object mode where the box is connected to a manipulator. Mode 4 is quadruped mode where the box is used as a robot body. We currently only allow quadruped robot for walking. This can be relaxed to incorporate more solutions such as simultaneously bipedal walking while manipulating boxes as in [134]. At each t, a box is subject to 1 mode, such that: $\sum_{i=1}^{4} z_{Bi}[b,t] = 1 \ \forall b, \ \forall t$.

Mode for LIMMS. We define 5 modes for each LIMMS represented by 5 binary variables: $z_{Li}[l,t]$, where $i = 1,...,5$ indicating the mode of LIMMS l at t. Mode 1 is free (wheeled) mode, where the corresponding LIMMS unit moves on the ground like a Segway robot. Mode 2 is manipulation mode, where LIMMS connects to one connection site on the wall and may connect to one box to manipulate it. Mode 3 is add arm mode, where a LIMMS can connect to another LIMMS to extend the length of the arm for a larger workspace. Mode 4 is leg mode, where LIMMS connects to a box and serves as a leg. Mode 5 is add leg mode, where it can connect to another LIMMS to extend the length of the leg similar to mode 3. At any time step, LIMMS can only be in one mode, such that: $\sum_{i=1}^{5} z_{Li}[l,t] = 1 \ \forall l, \forall t$.

Mode for Anchor Points. Each box has 4 anchor points on each side face. We define 3 modes for each anchor point by 3 binary variables $z_{Si}[b, s_b, l, t]$, where $i = 1,...,3$, denoting the connection mode for anchor point s_b on b to l at t. Mode

Table 6.2: Table of logic rules

	#	Logic Rule Description	Mathematical Formulation
Quadruped	1	Box b in quadruped mode, all 4 s_b is connected to leg mode LIMMS.	$z_{B,4}[b,t]=1 \implies \sum_l z_{S,3}[b,s_b,l,t]=1\ \forall s_b$
	2	Anchor point s_b on box b in leg mode, box b is in quadruped mode.	$z_{S,3}[b,s_b,l,t]=1\ \exists l,\ \exists S_B \implies z_{B,4}[b,t]=1$
	3	Anchor point s_b is connected to LIMMS l as a leg, l is in leg mode.	$z_{S,3}[b,s_b,l,t]=1 \implies z_{L,4}[l,t]=1$
	4	LIMMS l in leg mode, it's connected as a leg to one anchor point.	$z_{L,4}[l,t]=1 \implies \sum_s \sum_b z_{S,3}[b,s_b,l,t]=1$
Manipulation	5	b in manipulated mode, at least 1 s_b is connected to 1 arm mode l.	$z_{B,3}[b,t]=1 \implies \sum_l \sum_{s_b} z_{S,2}[b,s_b,l,t] \geq 1$
	6	s_b is connected to l in arm mode, l is in arm or add arm mode.	$z_{S,2}[b,s_b,l,t]=1 \implies z_{L,2}[l,t]=1$ or $z_{L,3}[l,t]=1$
	7	l is in arm mode, l is connected to one s_w on the wall or ground.	$z_{L,2}[l,t]=1 \implies \sum_{s_w} z_{W,2}[s_w,l,t]=1$
	8	s_w on wall or ground, s_w is connected to l, l is in arm mode.	$z_{W,2}[s_w,l,t]=1 \implies z_{L,2}[1,t]=1$
	9	LIMMS l is in add arm mode, it's connected to one other LIMMS.	$z_{L,3}[l,t]=1 \implies \sum_{l_{pr}} z_{Ac}[l_{pr},l,t]=1$
	10	LIMMS l_{pr} is connected to l_{po}, l_{pr} is in arm or add arm mode, l_{po} is in add arm mode.	$z_{Ac}[l_{pr},l_{po},t]=1 \implies \begin{cases} z_{L,2}[l_{pr},t]=1 \\ z_{L,3}[l_{po},t]=1 \end{cases}$
	11	LIMMS l_{pr} is connected to l_{po}, l_{pr} cannot connect to any box.	$z_{Ac}[l_{pr},l_{po},t]=1 \implies z_{S,2}[b,s_b,l_{pr},t]=0$
Free	12	Box b is in stable or free object mode, all anchor points s_b are empty.	$z_{B,1}[b,t]=1$ *or* $z_{B,2}[b,t]=1 \implies z_{S,1}[b,s_b,l,t]=1\ \forall s_b\ \forall l$

1 is empty mode, where s_b on b is not connected to l. Mode 2 is arm mode where l connects to s_b as an arm. Mode 3 is leg mode where l connects to s_b as a leg. Their summation has to be 1 at each time step: $\sum_{i=1}^{3} z_{Si}[b,s_b,l,t] = 1\ \forall b,\ \forall s_b,\ \forall l,\ \forall t$. In addition, at each time s_b can connect to no more than 1 LIMMS, while a given LIMMS can connect to no more than 1 s_b at its base point. This introduces two more constraints: $\sum_b \sum_{s_b} z_{Si}[b,s_b,l,t] \leq 1\ \forall l,\ \forall t$ and $\sum_l z_{Si}[b,s_b,l,t] \leq 1\ \forall b,\ \forall s_b,\ \forall t$. s_b on b is associated with a physical connection. If latching is enforced, the position and orientation of the base of LIMMS is constrained:

$$\begin{aligned} \mathbf{p}_L[j=0,l,t] &= \mathbf{p}_B[b,t] + \mathbf{R}_B[b,t]\mathbf{o}[s_b] \\ \mathbf{R}_L[j=0,l,t] &= \mathbf{R}_o[s]\mathbf{R}_B[b,t] \end{aligned} \tag{6.1}$$

Where $\mathbf{o}[s]$ is the constant offset vector from the center of the box b to the anchor point s_b. $\mathbf{R}_o[s_b]$ is the constant rotation matrix from the box frame located at the geometric center of the box to the s_b frame located at anchor point s_b. This conditional equality constraint can be enforced through big-M formulation such that if $z_{S,i}=1$, (6.1) is enforced.

Each anchor site on the wall s_w has two modes. Mode 1 is empty mode

where s_w is empty, and mode 2 is manipulation mode where l connects to s_w as an arm. Multiple s_w can exist on the ground. Two binary variables $z_{Wi}[s_w, l, t]$, where $i = 1, 2$ are used to represent those modes. The associated mode constraints and physical connection constraints are similar to the anchor points on the boxes. We also define binary variables for additional connections between LIMMS as arms or legs: $z_{Ac,i}[l_{pr}, l_{po}, t]$ or $z_{Lc,i}[l_{pr}, l_{po}, t]$, $i = 1, 2$. If $z_{Ac,i}[l_{pr}, l_{po}, t] = 1$, LIMMS l_{po} connects as an additional arm to LIMMS l_{pr} at t. Similar for $z_{Lc,i}$.

Logic for Boxes as Robot Bodies. We define 4 logic rules for any box detailed in rule 1 – 4 in Table 6.2. They constrain the boxes in quadruped mode and LIMMS units connected to it. The gist is to ensure that several things happen simultaneously: 1) b is used as the robot body, 2) 4 LIMMS are its legs, and 3) connections happen between them. On the other hand, if no box is used as a quadruped body, no LIMMS should be used as legs. This is enforced through formulating the rules into a loop as shown in figure 6.2. If 1 LIMMS is used as leg, it should connect to 1 anchor point on 1 of the boxes due to Logic 4, and the corresponding box should be in quadruped mode due to Logic 2.

Each logic can be formulated as constraints between integer variables through big-M formulation. For example, Logic 1 can be written as $1 - M(1 - z_{B,4}[b, t]) \leq \sum_l z_{S,3}[b, s_b, l, t] \leq 1 + M(1 - z_{B,4}[b, t])$ where M is a large constant (usually 10^5). Other constraints follow similarly.

Logic for Boxes as Manipulated Objects. We define 7 logic rules for boxes as objects and are manipulated by LIMMS as arms. They again constrain the modes for the box and LIMMS connected to it. The gist is to ensure that the following happens simultaneously: 1) box is being manipulated by 1 or more LIMMS, 2) 1 or more LIMMS operate in arm mode, 3) connections occur between the arm and box, and 4) arm is connected to 1 anchor point on the wall or ground. While most of the logic rules are single directional, the bidirec-

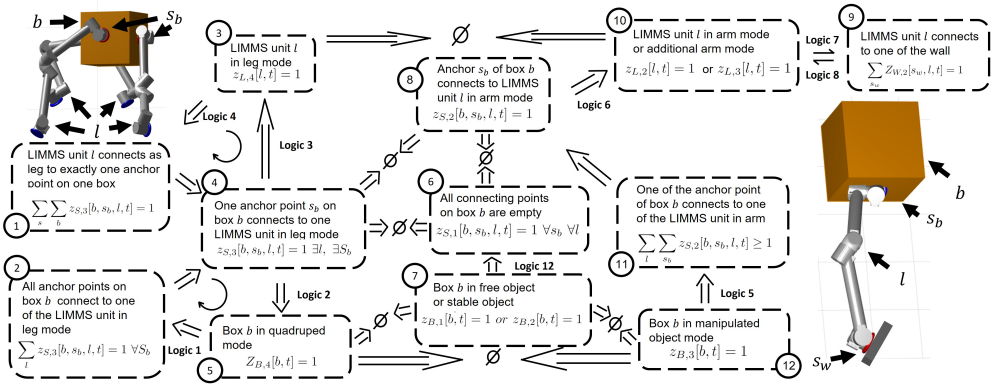


Figure 6.2: Implications of logic 1-12. Left half is associated with robot mode while right half is associated with manipulation mode. Arrows are labeled with specific logic rules. If not, the implication is mathematically correct. The symbol $\Rightarrow \emptyset \Leftarrow$ means mutually exclusive.

tional rules are enforced by formulating implicit loops. For example, there is no explicit rules enforcing a box to be in manipulated object mode (arrow from 8 to 12) if LIMMS arm is manipulating it (block 8 in Fig. 6.2). However, block 8 is mutually exclusive with block 4 and 6. This means that if 8 happens, 4 and 6 cannot happen, which means block 5 and 7 cannot happen (by reversing the direction of implication). This in turn indicates that 12, which is the complement of 5 and 7, has to happen. Note that we did not include the constraints for additional arm or leg mode in Fig. 6.2, but similar arguments carry over.

Logic for Boxes as Stable or Free Objects. Logic 12 in Fig. 6.2 is defined for boxes as stable or free object modes which states that if a box is stable or free, all its anchors are empty.

6.3.2 Continuous Constraints

Kinematics. Kinematics constraints are imposed for each LIMMS through a series of linear constraints and bilinear constraints in the same fashion as [61]:

$$
\begin{aligned}
&\mathbf{p}[j+1,l,t] = \mathbf{p}_L[j,l,t] + \mathbf{R}_L[j,l,t]\mathbf{p}_{j+1,j} \\
&\mathbf{R}_L[j-1,l,t]\mathbf{z}_{j-1,j} = \mathbf{R}_L[j,l,t]\mathbf{z}_{j,j} \\
&\mathbf{R}_L[j,l,t]\mathbf{R}_L[j,l,t]^T = \mathbf{I} \\
&\mathbf{R}_L[j,l,t] \text{ represents a right handed frame}
\end{aligned}
\tag{6.2}
$$

Where $\mathbf{p}_{j+1,j}$ is the constant position vector of the next joint as seen in the frame of the previous joint, and $\mathbf{z}_{j-1,j}$ is the constant orientation of the next joint as seen in the frame of the previous joint, where ours are $\mathbf{z}_{j-1,j} = [0,0,1]^T$.

Collision Avoidance with Environment. To enforce constraints such that LIMMS and boxes does not collide with the environment, we model the environment into discrete convex regions. All boxes and LIMMS have to stay within the convex regions during the process. We also need to discriminate if the LIMMS or box is making contact with the ground. This introduces additional binary variables $\delta_{Ba,i}$, $\delta_{Bg,i}$, $\delta_{La,i}$, and $\delta_{Lg,i}$. Note subscript g stands for ground, and a stands for air. If LIMMS and boxes are within a convex region, the joint points of LIMMS and corners of boxes are linear combinations of the vertices of the convex region:

$$
\mathbf{p} = \sum_v \lambda_v \mathbf{V}_v, \quad \sum_v \lambda_v = \delta, \quad \lambda_v \in [0,1] \tag{6.3}
$$

Where $\mathbf{p}$, λ and δ are associated with either corner points of the box $\mathbf{c}_B$ or position of joints of LIMMS $\mathbf{p}_L$ as listed in Table 6.1. λ_v's represent the vertices of the convex region. If one region is not selected, all λ_v's are zero due to $\delta = 0$.

Collision Avoidance Between Agents. To enforce collision avoidance for LIMMS-LIMMS and LIMMS-box contact, we use the formulation from [34] that uses separating planes. For convex polygons, the two polygons do not overlap with each other if and only if there exists a separating hyperplane $\mathbf{a}^T\mathbf{x} = b$ in be-

tween [135]. That is, for any point $\mathbf{p}_1$ inside polygon 1 then $\mathbf{a}^T\mathbf{p}_1 \leq b$, and for any point $\mathbf{p}_2$ inside polygon 2 then $\mathbf{a}^T\mathbf{p}_2 \geq b$. Our problem uses the following constraints:

$$\begin{gathered} \mathbf{a}^T\mathbf{p}_L[j,l_1,t] \leq b,\ \ \mathbf{a}^T\mathbf{p}_L[j,l_2,t] \geq b \\ \mathbf{a}^T\mathbf{c}_B[c,b,t] \leq b,\ \ \mathbf{a}^T\mathbf{p}_L[j,l,t] \geq b\ \forall t,\ \ \mathbf{a}^T\mathbf{a} \geq 0.5 \end{gathered} \tag{6.4}$$

Where $\mathbf{a}$ and b is the normal vector and offset for planes associated with the specific pair. 0.5 is just an arbitrary nonzero number that we choose, as $\mathbf{a}$ does not necessarily need to be a unit vector. With this method, we enforce collision avoidance purely through inequality constraints and avoid using complementary constraints such as [136].

Dynamics for Box. The dynamics are required for the agent to generate strictly feasible motions. However, enforcing dynamics for each LIMMS is expensive given its high DoF. We only enforce dynamics for the boxes. This serves as two purposes. First, it allows the system to generate dynamic motions such as throwing or jumping. Second, it allows the system to select motion plans based on the box weight.

When the box is in stable object mode, the gravity is compensated for by the ground. Additionally, LIMMS can only apply reaction forces to the box when it is connected through the anchor points on the box. We define the reaction force on the end effector of a LIMMS l to the box b as $\mathbf{f}_i[b,s,l,t]$, where the index s indicates that the force is through the anchor point s_b. When LIMMS connects to the box as a leg, $\mathbf{f}_i$ serves as the contact force on the ground, while when LIMMS connects to the box as an arm, $\mathbf{f}_i$ serves as the contact force to grasp the

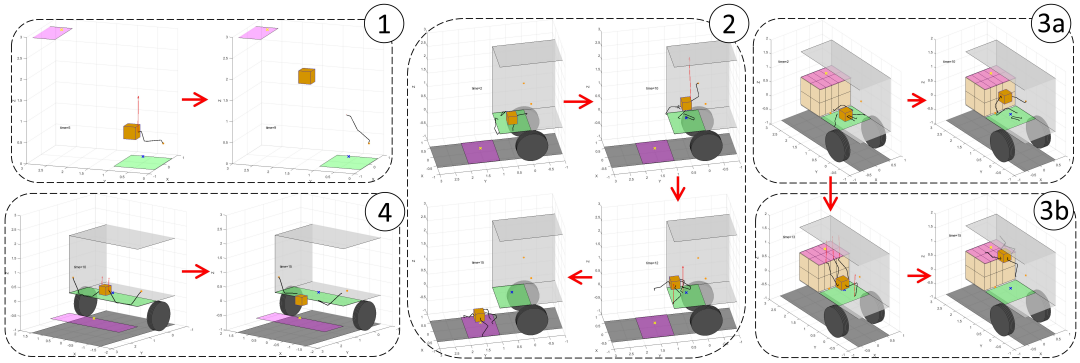


Figure 6.3: Results for 5 experiments. The green rectangle represents the initial region in which LIMMS is constrained in. LIMMS is trying to get the box to the magenta region, where LIMMS is allowed to move in as well.: 1) LIMMS picks up a box and throws it towards the goal. 2) LIMMS lifts box in manipulator mode and switches to quadruped mode to jump out of the vehicle. 3a) LIMMS attempts to use manipulator mode to reach goal. 3b) LIMMS climbs onto a step using quadruped mode. 4) LIMMS sends the box towards the goal using dual double arm manipulation to enlarge the workspace.

box. The box dynamics are:

$$m\ddot{\mathbf{p}}_{box}[b,t] = \sum_{s=1}^{4}\sum_{l=1}^{L}\mathbf{f}_i[b,s,l,t] - m\mathbf{g}(1 - z_{B,1}[b,t]) \tag{6.5}$$

$$\sum_{s=1}^{4}\sum_{l=1}^{L}(\mathbf{p}_L[j=6,l,t] - \mathbf{p}_B[b,t]) \times \mathbf{f}_i[b,s,l,t] = 0 \tag{6.6}$$

$\mathbf{f}_i[b,s,l,t]$ only exists when l is connected to the box:

$$\mathbf{f}_i[b,s,l,t] = 0 \;\text{ if }\; z_{S,1}[b,s_b,l,t] = 1 \tag{6.7}$$

When l connects to the box as leg mode, $\mathbf{f}$ represents the contact force from the ground. Therefore, $\mathbf{f}$ needs to satisfy the friction cone constraint:

$$\mathbf{f} \in Cone \;\text{ if }\; z_{S,i}[b,s_b,l,t] = 1 \text{ and } \delta_{Lg}[j=6,l,\exists p,t] \tag{6.8}$$

Since LIMMS has a loading capacity, we enforce the max norm constraint on any contact force: $\|\mathbf{f}\| \leq f_{max}$. Note that there is a contact moment across the latching. Missing this moment results in the box incapable of being manipulated with a single contact (moment balance will be violated). For simplicity, we fix the box orientation: $\mathbf{R} = \mathbf{I}$. This can be justified as in manipulated mode, the contact moment is sufficient to keep the orientation of the box, and in quadruped mode, the four ground support points is sufficient to keep the body orientation.

Support Polygon. The balance of moment constraint should guarantee the stability of the quadruped body. However, there are already many nonlinear constraints such as kinematics. To simplify the NLP formulation, one approach is to enforce a simple stability constraint in replacement of the moment balance constraint. Since the rotation matrix is fixed, we can simply enforce that the end effector of LIMMS stays to one side of the body which guarantees that the body's center of mass lies within the support polygon of the foot.

Stability of LIMMS. One drawback of our formulation is that we do not include the dynamics of LIMMS. As a compensation, there should be a constraint to guarantee the stability when LIMMS is on the ground and tries to reach an anchor point.

We just assume that we have many latching points on the ground so realizing this motion is relatively simple. Therefore, when LIMMS is in free mode, we enforce that the base stays on the ground and give a speed constraint:

$$\|\mathbf{p}_L[j,l,t+1] - \mathbf{p}_L[j,l,t]\| \leq \Delta\mathbf{P} \;\text{ if }\; \delta_{Lg}[j=0,l,p,t] = 1 \tag{6.9}$$

Continuity of Connection. One suboptimal solution to avoid is having LIMMS frequently latch on and off an anchor point. We enforce equality constraints for binary variables within a range $z_{S,i}[t] = ... = z_{S,i}[t+n]$. We usually choose n be-

tween 3 and 5, decided based on the speed of latching.

6.4 ADMM Formulation

Collecting the constraints defined previously, the problem to solve becomes:

$$
\begin{aligned}
&\underset{z,\,\delta,\,\mathbf{p},\,\mathbf{R},\,\mathbf{f},\,\lambda,\,\mathbf{a},\,b}{\text{minimize}} \quad f_{obj} \\
&\text{subject to} \\
&\textbf{Mixed integer constraints:} \\
&\text{Logic rules 1-12} \\
&\text{collision avoidance with the environment (6.3)} \\
&\text{Dynamics constraints: (6.5), (6.7), (6.8)} \\
&\text{Stability of LIMMS on ground (6.9)} \\
&\textbf{Nonlinear constraints:} \\
&\text{Kinematics (6.2)} \\
&\text{Collision avoidance between agents (6.4)} \\
&\text{Dynamics constraints (6.6)}
\end{aligned}
\tag{6.10}
$$

Where f_{obj} is a quadratic equation that minimizes the distance of the box to the goal position. The variables and constraints of problem (6.10) incorporate discrete and continuous variables with linear and nonconvex (bilinear) constraints. This results in an MINLP. The commercial solvers tend to perform non-satisfactory in this type of problem. There are generally two approaches that convert this type of problem: an MICP using convex envelope relaxations for nonlinear constraints [61] or conversion of the discrete variables into continuous ones through complementary formulation [63]. MIPs with convex envelopes tend to solve slowly when the problem scales up. Since there are many discrete variables in this problem, complementary formulations will be nu-

merically difficult [132]. In this paper, we adopt the ADMM. ADMM separates the problem into two sub-problems. Although those sub-problems have different constraints, ADMM iterates between sub-problems such that constraint 1 which may not appear in sub-problem 2 will be implicitly enforced as the iteration proceeds. In the end, sub-problems will reach a consensus meaning their solutions are close to each other. This procedure is detailed in Algorithm 3. In our problem, the logic rule constraints are resolved through MIPs, while the nonlinear kinematics and collision avoidance constraints are resolved through NLPs. Similar to [131], we first make copies $\mathbf{var}_2$ of the variables $\mathbf{var}_1 = [z, \delta, \mathbf{p}, \mathbf{R}, \mathbf{f}, \lambda, \mathbf{a}, b]$. Represent the feasible set of mixed-integer constraints through $0 - \infty$ indicator function by $\mathcal{I}_M$ and the nonlinear constraints by $\mathcal{I}_N$. The consensus problem between MIP and NLP is:

$$\begin{aligned} &\underset{\mathbf{var}_1\ \mathbf{var}_2}{\text{minimize}} \quad f_{obj} + \mathcal{I}_M(\mathbf{var}_1) + \mathcal{I}_N(\mathbf{var}_2) \\ &s.t. \quad \mathbf{var}_1 = \mathbf{var}_2 \end{aligned} \tag{6.11}$$

The constraints are moved to the objective function through the indicator function. Applying ADMM [137] to the Lagrangian $\mathcal{L}$ of (6.11) results in three iterative operations:

$$\mathbf{var}_1^{i+1} = \text{argmin}_{\mathbf{var}_1} \mathcal{L}(\mathbf{var}_1^i,\ \mathbf{var}_2^i,\ \mathbf{w}^i) \tag{6.12a}$$

$$\mathbf{var}_2^{i+1} = \text{argmin}_{\mathbf{var}_2} \mathcal{L}(\mathbf{var}_1^{i+1},\ \mathbf{var}_2^i,\ \mathbf{w}^i) \tag{6.12b}$$

$$\mathbf{w}^{i+1} = \mathbf{w}^i + \mathbf{var}_1^i - \mathbf{var}_1^i \tag{6.12c}$$

Where $\mathbf{w}$ is the dual variable of the Lagrangian of (6.11). In (6.12), (6.12a) solves the MIP problem:

$$\begin{aligned} &\underset{\mathbf{var}_1}{\text{minimize}} \quad \|\mathbf{var}_1^i - \mathbf{var}_2^i + \mathbf{w}^i\|_{\mathbf{W}_{\text{MIP}}^k} \\ &\text{s.t. Mixed-integer constraints in (6.10)} \end{aligned}$$

In the next step, (6.12b), solves the NLP:

$$\begin{aligned} \underset{\mathbf{var}_2}{\text{minimize}} \quad & \|\mathbf{var}_2^i - (\mathbf{var}_1^{i+1} + \mathbf{w}^i))\|_{\mathbf{W}_{\text{NLP}}^k} \\ \text{s.t. Nonlinear constraints in } & (6.10) \end{aligned}$$

And the next step, (6.12c), updates the dual variable $\mathbf{w}$. To finish one iteration, the weights for MIP, W_{MIP}, the weights for NLP, W_{NLP}, and the dual variable $\mathbf{w}$, are updated with line $6-7$ in Algorithm 3. Within one iteration, (6.12a), (6.12b), (6.12c) are solved in succession. This iterative procedure continues until the discrepancy between the MIP solutions and the NLP solutions $\theta = \mathbf{var}_1^i - \mathbf{var}_1^i$ are lower than the user-set error threshold θ_{th}.

It is well known that ADMM has convergence guarantees for convex problems and can significantly improve the solving speed. However, for complex MINLPs, there is no convergence guarantee. In this problem, both MIP and NLP can be slow and expensive. We avoid explicitly placing complementary constraints to represent discrete modes as [131] did, since it hinders convergence for NLP. However, NLP does need some information on discrete variables as it needs to reason connections and turn variables such as $\mathbf{f}$ on or off accordingly. After solving the MIPs, we directly use the solutions of $z_{S,i}$, $z_{W,i}$ in the NLP step to enforce connections. This improves the precision of the NLP step and the overall precision of consensus. The price to pay is an increase in difficulty for NLP solvers to find solutions.

6.5 Results

We performed 5 numerical experiments to evaluate the performance of the proposed formulation and ADMM algorithm. For all experiments, the MIP formulation was solved with Gurobi 9, and the NLP formulation was solved through Ipopt on a Intel Core i7-7800X 3.5GHz × 12 machine. We solved all

Algorithm 3 ADMM for LIMMS

Input ρ, $\mathbf{W}^0_{\text{MIP}}$, $\mathbf{W}^0_{\text{NLP}}$, $\mathbf{w}^0$, $\mathbf{var}^0_2$, θ_{th}

Initialization $i = 1$
while $\theta > \theta_{th}$ **and** $i < i_{max}$ **do**
  Compute $\mathbf{var}^{i+1}_1$ via (6.12a)
  Compute $\mathbf{var}^{i+1}_2$ via (6.12b)
  $\mathbf{w}^{i+1} \longleftarrow \mathbf{w}^i + \mathbf{var}^i_1 - \mathbf{var}^i_1$
  $\mathbf{W}^{k+1}_{\text{MIP}} \longleftarrow \rho \mathbf{W}^k_{\text{MIP}}$, $\mathbf{W}^{k+1}_{\text{NLP}} \longleftarrow \rho \mathbf{W}^k_{\text{NLP}}$
  $\mathbf{w}^{i+1} \longleftarrow \mathbf{w}^i / \rho$
  $\theta \longleftarrow \mathbf{var}^i_1 - \mathbf{var}^i_1$
  $i = i + 1$
return $\mathbf{var}^i_2$

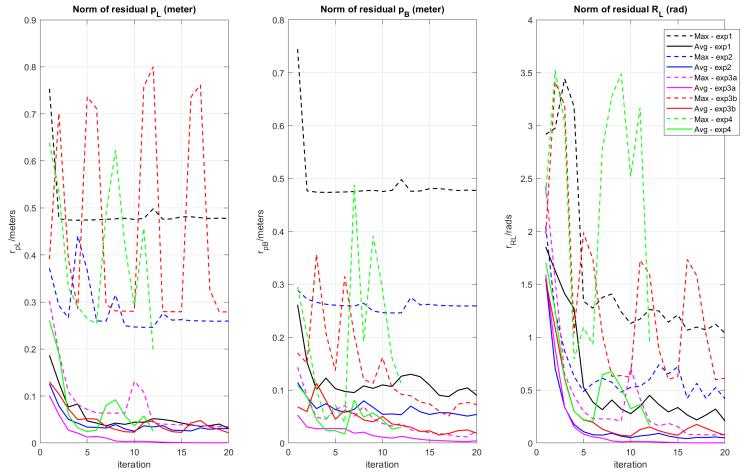


Figure 6.4: Convergence of mean and max residual r_{pB}, r_{pL}, r_{RL} for experiment 1-4. Solid lines denote mean residuals and dashed lines denote max residuals.

the scenarios for 15 iterations. Other parameters we have chosen are: $\Delta t = 1sec$, $f_{max} = 20N$, max moving speed for LIMMS on the ground of $0.5m/s$, and a $30cm$ cube box. The 6 Dof LIMMS unit lengths are $5cm$, $7.4cm$, $33cm$, $7.8cm$, $33cm$, $7.4cm$, $5cm$, respectively for each link starting at one end. Fig. 6.3 shows the MATLAB visualization of the first 4 experiments. Animations of all tests are included in the attached video available at `https://www.youtube.com/watch?v=RH9gMOK24L0`.

1) *Throwing*. In the first experiment, we placed 1 LIMMS and 1 box in the

scene and provided 1 anchor point on the wall. The goal is set higher than a LIMMS unit can reach. The solver gives a solution where the LIMMS unit connects to the wall and picks up the box, then throws the box to the goal. This simple test shows the ability of the solver to generate manipulation motions using dynamics.

2) *Jumping*. In the second experiment, we placed 4 LIMMS and 1 box on a raised platform or inside a truck and set the goal position to be lower on the ground. In addition, 2 anchor points are provided on the wall. The planner provides the solution where 1 LIMMS latches itself onto the wall and lifts the box. Then the other 3 LIMMS connects to the anchor points on the sides of the box which enters quadruped mode. This quadruped robot then jumps down the step to reach the goal. The solver automatically changes modes from manipulation to quadruped.

3a,b) *Weight Lifting*. In the third experiment, we investigate the behavior change due to a change in the weight of the box. We place 4 LIMMS and 1 box down a ledge and set the goal to be above the ledge. Two anchor points on the wall are provided. First, we set the box weight to be 0.5kg. The planner provides a manipulation trajectory where two LIMMS connect to the wall and lift the box to move it to the goal. We then increase the box weight to 7kg. In this case, if we force LIMMS to manipulate the object, the planner returns infeasible. The feasible trajectory returned by the solver set 3 LIMMS to be anchored onto the ledge and box, while 1 leg stays down from the ledge to push the body up onto the platform. As the box weight exceeds the capacity of dual arm manipulation, a quadruped motion is necessary to lift the 7kg box.

4) *Manipulation with Double-dual Arms*. In the fourth experiment, we again place 4 LIMMS and 1 box in the same scenario, below a desired platform or step. Two anchor points are provided on the wall. However, we enlarged the width of the truck such that 1 LIMMS cannot reach the box from the wall. In

this case, the solver connected a second LIMMS as an additional arm to the first which is connected to the wall, allowing the arms to reach the box and perform dual arm manipulation.

5) *Quadruped Walking with Refinement*. In the fifth experiment, we placed 4 LIMMS and a box on flat ground and set the goal to be at the same elevation but separated by a distance. LIMMS moved to the box and assembled a quadruped robot which then moved towards the goal. Furthermore, since there is no gait optimization in our problem formulation, we used an additional planner similar to [65] but included 6 DoF kinematics to provide the quadruped walking motion to the goal. This experiment demonstrated that, although the planner may give rough trajectories, they can be further refined with another planner to correct kinematics discrepancies and gait cycles. This readies it to be implemented on the hardware.

6) *Box Lifting on Hardware* In this experiment, one LIMMS is initially anchored to the ground. There is another anchor point on the wall. The objective is to lift the box higher. If LIMMS lifts the box with the other end anchored to the ground, the kinematics quickly becomes infeasible. The planner instead lets LIMMS first anchor to the box, then anchor to the wall from the box. The LIMMS can easily lift the box higher with the other end anchored to the wall. The hardware implementation is included in the video.

7) *Convergence* Define the residual to be the mismatch between the MIP and NLP solutions. The mean residuals for $\mathbf{p}_B$, $\mathbf{p}_L$ and $\mathbf{R}_L$ are the mean value of all the norms:

$$r_{pB}[i] = \underset{b}{\text{mean}} \, \|\mathbf{p}_{B,\text{MIP}}[b,i] - \mathbf{p}_{B,\text{NLP}}[b,i]\| \tag{6.13a}$$

$$r_{pL}[i] = \underset{j,\, l}{\text{mean}} \, \|\mathbf{p}_{L,\text{MIP}}[j,l,i] - \mathbf{p}_{L,\text{NLP}}[j,l,i]\| \tag{6.13b}$$

$$r_{RL}[i] = \underset{j,\, l}{\text{mean}} \, \|\text{Vec}(\mathbf{R}_{L,\text{MIP}}[j,l,i] - \mathbf{R}_{L,\text{NLP}}[j,l,i])\| \tag{6.13c}$$

#	T	# of variables		# of constraints		Time in Minutes		
		MIP	NLP	MIP	NLP	T-MIP	T-NLP	T-Total
1	10	4857 cont. 670 bin.	1777	3623	eq. 582 ineq. 5206	0.88	7.17	8.05
2	15	25917 cont. 3150 bin.	8787	34545	eq. 621 ineq. 28990	129	24	153
3a)	15	25917 cont. 3150 bin.	8787	33102	eq. 1686 ineq. 28990	41	107	148
3b)	10	17277 cont. 2100 bin.	5857	22032	eq. 1131 ineq. 19180	5	50	55
4	15	25917 cont. 3690 bin.	8787	35924	eq. 606 ineq. 31342	19	244	263

Table 6.3: Solving time for experiment 1-4. Note: Far left column in ascending order is: 1 for Throw, 2 for Jump, 3a for Lifting with Arm, 3b for Lifting with Climb, and 4 for Double-dual Arm.

The max residual is the maximal value of all the norms. Figure 6.4 depicts the change of mean and max residuals as a function of time. ADMM generally showed decent average consensus after iteration 10, where r_{pB} or r_{pL} usually converges to cm-mm level and r_{RL} usually less than 0.1 rad. The maximal residual can sometimes be large. If we put the MIP solutions on the real hardware, we can run another kinematics refiner to solve the kinematics to ensure that the nonlinear constraints are strictly satisfied.

The number of variables, constraints and time cost for solving the experiments above are listed in Table 6.3. Generally, the NLP portion is the more challenging portion of ADMM, since it includes kinematic and collision avoidance constraints. To speed up the solving process, some linear constraints in can be moved into the MIP formulations. This will speed up the NLP solver but the residual may increase.

6.6 Conclusion: Why Do We Need Stronger Optimization Methods?

An optimization-based motion planner for the multi-agent modular robot system LIMMS is presented. We demonstrated solving the proposed formulation with ADMM. The results show how LIMMS autonomously coordinates between different modes and generates trajectories of the system under different situations. With proper refinement, the trajectories can be implemented on the hardware.

It is worthwhile to mention that due to the separating plane for collision avoidance, the NLP part of the problem takes a long time to solve. Table 6.3 shows that the solving time can be as long as hours. Clearly, this takes too long for real-time implementations. A well-trained human commander can easily exceed this speed. In addition, we have found that the optimal costs given by ADMM solutions are also not satisfactory [34]. Note that we have not yet put the most challenging multi-body dynamics inside the problem formulation. To conclude, the proposed ADMM method works in some cases, but is too slow and not reliable enough for real-time implementations.

ADMM works better than poorly initialized formulation with complementary constraint or mixed-integer formulation. However, with pre-solved data, we can ask for a faster solving speed and a better optimal cost. In the previous work [34], we used data-driven methods to solve those constraints fast online. A similar approach can be adopted here such that a learning agent can be trained to give a good initial guess for the separating planes.

CHAPTER 7

Data-driven Methods for Mixed-integer Non-convex Optimization: Algorithms

7.1 Background on Data-driven Methods for Optimization

7.1.1 Motion Library

Motion library is a typical method to leverage offline computing power. A large number of controllers are pre-computed offline and stored inside a library. Online, the controllers are selected based on the robot state in the nearest neighbor manner. This method can provide more robust and global policies faster than online solving techniques such as dynamic programming [138]. Motion libraries have been used to synthesize controllers for bipedal walking [139], and simulated multi-link swimming [140]. However, this method does not adapt well to situations not previously seen in the library.

7.1.2 Parametric Programming

Parametric programming is a technique to build a function that maps to the optimization solutions from varying parameters of the optimization problem [141, 142]. Previous works has investigated parametric programming on linear programming [143], quadratic programming [144], mixed-integer nonlinear programming [145]. As an implementation of parametric programming on controller design, explicit MPC [144, 146] tries to solve the model-predictive control problem offline, and stores active sets. When computed online, the problem parameter can be used to retrieve the active sets. [144] solved a con-

strained linear quadratic regulator problem with explicit MPC and proved that the active sets are polyhedrons. Therefore, the parameter space can be partitioned using an algorithm proposed by [147]. However, when the problem is non-convex, the active sets do not in general form polyhedrons. Therefore, it is significantly more complex to compute critical regions offline.

7.1.3 Learning Problem-solution Mapping

The basic ideas behind parametric optimization is to construct a mapping from the descriptor of the problem to the solutions of the problem. For simple problems, this may be done analytically with parametric optimization techiques. For more challenging problems, it makes more sense to use learning approaches. For relatively smaller datasets, previous works [28, 148, 149] directly store the data points and pick out warm-start using non-parametric learning such as K-nearest neighbor (KNN), and solve the online formulation. Effectively, this approach adds an online adaptation step to the motion library method, which turns out to work well. On the other hand, modern learning techniques such as neural-networks can learn an embedding of a larger set of parameters that maps to the solutions [27]. One advantage of using neural-network methods is the capability to deal with out-of-distribution situations that are not included in the training set [150].

7.1.4 System Identification Approach

A different perspective to look at the learning problem-solution mapping approach is to treat this as a system identification problem, where we intend to identify a function that transforms the problem parameter space into the solution space. In certain cases such as [144], the optimal control law is a piecewise affine (PWA) function. This indicates that we can use piece-wise affine

system identification tools to fit the problem-solution mapping. The classical approaches include [30] which utilized a clustering approach, and [151] which used polynomial factorization. An overview of the classical methods can be seen in [33]. Those methods, however, do not scale up to larger problems. More recent approaches such as [152] proposed methods that can scale to thousands of discrete modes. An interesting comparison can be made between the learning methods and system identification methods. In general, the system identification approach can be difficult to perform model training and does not scale as well as learning approaches such as neural networks. However, they are more interpretable as each mode is explicitly represented by planes. It is also easier to leverage the optimization formulation with explicit plane models. The system identification approach may be a valid choice for smaller-scale problems.

7.1.5 Online Formulation

To allow online adaptation, one needs to solve an online optimization formulation in real-time. This online formulation can have several forms. Typical robotics problem includes discrete contact and discrete operating modes. There are generally two ways to include discrete variables in the optimization formulation. Mixed-integer programs include continuous variables and explicit discrete variables. The definition of mixed-integer variables is that if the discrete variables are relaxed into continuous ones, the problem becomes convex [61]. Typical mixed-integer programming solvers use branch-and-bound methods [21], cutting plane methods [153], and rely on heuristics to choose the branch to explore. Despite the worst case of solving time, a large portion of problems only requires exploring a small portion of the search tree [123]. Mixed-integer programs have been implemented for online motion planning such as [22].

On the other hand, mathematical programs with complementary constraints (MPCC) models discrete modes through continuous variables with complementary constraints. Complementary constraints enforce a pair of variables such that if one of them is non-zero, the other one should be zero. This constraint is traditionally hard to solve and is sensitive to initial guesses. Algorithms such as time-stepping [154], pivoting [155], central path methods [156] are proposed to resolve complementarity. In the robotics community, complementary constraints are typically used to optimize over gaits for trajectory optimization [11, 64] or control with implicit contacts [157] where contact forces and distance to the ground are complementary with each other. On the other hand, complementary constraints can also be used to model binary variables [158].

7.1.6 Solving Techniques for MICP and MINLP

Mixed integer convex programs are defined such that

$$\begin{aligned} &\text{find} \quad \mathbf{x}, \mathbf{z} \\ &\text{s.t.} \quad \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} \in \mathcal{C} \\ &\mathbf{x} \in \mathbb{R}^{n_x} \quad \mathbf{z} \in \{0,1\}^{n_z} \end{aligned} \tag{7.1}$$

Where $\mathbf{x}$'s are continuous variables, $\mathbf{z}$'s are discrete (usually binary) variables. $\mathcal{C}$ is a convex set. The definition states that if the binary variables $\mathbf{z} \in \{0,1\}^{n_z}$ are relaxed into continuous variables $\mathbf{z} \in [0,1]^{n_z}$, the problem becomes a convex optimization problem. Mixed-integer programs are generally solved through branch-and-bound method [159, 160] and cutting plane methods [161]. The basic idea behind branch and bound algorithm is to relax the integer variables and solve the corresponding convex programming in an iter-

ative fashion, while keeping track of two bounds of global optimal solution: the upper bound - best solution for the convex programming, and the lower bound - best current feasible solution. Once those two bounds get close to each other, the algorithm converges. On the other hand, cutting planes generate constraints that are equivalent to the current constraint given that some variables are integers, but stronger in terms of its continuous relaxations. This method helps to quickly cutout globally infeasible regions. Aside from branch-and-bound and cutting planes, mixed-integer optimization solvers also rely on heuristics [38] that guides the search to e.g. which branch should be expanded in the branch and bound process.

Mixed integer nonlinear programs are defines such that

$$\begin{aligned} minimize \;\; & \mathbf{d}^T\mathbf{x} \\ \text{s.t.} \;\; & g_i(\mathbf{x}) < 0 \;\; \forall i \in \mathcal{M} \\ & L_i \leq x_i \leq U_j \;\; \forall j \in \mathcal{N} \\ & x_j \in \mathbb{Z} \;\; \forall j \in \mathcal{I} \end{aligned} \tag{7.2}$$

where $\mathcal{I} \subseteq \mathcal{N} := \{1, ..., n\}$ is the index set of integer variables, $\mathbf{d} \in \mathbb{R}^n$. L and U are lower and upper bounds on the variables. There are different approaches to relax MINLPs. One can omit the integer constraints resulting in nonlinear programming relaxations, or one can relax the nonlinear constraints (if such a relaxation exists), resulting in mixed-integer programs. One example is the McCormick envelope relaxation of bilinear constraints [62], or convex envelope on trilinear monomials [162]. However, such convex envelope relaxations of the non-convex constraints tend to generate mixed-integer programs with many integer variables and are slow to solve.

Many MIP heuristics rely on finding small sub-MIP which contains good solutions. This set of methods are usually called *Local Branching* (LNS). In [163]

the authors proposed local branching which imposes linear soft fixing constraints to “fix" a part of the integer variables given a feasible solution. In [164] the authors proposed RINS which fixes the variables that has the same values in the current best solution and the solution from global continuous relaxation, and solve a sub-MIP on the remaining variables. In [165], the authors proposed DINS which combines the soft fixing from local branching and hard fixing from RINS based on an intuition that the improved MIP solutions are more likely to be the close ones to the current relaxation solution. Those LNS improvement heuristics can be extended to more general case of MINLP [166–168]. Some of them are special to MINLPs [169]. Some of the recent works have also use learning methods to perform branch and bound [170], LNS [171,172]. Another perspective to look at this set of methods is that it uses heuristics to pick good solution and warm start the problem. Similar work has done for mixed-integer MPC [173].

7.1.7 Collision Avoidance with Mode Switch

Collision avoidance constraints typically appear in multi-agent motion planning problems. Single-agent collision avoidance with the stationary environment is relatively straightforward as the collision-free regions can be precomputed such that the moving agent stays within them. Polyhedrons are typical models for collision-free environment [174]. If moving agents have simple shapes such as circles, collision-free constraints can simply enforce that distance between two agents are larger than a known value [175]. However, if moving agents have complex shapes, collision avoidance becomes more challenging. Naive methods include enforcing collision avoidance between a large number of points sampled within each agent. This approach is obviously computationally expensive. Methods based on KKT conditions are proposed [176] to enforce collision avoidance between convex shapes. This method does not

allow the agent to make contact without sacrificing precision. In this paper, we propose a novel approach using separating planes to enforce collision avoidance between convex agents but still allow them to make precise contacts. If the agent has a non-convex shape, it can be decomposed into convex shapes such that this approach can still be used. For many multi-agent optimization problems, agents can operate under various modes encoded with discrete variables. The discrete mode switching with collision avoidance constraints leads to a mixed-integer bilinear formulation which is challenging to solve. Leveraging on pre-solved data is one approach to solve these formulations fast and reliably.

7.2 Data-driven Methods for Fast Online Optimization: Algorithms

Assume that we are given a set of problems parametrized by Θ that is drawn from a distribution $D(\Theta)$. For each Θ, we seek a solution to the optimization problem:

$$\begin{aligned} &\underset{\mathbf{x},\,\mathbf{z}}{\text{minimize}}\ f_{obj}(\mathbf{x},\mathbf{z};\Theta) \\ \text{s. t. }\ &f_i(\mathbf{x},\mathbf{z};\Theta) \leq 0,\ \ i = 1,...,m_f \\ &b_j(\mathbf{x},\mathbf{z};\Theta) \leq 0,\ \ j = 1,...,m_b \end{aligned} \tag{7.3}$$

Where $\mathbf{x}$ denotes continuous variables and $\mathbf{z}$ binary variables with $z_i \in \{0,1\}$ for $i = 1,...,dim(\mathbf{z})$. Constraints f_i are mixed-integer convex, meaning if the binary variables $\mathbf{z}$ are relaxed into continuous variables $\mathbf{z} \in [0,1]$, f_i becomes convex. Constraints b_j are mixed-integer bilinear, meaning that relaxing the binary variables gives bilinear constraints. Without loss of generality, $\mathbf{x}$ and $\mathbf{z}$ are assumed to be involved in each constraint. We omit equality constraints in (7.3) as they can be turned into two inequality constraints from opposite directions.

In general, there are two directions to convert a mixed-integer bilinear program to either a mixed-integer linear program or a nonlinear program. To convert to a mixed-integer linear program, one needs to convert non-convex constraints to mixed-integer linear constraints. For example, we can convert trigonometry constraints into piece-wise linear constraints [18], or convert bilinear constraints into mixed-integer envelope constraints [61]. However, mixed-integer envelope constraint tend to significantly increase the solving time. In [23], the trajectory planner is based on mixed-integer formulation with envelopes to approximate the bilinear moment constraints. To walk a few steps upstairs, the solver can take several hours to find a feasible solution. The other approach is to convert the binary variables into continuous variables with complementary constraints. This approach requires the solver to deal with a potentially large number of complementary constraints known to be computationally difficult. Directly solving the NLP with complementary constraints without an informed initial guess result in high chance of infeasibility, unless special treatment such as [154] are used. Those methods, however, result in slower solving speed from our benchmark results.

For both directions mentioned above, incorporating pre-solved data to learn a good warm-start online can improve the solving speed or rate of feasibility, as we will show in our experiments. If the learning agent with the problem description Θ can partially provide feasible integer variables, the size of the mixed-integer optimization problem can be reduced and hence solved faster. If the learning agent can provide a complete list of integer variables, the problem reduces to a convex optimization permitting convex solvers such as OSQP [177]. On the other hand, if the learning agent provides a good initial guess for the variables involved in complementary constraints, the feasibility to solve the problem significantly increases.

As the online formulation can be MPCC, MIP, or a convex formulation, we

have compared their performances on benchmark problems. In the following sections, we describe the implementation details for each online formulation.

7.2.1 Complementary Formulation

An equivalent formulation of mixed-integer programs with binary variables is to turn all the binary variables z_i into continuous variables with complementary constraints, such that the complete formulation can be solved through NLP solvers. One implementation is from [158]:

$$z_i(1 - z_i) = 0 \tag{7.4}$$

Which is equivalent to the constraint $z_i \in \{0, 1\}$. Other implementations can be found in [63].

After collecting the dataset, we can learn the optimal problem-solution mapping from the problem parameter Θ to the solution $(\mathbf{x}, \mathbf{z})$. Online, the learner samples multiple warm-start points $(\mathbf{x}, \mathbf{z})$ for the NLP. The NLP with complementary constraints are then solved using the sampled warm-start one-by-one until a feasible solution is returned.

7.2.2 MIP Formulation

We convert bilinear constraints into mixed-integer linear constraints by gridding the solution space $\mathbb{S}$ and approximating the constraints locally inside grids with McCormick envelopes similar to [61]. A McCormick envelope relaxation of one bilinear constraint $w = xy$ [119] is the best linear approximation defined over a pair of lower and upper bounds $[x^L, x^U]$ and $[y^L, y^U]$. Therefore, we first assign grids $G(\mathbf{x})$ to $\mathbb{S}$. Let $\{g_l\}$, $l = 1, ..., L$ be one cell in the grid with upper and lower bounds $[\mathbf{x}^L, \mathbf{x}^U]$. We introduce additional integer vari-

ables $\mathbf{n}$, $n_i \in \{0,1\}$. Each unique value of $\mathbf{n}$ corresponds to one cell in the grid within which McCormick envelope relaxations are applied. The constraints $b_j(\mathbf{x},\mathbf{z};\Theta) \leq 0$, $j = 1,...,m_b$ are converted into L constraints: $E_{b_j,l}(\mathbf{x},\mathbf{z},\mathbf{n};\Theta) \leq 0$ for $j = 1,...,m_b$ and $l = 1,...,L$, turning (7.3) into an MICP:

$$\begin{aligned} & \underset{\mathbf{x},\ \mathbf{z},\ \mathbf{n}}{\text{minimize}} \ f_{obj}(\mathbf{x},\mathbf{z},\mathbf{n};\Theta) \\ \text{s. t.} \quad & f_i(\mathbf{x},\mathbf{z};\Theta) \leq 0, \ \ i = 1,...,m_f \\ & E_{b_j,l}(\mathbf{x},\mathbf{z},\mathbf{n};\Theta) \leq 0, \ \ j = 1,...,m_b, \ \ l = 1,...,L \end{aligned} \tag{7.5}$$

We use a $log_2 N$ formulation which means N grids are represented by $\lceil log_2 N \rceil$ binary variables. For example, $17 \sim 32$ grids are represented by 5 integer variables. Since the intervals generated by the clustering methods are not necessarily connected, the proposed method in [61] does not work. The formulation we use can incorporate intervals that are not connected hence more general. The price to pay is additional continuous variables. Please see Appendix for details of the formulation. Despite the relatively smaller number of binary variables due to $log_2 N$ formulation, approximating nonlinear constraints with mixed-integer convex constraints still generates a large number of integer variables when high approximation accuracy is desired. As a result, the formulation suffers from extended solving time. We use data-driven methods to reduce the number of integer variables and learn good warm-starts for online applications.

7.2.2.1 Reduced Scale MIP Formulation

[178] proposes ReDUCE as a method to generate smaller scale MIP problems. The basic idea of using unsupervised learning to reduce the size of mixed-integer programs is to identify and cluster the important regions on the solu-

tion manifold, and generate MIP formulations with smaller scale. If $\mathbb{R}^{dim(\mathbf{x})}$ is segmented into smaller regions, e.g. clusters, the required integer variables for each cluster can be reduced. This may be seen in Fig. 5.9 which is an instance of 2-dimensional (dim) $X(\Theta)$ from a bookshelf experiment described in Sec. 8.1.3.1.

This paragraph reviews the general steps of ReDUCE. To begin, ReDUCE uses pre-solved dataset $(\Theta_k, \mathbf{x}_k)$, $k = 1,...,K$. We pre-assign grids $G(\mathbf{x})$ to the solution space $\mathbb{S}$. The size of grids depends on the approximation accuracy requirement for bilinear constraints. ReDUCE begins by performing unsupervised learning on the pre-solved dataset to retrieve clusters that indicates regions on $X(\theta)$. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [179] was used to cluster on $\mathbf{x}$ which gives clusters $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_C$, where C is the number of clusters. We then trace $\Theta \rightarrow \mathbf{x}$ map backwards to create clusters in Θ space, i.e. $(\{\Theta\}_1, \{\mathbf{x}\}_1), ..., (\{\Theta\}_C, \{\mathbf{x}\}_C)$. Next, a supervised classifier is trained to classify (Θ, c). We use Random Forest which requires relatively smaller amounts of data to train than deep learning methods. For all $\{\mathbf{x}\}_c$ in cluster c, we find the grid cells that they occupy and re-assign integer variables $\mathbf{z}_c$ and $\mathbf{n}_c$ to each cluster. At this point, the original MIP problem is segmented into smaller scale MIP problems with faster solving speed. When a new problem instance comes online, the classifier is used to point the new problem to one of the existing clusters and the reduced scale MIP is solved.

The idea of solving smaller sub-MIPs bears similarities with algorithms such as RENS [167] and Neural Diving [38] where a subset of integer variables are fixed from linear program solutions or a learned model, while the others are solved. Our method, however, segments the problem through an unsupervised clustering approach.

7.2.2.2 Convex Formulation

In the extreme case, if the learner can provide a complete warm-start of the binary variables, the problem online becomes convex permitting fast QP solvers. [27, 180, 181] define integer strategies to be tuples of $\mathcal{I}(\Theta_i) = (\mathbf{z}^*, \mathbf{n}^*, \mathcal{T}(\Theta_i))$, where $(\mathbf{x}^*, \mathbf{z}^*, \mathbf{n}^*)$ is an optimizer for problem (7.5), $\mathcal{T}(\Theta_i) = \{i \in 1, ..., m_f,\ l \in 1, ..., L | f_i(\mathbf{x}^*, \mathbf{z}^*; \Theta) = 0,\ E_{b_j,l}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{n}^*; \Theta) = 0\}$ is the set of active inequality constraints. Given an optimal integer strategy $\mathcal{I}(\Theta_i)$, solutions to (7.5) can be retrieved through solving a convex optimization problem. This idea is adopted for learning warm-start with a convex formulation online. We have implemented both K-nearest neighbor to sample the points that are closest to the new problem Θ. We have also tested a simple neural-network structure identical to [27].

7.2.3 Conclusion

We have listed several online formulations for receiving a warm-start and solving the problem fast to meet the real-time planning and control requirements. In the next chapter, we implement those methods for planning item manipulation motions in a cluttered environment and controlling a quadruped robot walking with fast gait selection online.

CHAPTER 8

Data-driven Methods for Mixed-integer Non-convex Optimization: Applications

In this section, we demonstrate several implementations of the data-driven methods proposed in the previous section to solve optimization problems. Examples include the organization of stored items using a manipulator, hybrid non-convex model predictive control, etc. One common feature of those problems is that the problem size can easily be large, and they require real-time solving speed (within a few seconds for motion planning problems, and more than 50Hz for control problems).

8.1 Book Shelf Organization Problem

We introduce the bookshelf organization problem. Given a bookshelf with several books on top, an additional book needs to be placed on the shelf with minimal disturbance on the existing books. Some of the original and solved cases are given in Fig. 8.2. This problem has practical applications in e.g. logistic industry. A more comprehensive presentation of the formulation and results are available at [182]

8.1.1 Problem Formulation

Assume a 2D bookshelf with limited width W and height H contains rectangular books where book i has width W_i and height H_i for $i = 1, ..., N-1$. A new book, $i = N$, is to be inserted into the shelf. The bookshelf contains enough books in various orientations, where in order to insert book N, the other $N-1$

books may need to be moved, i.e., optimize for minimal movement of $N-1$ books. This paper focuses on inserting one book using a single robot motion during which the book is always grasped. The problem settings can be extended to a sequence of motions with re-grasping.

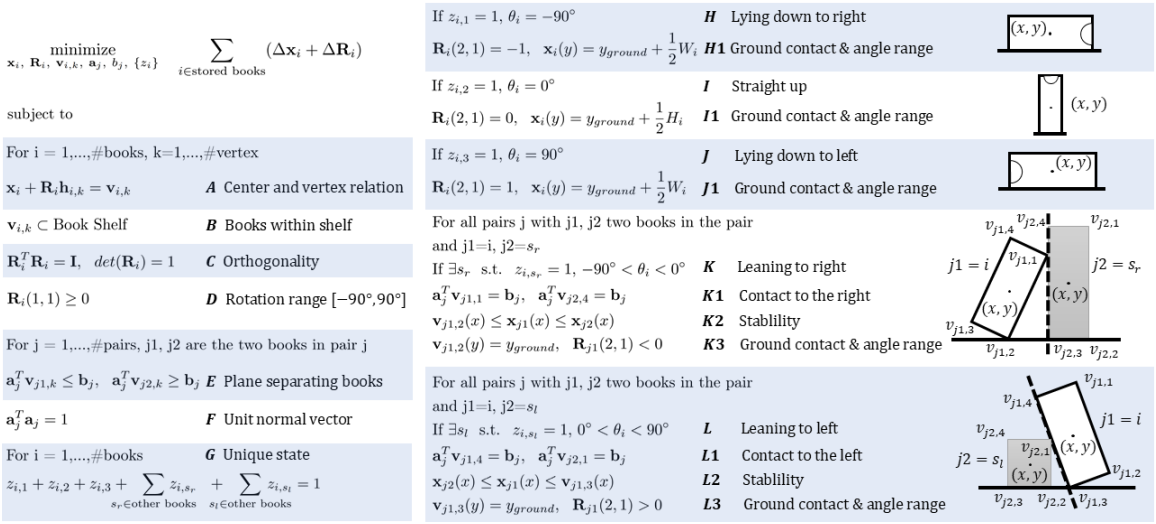


Figure 8.1: Complete formulation of the bookshelf organization problem.

The variables that characterize book i are: position $\mathbf{x}_i = [x_i, y_i]$ and angle θ_i about its centroid. $\theta_i = 0$ when a book stands upright. The rotation matrix is: $\mathbf{R}_i = [cos(\theta_i),\ -sin(\theta_i);\ sin(\theta_i),\ cos(\theta_i)]$. Let the 4 vertices of book i be $\mathbf{v}_{i,k}$, $k = 1, 2, 3, 4$. The constraint:

$$\mathbf{x}_i + \mathbf{R}_i\mathbf{h}_{i,k} = \mathbf{v}_{i,k} \tag{8.1}$$

shows the linear relationship between $\mathbf{x}_i$ and $\mathbf{v}_{i,k}$, where $\mathbf{h}_{i,k}$ is the constant offset vector from its centroid to vertices.

Constraint:

$$\mathbf{v}_{i,k} \subset \text{Book Shelf} \tag{8.2}$$

enforces that all vertices of all books stay within the bookshelf, a linear constraint.

Constraint:

$$\mathbf{R}_i^T \mathbf{R}_i = \mathbf{I}, \;\; det(\mathbf{R}_i) = 1 \tag{8.3}$$

enforces the orthogonality of the rotation matrix, a bilinear (non-convex) constraint.

Constraint:

$$\mathbf{R}_i(1,1) \geq 0 \tag{8.4}$$

enforces that the angle θ_i stays within $[-90°, 90°]$, storing books right side up.

To ensure that the final book positions and orientations do not overlap with each other, separating plane constraints are enforced. For convex shapes, the two shapes do not overlap with each other if and only if there exists a separating hyperplane $\mathbf{a}^T\mathbf{x} = b$ in between [135]. That is, for any point $\mathbf{p}_1$ inside shape 1 then $\mathbf{a}^T\mathbf{p}_1 \leq b$, and for any point $\mathbf{p}_2$ inside shape 2 then $\mathbf{a}^T\mathbf{p}_2 \geq b$. This is represented by:

$$\begin{aligned} &\mathbf{a}_j^T \mathbf{v}_{j1,k} \leq \mathbf{b}_j, \;\; \mathbf{a}_j^T \mathbf{v}_{j2,k} \geq \mathbf{b}_j \\ &\mathbf{a}_j^T \mathbf{a}_j \geq 0.5 \end{aligned} \tag{8.5}$$

Both constraints are bilinear constraints. The second constraint speaks that the norm of the plane normal vector should be larger than 0. We avoid directly using an equality constraint of unit norm vector to reduce the numerical difficulty of the solver.

Finally, we need to assign a state to each book i. For each book, it can be standing straight up, laying down on its left or right, or leaning towards left

or right against some other book, as shown in the far right column in Fig. 8.1. For each book i, we assign a set of integer variables z_i. If book i stands upright ($z_{i,2} = 1$) or lays flat on its left ($z_{i,1} = 1$) or right ($z_{i,3} = 1$), constraints:

$$\mathbf{R}_i(2,1) = -1, \;\; \mathbf{x}_i(y) = y_{ground} + \frac{1}{2} W_i \tag{8.6}$$

or

$$\mathbf{R}_i(2,1) = 0, \;\; \mathbf{x}_i(y) = y_{ground} + \frac{1}{2} H_i \tag{8.7}$$

or

$$\mathbf{R}_i(2,1) = 1, \;\; \mathbf{x}_i(y) = y_{ground} + \frac{1}{2} W_i \tag{8.8}$$

are enforced, respectively. If book i leans against another book (or left/right wall which can be treated as static books) on the left or right, constraints

$$\begin{aligned} &\mathbf{a}_j^T \mathbf{v}_{j1,1} = \mathbf{b}_j, \;\; \mathbf{a}_j^T \mathbf{v}_{j2,4} = \mathbf{b}_j \\ &\mathbf{v}_{j1,2}(x) \leq \mathbf{x}_{j1}(x) \leq \mathbf{x}_{j2}(x) \\ &\mathbf{v}_{j1,2}(y) = y_{ground}, \;\; \mathbf{R}_{j1}(2,1) < 0 \end{aligned} \tag{8.9}$$

or

$$\begin{aligned} &\mathbf{a}_j^T \mathbf{v}_{j1,4} = \mathbf{b}_j, \;\; \mathbf{a}_j^T \mathbf{v}_{j2,1} = \mathbf{b}_j \\ &\mathbf{x}_{j2}(x) \leq \mathbf{x}_{j1}(x) \leq \mathbf{v}_{j1,3}(x) \\ &\mathbf{v}_{j1,3}(y) = y_{ground}, \;\; \mathbf{R}_{j1}(2,1) > 0 \end{aligned} \tag{8.10}$$

are enforced, respectively. Fig. 8.1 shows the constraints, variables and objective function for the bookshelf problem. To this end checks need to indicate the contact between books. By looking at the right column in Fig. 8.1, we can

reasonably assume that the separating plane $\mathbf{a}^T\mathbf{x} = b$ always crosses vertex 1 of the book on the left and vertex 4 of the book on the right. This is represented by bilinear constraints, **K1** and **L1**. In addition, the books need to remain stable given gravity. Constraints **K2** and **L2** enforce that a book is stable if its x position stays between the supporting point of itself (vertex 2 if leaning rightward and vertex 3 if leaning leftward) and the x position of the book that it is leaning onto. Lastly, constraint **K3** and **L3** enforce that the books have contact with the *ground*. For practical reasons, we assume that books cannot stack onto each other, i.e, each book has to touch the *ground* of bookshelf at at least one point. We note that constraints in **H**, **I**, **J**, **K**, and **L** can be easily formulated as MICP constraints using big-M formulation [183], such that they are enforced only if the associated integer variable $z = 1$. Also, it can easily be extended to allow stacking for our problem. Any contact conditions between pairs of books may also be added into this problem as long as it can be formulated as mixed-integer convex constraint. Overall, this is a problem with integer variables, z_i, and non-convex constraints **C**, **E**, **F**, **K1**, and **L1**, hence, an MINLP problem.

Practically, this problem presents challenges for retrieving high quality solutions. If robots were used to store books, the permissible solving time is several seconds, and less optimal solutions means longer realization times. For example, in Fig. 8.2 a non-optimal insertion induces multiple additional robot motions that dramatically increase the chance of failure. There are several potential approaches to resolving this issue: fix one set of nonlinear variables and solve MICP [49], convert the nonlinear constraints into piece-wise linear constraints and formulate them into an MICP [18], or directly apply MINLP solvers such as BONMIN [184]. As expected, these approaches struggle to meet the requirements.

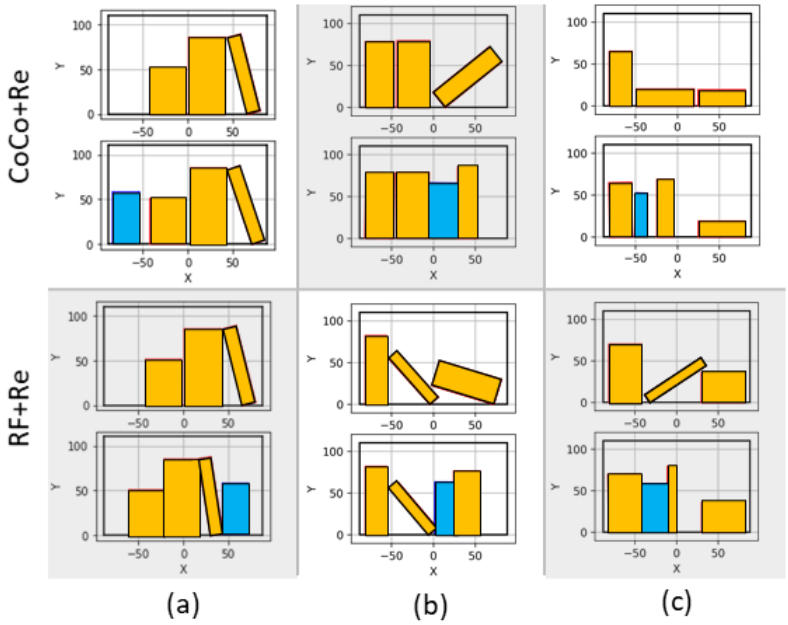


Figure 8.2: Solved scenes of bookshelves. *Top Row*: Instance solved by CoCo with ReDUCE. *Bottom Row*: Instance solved by RF with ReDUCE. Column (a), (b), (c) corresponds to cluster 0, 1, 2, respectively. Each cell contains a before and after scene with the upper diagram showing the original bookshelf with orange rectangles representing stored books and the lower diagram showing the solved bookshelf where the blue rectangle represents the inserted book. For column (a), we demonstrate a scene where RF gives a much worse result than CoCo such that multiple books are moved. However, CoCo and RF usually give similar results.

8.1.2 Mixed-integer Formulations

We explain the formulation that is used to solve the MIPs in the previous section.

For a non-convex constraint, we segment it into multiple regions and locally approximate or relax them into convex constraints. In this paper, we relax the bilinear constraints locally into convex polytopes (McCormick envelopes). Each polytope is associated with a unique combination of integer variable values, hence, mixed-integer convex constraints. Assume the number of regions used is N. Depending on the number of integer variables used, the formulation can generally be divided into 2 categories: 1) If the number of integer variables

is N, we call it *N formulation*, e.g., the convex hull formulation [185]. 2) If the number of integer variables is $log_2 N$, we call it *$log_2 N$ formulation*, e.g., [186]. [186] presents a *$log_2 N$ formulation* to model the special ordered sets of type 2 (sos2). However, there are several limitations of this formulation. For example, the segmented regions need to be connected to be a valid sos2 constraint, i.e., the two consecutive couple of non-zero entries can be any consecutive couple in the set. In this paper, we use MIP to model convex polytopes of arbitrary locations. This can increase the complexity for sos2 techniques as the polytopes can be disjunctive. The disjunctive constraints can be handled with convex hull formulations at a price of introducing more integer variables which may result in slower solving speeds.

In this appendix, we demonstrate an intuitive but general *$log_2 N$ formulation* to model combinations of convex polytopes at any locations which serves as the base MIP formulation for the bilinear constraints in our paper. Assume the variable $\mathbf{x}$ is enforced to be within one of the N convex polytopes, denoted by $\mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i$, $i = 1,...,N$. We introduce $m = log_2 N$ binary variables $z_1,...,z_m$, $z_i \in \{0,1\}$. Each combination of unique values of binary variables can be assigned to a convex polytope. Let the assignment be:

$$\mathbf{z} = \bar{\mathbf{z}}_i \Rightarrow \mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i \tag{8.11}$$

Where $\bar{\mathbf{z}}_i = [\bar{z}_{i,1},...,\bar{z}_{i,m}]$ are constant binary values associated with polytope i. Note $\bar{z}_i \neq \bar{z}_j$ if $i \neq j$. In other words, we require that when $\mathbf{z} = \bar{\mathbf{z}}_i$, $\mathbf{x}$ stays within the polytope $\mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i$; otherwise, the constraint is unenforced.

Denote the vertices of the polytopes by $\mathbf{v}_{i,1},...,\mathbf{v}_{i,n_i}$, $i = 1,...,N$, where n_i is the number of vertices associated with polytope i. Each vertex $\mathbf{v}_{i,j}$ is assigned a continuous non-negative variable $\lambda_{i,j} \in [0,1]$. In general, one can run a mathematical program (e.g. [187]) to get vertices from the nondegenerate system of

Table 8.1: Benchmark Results

	ADMM - Knitro	ADMM - IPOPT	MIP - 50	MIP - 20	MPCC_default - Knitro	MPCC_default - IPOPT	MPCC_Manual - Knitro	MPCC_Manual - IPOPT	MPCC_KNN3 - Knitro (15000data)	MPCC_KNN3 - IPOPT (15000 data)	MPCC_KNN3 (Rev) - Knitro (36000 data)	Convex - 200 (15000 data)	Convex - 600 (15000 data)	Convex - 200 (78000 data)
Note			Each cluster 50 samples	Each cluster 20 samples	Using default initial guess from CasADi		Using manually selected 1 initial guess		KNN selects top 3 guesses from 15000 data		Same to left but select the worst 3 guesses	Max iteration of OSQP=200	Max iteration of OSQP=600	Currently KNN learner
Success Rate	96.5%	94.75%	100 %	98.75%	1.25 %	0 %	78.25%	78.25%	99.25%	98.5%	92.5%	94.5%	98.75%	96%
Avg. Solve Time	260 ms	522 ms	790 ms	616 ms	72 ms		29 ms	81 ms	30 ms	96 ms	61 ms	65 ms	80 ms	16 ms
Max. Solve Time	1.29 sec	3.59 sec	17.3 s	70 s	82 ms		198 ms	1.96 sec	280 ms	990 ms	327 ms	190 ms	400 ms	174 ms
Avg. Objective	-7250	-7339	-8606	-8435	-5703		-8620	-8952	-8258	-8618	-6461	-8631	-8687	-8637
Avg. # of iterations	4.73	4.6	1	1			1	1	1.05	1.06	1.25	4.85	3.85	2.57
Solver	Gurobi+Knitro	Gurobi+IPOPT	Gurobi	Gurobi	Knitro	IPOPT	Knitro	IPOPT	Knitro	IPOPT	Knitro	OSQP	OSQP	OSQP

inequalities $\mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i$. As a result, the assignment becomes:

$$\mathbf{z} = \bar{\mathbf{z}}_i \Rightarrow \begin{array}{c} \mathbf{x} = \sum_{j=1}^{n_i} \lambda_{i,j}\mathbf{v}_{i,j} \\ \sum_{j=1}^{n_i} \lambda_{i,j} = 1, \quad \lambda_{i,j} \in [0,1] \end{array} \tag{8.12}$$

The formulation can be written as:

$$\begin{aligned} (5.a) \quad & \mathbf{x} = \sum_{i=1}^{N}\sum_{j=1}^{n_i} \lambda_{i,j}\mathbf{v}_{i,j} \\ (5.b) \quad & \sum_{i=1}^{N}\sum_{j=1}^{n_i} \lambda_{i,j} = 1, \quad \lambda_{i,j} \in [0,1] \\ (5.c) \quad & \sum_{k=1,\ldots,N}^{k\neq i}\sum_{j=1}^{n_k} \lambda_{k,j} \leq \sum_{l=1}^{m} |z_l - \bar{z}_{i,l}| \end{aligned} \tag{8.13}$$

The set of constraint in (8.13) enforces (8.12) for all polytopes, as $\sum_{l=1}^{m} |z_l - \bar{z}_{i,l}| = 0$ only when $\mathbf{z} = \bar{\mathbf{z}}_i$, enforcing that all λ's that are not associated with polytope i to be zero. If $\mathbf{z} \neq \bar{\mathbf{z}}_i$, $\sum_{l=1}^{m} |z_l - \bar{z}_{i,l}| \geq 1$ and constraint (5.c) is looser than constraint (5.b), hence, trivial.

Note that formulation (8.13) works for any convex polytope that can be written as $\mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i$. In this paper, the polytopes are McCormick envelope constraints which is a special case.

8.1.3 Solving with Data-driven Methods

8.1.3.1 Setup

We place 3 books inside the shelf where 1 additional book is to be inserted. Grids are assigned to the variables involved in the non-convex constraint **C**, **E**, **F**, **K1**, **L1**: $\mathbf{R}_i(\theta_i)$, $\mathbf{a}_j$ and $\mathbf{v}_{i,k}$. These variables span a 48-dim space. The rotation angles θ_i, which includes $\mathbf{R}_i$, are gridded at $\frac{\pi}{8}$ intervals. Elements in $\mathbf{a}_j$ are gridded on 0.25 intervals. Elements in $\mathbf{v}_{i,k}$ are gridded at intervals $\frac{1}{4}$ the shelf width W and height H. Table 8.3 lists the number of grids for each non-convex variable. Our MICP formulation results in 130 integer variables in total. The input vector includes the center positions, angles, heights and widths of stored books and height and width of the book to be inserted. The input dimension is 17.

The problem instances are generated using a 2-dim simulated environment of books on a shelf. Initially, 4 randomly sized books are arbitrarily placed on the shelf, and then 1 is randomly removed and regarded as the book to be inserted. Contrary to the sequence, the initial state with 4 books represents one feasible (not necessarily optimal) solution to the problem of placing a book on a shelf with 3 existing books. This guarantees that all problem instances are feasible. Since this problem can be viewed as high-level planning for robotic systems, the simulated data is sufficient. For applications outside of the scope of this paper real-world data may be preferable in this pipeline.

All methods are tested on 400 randomly sampled bookshelf problems in the same distribution using the same method mentioned above. All results are listed in table 8.1.

8.1.3.2 Complementary Formulation

We turn the binary variables into continuous variables with complementary constraints. We selected both IPOPT and Knitro solvers to solve the MPCC formulation. A small relaxation of $\epsilon = 10^{-3}$ on the right hand side of the complementary constraints is used to increase the chance of getting a feasible solution.

Zero Warm Start As a baseline, we solve the problem with the default initial guess of the solvers (all zeros). The results are shown in the column of Table 8.1 named "MPCC_default". As we can see, despite the relaxation of complementary constraints, the rate of solving the problem successfully with zero warm-start is almost zero.

Warm Start with Manually Designed Heuristics Another baseline is to warm-start the formulation with human-designed heuristics. The bookshelf problem requires inserting one book minimizing the movements of the existing books. Therefore, a good initial guess is to directly use the continuous and discrete variables from the original scene. We also pre-solve the separating planes using the original scene to warm-start the parameters $\mathbf{a}$ and b. The authors believe this is the best initial guess readily available without paying much more effort. The results are shown in Table 8.1 in the column "MPCC_Manual".

Implementing this initial guess increases the success rate to more than 70%, significantly better than the default initial guess from the solver. However, we noticed that this success rate drops as the number of books increases. When there are only 2 books on the shelf, inserting the 3rd book has over 80% success rate with the same approach for the initial guess. Therefore, the manually selected initial guess may not scale to problems with much larger sizes.

Warm Start with pre-solved Dataset We show the result of using K-nearest neighbor learner to pick out the top 3 candidates from a 15000 pre-solved dataset in the column "MPCC_KNN3" column in Table 8.1. The results show

that a simple KNN solver is able to significantly increase the rate of getting a feasible solution to more than 98%. The average iteration of solving the problem is only slightly more than 1 showing that most problem instances are solved with only 1 K-nearest neighbor sample. Since the MPCC formulation keeps the non-convexity of the original formulation, it permits the solver to explore larger regions, hence the learner does not need to pick out a point or region that is very close to the optimal solution as required by the convex formulation. Note that we found warm-starting the non-convex continuous variables is as important as warm-starting the discrete (complementary) variables.

8.1.3.3 Reduced Order Mixed-integer Programming Formulation

We randomly sample 4,000 bookshelves and implement DBSCAN to realize 100 clusters. DBSCAN was tuned to ensure the number of clusters did not grow with samples while maintaining a minimal amount of outliers. Fig. 5.9 shows the first 2 dimensions of the projected solution set $X(\Theta)$ and 6 modes with distinct labels and colors. Fig. 8.3 shows the solution set packed into tighter groups compared to the same sample of features using t-distributed Stochastic Neighbor Embedding (t-SNE) [188] as a projection of the high dimensional space to 2-dim. The upper graph in Fig. 8.3 shows the clusters in the solution space, while the lower one depicts the corresponding clusters in the feature space according to $(\Theta, \mathbf{x})$ from the *kick off* data. The colors in Fig. 8.3 denote the different clusters. We can tell obvious separations in the solution space. Although the clusters are more intertwined in the feature space because of the complexity of $\Theta \longrightarrow \mathbf{x}$ mapping, there are still distinct regions where certain colors are more dominant. We trained a random forest (RF) classifier on the features Θ. The classification accuracy reaches 97% indicating that $\text{RF}(\{\Theta\}) \longrightarrow \{\mathbf{x}\}$ is able to achieve a reasonable mapping.

With the RF classifier trained on 100 clusters, we can quickly sample differ-

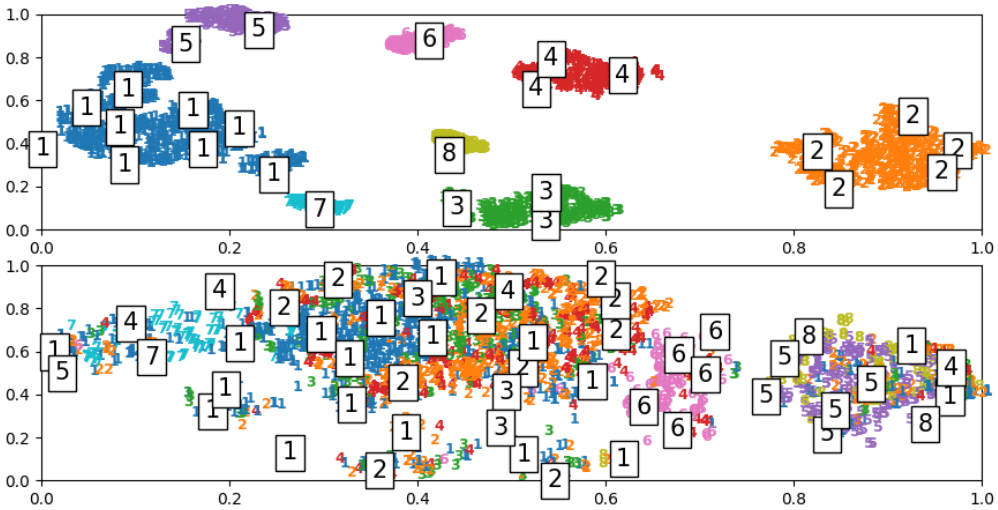


Figure 8.3: *Top*: projection onto a 2-dim manifold using t-SNE to depict clustering of a 48-dim solution space. Each color and label specify a solution clustering. Under this projection the solution appears very much structured. *Bottom*: projection of the 17-dim feature space corresponding to the solution using t-SNE. The clusters have some structure over the feature space with certain labels only being found in certain regions. For visualization purposes only the top 8 clusters out of 100 are being displayed.

ent bookshelf problems and solve the reduced MIP problem within seconds to collect more data for supervised learning. To verify the benefits with ReDUCE, we first run an experiment that observes an increase in performance with increasing amounts of data over several clusters each containing different number of integer variables (denoted as Int in Fig. 8.4). We pick 4 different clusters with number of integer variables 77, 77, 66, and 46, respectively. Five different methods to solve the reduced problem within a cluster are used and tested on a fixed testing set of 500 data points. The results are shown in Fig. 8.4. The column titled ReDUCE shows the total number of unique integer strategies (N) over training data (total), and the uniqueness percentage (%) is the resulting quotient. For the column titled CoCo+Re, we train a neural network with one hidden layer of size 10,000. The input size is equal to dimension of feature space (17) and the output layer is equal to the number of unique strategies within training data (N). The neural network then samples 30 candidate

strategies online according to the rank of the softmax scores of the network and solves the associated convex optimization problems. If one strategy gives a feasible solution, we terminate the process and record the solving time and optimal cost. Otherwise, infeasibility is recorded. Since the maximum number of convex problems considered is 30 to be solved online, the problem setup time is non-negligible. To avoid additional overhead we only setup the problem once and keep on modifying the integer constraints for more instances. Thus, all solving times include the problem setup time. The solving process is done on a Core i7-7800X 3.5GHz $\times$ 12 machine with Gurobi. For column RF+Re, we instead train a random forest with 150 decision trees and get the top 30 most voted strategies for candidate solutions. For column titled KNN+Re, we fit a K-nearest neighbor model to the feature-strategy mapping which provides the top 30 strategies whose features are closest to the feature of the problem to be solved online. For column titled Baseline+Re, we simply randomly sample 30 unique strategies from the training strategies. For column MIP+Re, we solve the MIP with reduced number of integer variables, instead of using supervised learning or sampling method. For all sampling methods, we record the rate of feasibility (S%), the deterioration (Det) of optimal cost compared to the optimal cost from MIP+Re column. That is, for the MIP+Re column, its feasibility rates are all 100%, and its optimal deteriorations are 0. Solving times for baseline is omitted as they are significantly longer than learning based methods. All times under Avg and Max (average and maximum solving time) are in seconds.

Generally, learning based methods improved with more data. As ReDUCE improves the solving speed to collect larger amounts of data, we can further increase the performances with more data. The random sampling baseline (Baseline+Re) has significantly worse performance than learning methods. This column ensures that the clusters are not too small making the reduced problem too simple. For larger clusters with more integer variables (e.g. cluster 0),

	ReDUCE			CoCo+Re				KNN+Re				RF+Re				Baseline+Re		MIP+Re	
Cluster	N/total	%	Int	S%	Det	Avg	Max	S%	Det	Avg	Max	S%	Det	Avg	Max	S%	Det	Avg	Max
0	997/1000	99.7%	77	93.4%	210	1.39	7.39	92.6%	188	1.32	7.40	93.2%	348	1.42	7.58	65.4%	856	4.52	37.86
	2984/2999	99.4%	77	97.0%	144	1.25	7.35	94.8%	122	1.09	7.16	97.0%	253	1.31	7.46	63.7%	862	4.52	37.86
	4947/5000	98.9%	77	98.0%	119	1.05	7.23	97.8%	96	0.95	7.39	96.0%	194	1.17	6.75	65.5%	933	4.52	37.86
1	698/699	99.9%	77	91.2%	148	1.42	7.37	92.2%	124	1.39	7.78	90.0%	267	1.35	7.10	57.4%	608	2.27	16.61
	1996/1999	99.8%	77	96.2%	93	1.30	7.58	96.2%	72	1.07	7.71	92.6%	204	1.27	7.21	60.0%	554	2.27	16.61
	5976/5999	99.6%	77	99.0%	73	1.24	7.12	98.4%	55	1.00	7.69	96.8%	169	1.12	7.57	60.5%	543	2.27	16.61
2	599/600	99.8%	66	91.0%	95	1.36	6.70	91.4%	62	1.10	7.51	90.6%	122	1.33	7.51	59.5%	281	0.92	5.46
	2970/2999	99.0%	66	96.8%	60	1.13	7.37	97.0%	48	0.94	7.07	93.8%	88	1.00	7.42	57.0%	293	0.92	5.46
	5476/5599	97.8%	66	98.4%	47	1.07	7.47	98.8%	37	0.89	7.52	95.2%	84	1.04	6.97	57.7%	251	0.92	5.46
3	339/399	85.0%	46	94.0%	28	1.18	7.03	94.8%	29	0.86	6.86	92.6%	37	0.94	7.04	63.2%	162	0.21	0.78
	681/900	75.6%	46	97.2%	20	0.95	6.92	97.8%	26	0.84	7.08	97.8%	29	0.73	6.94	57.5%	132	0.21	0.78

Figure 8.4: Comparison of different algorithms with ReDUCE (+Re) for CoCo (CoCo+Re), Random Forest (RF+Re), random sampling (Baseline+Re), and MIP (MIP+Re). Success rate and deterioration on the objective is denoted as S% and Det, respectively. Average solving time and maximum solving time are denoted as Avg and Max, respectively. Int stands for the number of integer variables within a cluster. With more data the algorithms typically improved.

the learning methods demonstrate an increase in solving speed over MIP. For smaller clusters (e.g. cluster 2), MIP has faster solving speeds. Therefore, if the clustering algorithm can decompose the problem into uniformly small clusters, directly solving the classified mixed-integer programming problem online may be feasible. A surprising result, as opposed to what [27] shows, is that the K-NN demonstrates good performance matching what the neural-network model can achieve. The unique aspect of the bookshelf organization problem is that many integer variables come from the envelope relaxation of continuous nonlinear variables. This may cause the mapping function from feature to the integer strategy to have good continuity favored by the KNN model.

	Success (%)	Det	Avg Time	Max Time
CoCo+Re	99.2%	140	1.21 sec	13.47 sec
MIP+Re	100 %	0	2.36 sec	48 sec
MIP	n/a	n/a	> 851 sec	n/a
MINLP	n/a	n/a	> 10 min	n/a

Table 8.2: Comparison of different solving techniques for the bookshelf problem. CoCo+Re (+Re denotes using ReDUCE) returned the top 50 candidate solutions over 100 clusters. MIP+Re used those same clusters to solve the problem on Gurobi. MIP also used Gurobi. MINLP ran using BONMIN as the solver. MIP and MINLP solving time for the full set of samples have exceeded reasonable limitations beyond practical purposes.

For a second set of experiments, we collect data for all 100 clusters, in total 17,000, and train a neural network with the same size as above. The network is then tested on a 1,000 testing set with 100 clusters blended together. The network then samples 50 strategies. The percentage of feasibility (Success (%)), the optimal cost deterioration (Det), the average (Avg Time) and max (Max Time) solving time are shown in Table 8.2. For comparison, we record the average and maximal solving time for MIP with ReDUCE (all clusters blended together). We also record the solving time for MIP without ReDUCE, the original formulation, and the solving time for formulation (7.3) solved with BONMIN, an MINLP solver. CoCo solves faster than MIP for larger clusters which is associated with more data from the *kick off* data. When sampling $D(\Theta)$, we get more samples for larger clusters. Therefore, solving time is improved when averaged. For non-reduced MIP, Table 8.2 shows the average solving time over 10 samples where the solving process is interrupted if it exceeds 1,000 sec. Thus, the average solving time is at least 851 sec. It is clear that without ReDUCE, it is intractable to gather the amount of data required to train a learner of decent performance. Similar speed is seen for the MINLP solver. All the code is available at: `https://github.com/RoMeLaUCLA/ReDUCE.git`.

8.1.3.4 Convex Formulation

To make the online formulation convex, the non-convex part of the continuous solutions is first discretized, then the complete list of integer variables is used for training. We discretize the variables using the range given by Table 8.3. A K-nearest neighbor learner is trained to pick out the first 10 integer solutions. They are used to turn the mixed-integer program into convex programs which are then solved one by one until a feasible solution is retrieved. We select the solver OSQP to solve those convex problems. This solver automatically uses the previous solution to warm-start the next solution for faster solving speed.

Table 8.3: Segmentations of Non-convex Variables

variable	range	# of segmentations
Item orientation θ (rad)	[-pi/2, pi/2]	8
Separating plane normal a	[-1, 1]	8
Vertex x position v_x (cm)	[-9, 9]	4
Vertex y position v_y (cm)	[0, 11]	4

The maximal iterations can be tuned. As most of the problems take no more than 200 iterations to solve, we set this parameter to 200. Setting the maximal iteration to 600 will incorporate more solutions, but the average solving time is longer as it takes more time before the solver decides infeasibility. The result is shown in the columns named "Convex" in Table 8.1.

The result shows that the convex formulation achieves the fastest solving speed due to the strong performance of convex solvers. The authors suspect that the solvers such as OSQP utilized the heuristics specific to convexity, hence run faster than well-initialized NLP solvers such as Knitro. On the other hand, the price to pay for turning the problem into convex is that the learner needs to perform better than in the MPCC case. This is indicated by the 4.85 average iterations required to find a feasible solution. Due to the inherent issue with KNN, some instances are classified incorrectly, and the solver wastes much longer time going over the infeasible integer variables.

As the space is discretized, the Θ space is no longer Euclidean. The quantities that are close to each other in the Euclidean distance sense, but on different sides of the discretization boundary will be assigned different integer variables, hence no longer close to each other. The KNN utilizes Euclidean distance, thus ignoring this non-Euclidean effect and failing to select the correct warm-start integers in those cases.

8.1.3.5 Nonlinear ADMM Formulation

As a non-data-driven benchmark, we implemented the nonlinear ADMM method on the MINLP formulation by iterating between a mixed-integer formulation and a nonlinear formulation as described in Section 6.4. To keep this method non-data-driven, manually designed heuristics same as 8.1.3.2 are used to warm-start the nonlinear formulation to increase the feasibility rate. A sufficient effort is devoted to tuning the weights $\mathbf{G}$, γ, and scaling the variables properly to ensure the performance is on the better end. Since the mixed-integer approximations on the bilinear constraints induce numerical errors, termination conditions and accuracy of bilinear collision avoidance constraints are set lower to match the accuracy of mixed-integer envelope formulation. The results are shown in the column named "ADMM" in Table 8.1.

According to the results, ADMM works fairly well for getting a feasible solution. The solving speed is several times slower compared to the data-driven MPCC formulation and convex formulation. One important reason is that it usually takes 4-5 iterations (1 iteration includes 1 MIP and 1 NLP) to get a feasible solution. The nonlinear formulation takes more than 70% of the solving time, due to the separating plane constraints. If data-driven methods can be used to warm-start the nonlinear formulation, a faster solving speed may be achieved. In addition, the average optimal cost from ADMM is worse than most of the data-driven methods. The authors suspect that the reason is due to the extra consensus terms in the objective function which guides the convergence.

8.1.3.6 Hardware Experiment

We implement our optimization results on hardware with a 6 degree of freedom manipulator [189] to insert a book onto a shelf. To automate the system, a trajectory planner is required to generate the insertion motion, which is com-

mon in practice. As the focus of this paper is on high level planning, we simplify this part by manually selecting waypoints. The hardware implementation is shown in the attached video.

8.1.3.7 Discussion

According to our test, data-driven methods can simultaneously achieve good optimality, fast solving speed, and high success rate leveraging on a reasonable amount of pre-solved data. Among the online formulations, MPCC keeps the original non-convex formulations hence can converge better even from a relatively bad initial warm-start. The implementation of MPCC on NLP solvers is easier than the convex formulation which requires converting the non-convex constraints into mixed-integer envelope constraints. The solving speed of MPCC formulation is also fast. The convex formulation has the fastest solving speed due to well-performed convex solvers. However, since additional constraints are added to make the problem convex, it is more difficult for the learner to select a good warm-start, resulting in more trials. Finally, non-data-driven methods such as nonlinear ADMM have a decent chance to get a feasible solution. However, both the optimality and the solving speed are much worse.

The obvious merit of the data-driven approach is that it can solve the problem fast, but also optimally and reliably as long as the new problem instance is within the trained parameter regime. An obvious challenge with this method is how to generalize the training to cases outside the dataset. To achieve this goal, simple learners such as KNN do not perform well, and more modern learning methods such as auto-encoders are to be explored.

8.2 Mixed-integer Non-Convex Model Predictive Control

In this section, a model predictive controller based on data-driven methods to solve mixed-integer non-convex programming is demonstrated. This shows that the data-driven methods can significantly speed up the otherwise difficult optimization problem such that it can be used for high-speed control applications. The materials are from [190].

8.2.1 Dynamic Model

In this section, we explain the dynamic model used for control in this paper. Following the common approach seen in literature, we model the robot into a single rigid body ignoring the swing leg dynamics. However, we do not make any simplification of the model such as the small angle approximation which is commonly seen in the literature [67]. We also do not use a pre-planned gait.

8.2.1.1 Nonlinear Dynamics

The dynamics of a single rigid body including the Newton second law and the angular momentum are:

$$m\ddot{\mathbf{p}} = \sum_{i=1}^{N} \mathbf{f}_i - m\mathbf{g} \tag{8.14}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) = \sum_{i=1}^{n} \mathbf{r}_{b,i} \times \mathbf{f}_i \tag{8.15}$$

Where $\mathbf{p}$ represents the geometric center of the body which is also assumed to be the center of mass. $\mathbf{f}_i$ represents the contact force on toe i. $\boldsymbol{\omega}$ is the angular velocity and $\mathbf{I}$ is the moment of inertia. $\mathbf{r}_{b,i}$ is the vector from body center of mass position $\mathbf{p}$ to the contact position $\mathbf{p}_i$ which is the moment arm of $\mathbf{f}_i$.

We define the Z-Y-X Euler angles as $\mathbf{\Theta} = [\phi, \theta, \psi]$. The transformation matrix from body frame to world frame is:

$$\mathbf{R}_{wb} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - s\psi c\phi & s\phi s\psi + s\theta c\phi c\psi \\ s\psi c\theta & c\phi c\psi + s\phi s\theta s\psi & s\theta s\psi c\phi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{8.16}$$

The transformation from the rate of change of Euler angles to angular velocities $\boldsymbol{\omega}$ is:

$$\boldsymbol{\omega} = \begin{bmatrix} c\theta c\psi & -s\psi & 0 \\ c\theta s\psi & c\psi & 0 \\ -s\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{8.17}$$

In this paper, we assume that the robot is making point contact to the ground. The friction cone constraint on each toe can be written as:

$$f_{i,z} \geq \mu \sqrt{f_{i,x}^2 + f_{i,y}^2} \tag{8.18}$$

Where μ is the coefficient of friction. Following the common approaches of trajectory optimization, the continuous trajectory is discretized into knot points at step $1, 2, ..., N$ with fixed ΔT. Between knot points, the forward Euler integration constraint is enforced:

$$\begin{aligned} \mathbf{p}[n+1] - \mathbf{p}[n] &= \mathbf{v}\Delta T \\ \mathbf{v}[n+1] - \mathbf{v}[n] &= \mathbf{a}\Delta T \\ \mathbf{\Theta}[n+1] - \mathbf{\Theta}[n] &= \mathbf{\Theta}\Delta T \\ \boldsymbol{\omega}[n+1] - \boldsymbol{\omega}[n] &= \boldsymbol{\alpha}\Delta T \end{aligned} \tag{8.19}$$

Where $\mathbf{v}$ is the center of mass velocity and $\mathbf{a}$ is the center of mass acceleration. $\boldsymbol{\alpha}$ is the body angular acceleration.

The relation between the vector of toe position $\mathbf{p}_{w,i}$ originated from world frame origin to the vector of toe position $\mathbf{p}_{b,i}$ originated from the body center of mass is:

$$\mathbf{p}_{w,i}[n+1] - \mathbf{p}[n] = \mathbf{p}_{b,i}[n+1] \tag{8.20}$$

For easiness of solving, we also make approximation of the leg workspace into a box:

$$\begin{aligned} \mathbf{p}[n] + \mathbf{R}_{wb}[n]\mathbf{H}_{b,i} &= \mathbf{H}_{w,i}[n] \\ \mathbf{p}_{w,i}[n] - \mathbf{H}_{w,i}[n] - \mathbf{R}_{wb}[n]o_i &\in \mathbf{R}_{wb}[n]B \end{aligned} \tag{8.21}$$

Where o_i is the offset from body center to the shoulder. $\mathbf{H}_{b,i}$ is the vector from the robot center of mass to the shoulder position (base position of leg) of the robot body. $\mathbf{H}_{w,i}$ is the vector from the origin of the world frame to the shoulder position. B is the size of box for leg workspace approximation. Note that since we collect data offline, we can use full kinematics to completely explore the workspace. Online, we can discretize the workspace into convex regions and formulate the kinematics as mixed-integer convex constraint [7].

In order to characterize gait, we define contact variable c_i for each toe. If $c_i = 0$, the leg i is lifted to the air. If $c_i = 1$, the leg is down on the ground. Since each toe can only have one state, c_i is a binary variable.

In addition, the no-slip condition is enforced:

$$|\mathbf{p}_{w,i}[n+1] - \mathbf{p}_{w,i}[n]| \leq (1 - c_i[n])M + (1 - c_i[n+1])M \tag{8.22}$$

Where M is the standard bigM constant. This mixed-integer linear constraint says that if the toe i is on the ground at both iteration n and $n+1$, it should not move (slip).

When the toe is lifted into the air, we enforce the lift height constraint and the zero force constraint:

$$\begin{aligned} p_{b,i}(z) &= H \\ |\mathbf{f}_i| &\leq \mathbf{f}_{max} c_i \end{aligned} \tag{8.23}$$

The lift height H is pre-defined. It can be a variable to deal with situations such as climbing on stairs with variable height. This will be explored in later papers.

When $c_i = 1$, the toe is making contact with the ground. For pre-solved trajectories, we generate randomized terrain shapes discretized into multiple convex polygon regions $s = 1, ..., S$, and ensure that the robot makes contact with one of the polygon region. This constraint is again mixed-integer convex. We define binary variables $z_{i,s}$ for toe i and convex region s such that if $z_{i,s} = 1$, toe i makes contact with the convex region s. The constraint is:

$$\mathbf{p}_{w,i} \in \text{Region}_s \text{ if } z_{i,s} = 1 \tag{8.24}$$

Since the toe can be lifted or make contact with one of the convex region, we enforce:

$$\sum_s z_{i,s} + (1 - c_i) = 1 \tag{8.25}$$

8.2.1.2 Envelope Approximation

A standard approach to formulate the linear relaxations for bilinear constraints is through McCormick envelopes [62]. For a bilinear constraint $z = xy$, where $x \in [x^L, x^U]$ and $y \in [y^L, y^U]$, the McCormick envelope relaxation can be defined as:

$$
\begin{aligned}
&z \geq x^L y + x y^L - x^L y^L,\ z \geq x^U y + x y^U - x^U y^U \\
&z \leq x^U y + x y^L - x^U y^L,\ z \leq x^L y + x y^U - x^L y^U
\end{aligned}
\tag{8.26}
$$

Which is the best set of linear relaxation for variable range $[x^L, x^U]$ and $[y^L, y^U]$. Multi-linear terms such as $a_1 a_2 ... a_n$ usually exist in the equation of dynamics. For trilinear terms, although relaxations of higher accuracy exist [162], we simply use two McCormick envelopes. Specifically, for terms such as $a = a_1 a_2 a_3$, we first define $a_{12} = a_1 a_2$ and implement Eqn. (8.26) between $[a_1, a_2, a_{12}]$, then implement Eqn. (8.26) again between $[a_{12}, a_3, a]$. We found reasonable approximation accuracy out of this approach.

Usually, this approximation accuracy is insufficient if we use the nominal variable range to generate a single McCormick envelope. The standard approach is to separate the variable range into smaller regions and implement multiple envelopes with integer variables pointing to a specific region [61]. For our set of single rigid body dynamics, we identify non-convex multi-linear terms to be the $\mathbf{w} \times (\mathbf{I}\mathbf{w})$ which is bilinear, $\mathbf{r}_{b,i} \times \mathbf{f}_i$ which is bilinear, terms in $\mathbf{R}_{wb}$ which are either bilinear or trilinear, and terms from Eqn. (8.17) which are either bilinear or trilinear. We separate the variables into the regions as shown in Table 8.4 for envelope relaxation:

Table 8.4: Segmentations of the nonconvex variables

variable	range	# of regions
angles [ϕ, θ, ψ] (rad)	[-pi/2, pi/2]	4
bilinear trig terms e.g. $c\theta c\psi$	[-1, 1]	4
rate of change for Euler angles e.g. $\dot{\theta}$ (rad/s)	[-10, 10]	16
angular velocity $\mathbf{w}$ (rad/s)	[-10, 10]	16
toe position $\mathbf{p}$ (cm)	[-8, 8]	4
contact force $\mathbf{f}$ (N)	[-15, 15]	16

For trigonometry terms such as $s\theta$ and $c\theta$, we implement standard piecewise linear relaxations similar to [5].

8.2.2 Control Implementation

One key feature of the proposed controller is that it outputs footstep position and contact force simultaneously, as opposed to the standard force MPC approaches which typically decouple the footstep planning from the force control [67]. One advantage of this is that the footstep planner has complete information about the force and hence can make contact decisions based on the dynamic property of the single rigid body model. We implement this controller on a quadruped robot designed and built by ourselves. The quadruped robot, named Spine Enhanced Climbing Autonomous Legged Exploration Robot (SCALER) [104], is a 18 DoF (12 joints + 6 rigid body DoF) position controlled robot for walking and wall-climbing. Each joint is actuated by a pair of dynamixel XM-430 servo motors which are designed for position control. To do force control, force-torque sensors are equipped on each end-effector. If the robot is given position and force commands simultaneously, it can track one of them perfectly, or stays somewhere in between. This controller is implemented by admittance control. Admittance controller measures external force and uses position control to track position and force profiles. Admittance controller can be formulated in task space as:

$$M_d\ddot{\mathbf{x}} + D_d\dot{\mathbf{x}} + K_d(\mathbf{x} - \mathbf{x}_0) = K_f(\mathbf{f}_{meas} - \mathbf{F}_{ref}) \tag{8.27}$$

Where M_d, D_d and K_d are desired mass, damping and spring coefficients. $\mathbf{F}_{ref}$ is the wrench profile to be tracked. This equation can be re-written for controller design:

$$\ddot{\mathbf{x}} = M_d^{-1}(-D_d\dot{\mathbf{x}} - K_d(\mathbf{x} - \mathbf{x}_0) + K_f(\mathbf{f}_{meas} - \mathbf{F}_{ref})) \tag{8.28}$$

For admittance control, the control input is position u. We can use:

$$u = \iint \ddot{\mathbf{x}}\, dtdt \tag{8.29}$$

Equation (8.28) (8.29) turns the force controller into a position controller that can be directly implemented with our dynamixel servo motors.

Admittance control makes a trade-off between tracking force profile and position profile. If we set $\mathbf{D}_d = \mathbf{K}_d = \mathbf{0}$, the position tracking is turned off and the controller becomes a force tracker. If we set $\mathbf{K}_f = \mathbf{0}$ the controller becomes a pure position tracker ignoring all measured forces. Since our trajectory contains both position and force, we feed two profiles simultaneously to the admittance controller. It can find a balance between position and force tracking.

8.2.3 Learning for Warm Start

8.2.3.1 Data-driven methods for MINLPs

Since the problem incorporating online gait selection and non-convex dynamics is a mixed-integer nonlinear (non-convex) problem (MINLP) which is known to be difficult to solve, we implement data-driven methods to learn warm-start out of pre-solved instances and solve a simplified problem online. This idea was tested in [34]. We first list some of the results. There are in general two approaches to convert the MINLP into either mixed-integer programs with envelopes, or nonlinear programs with complementary constraints. The first approach, as described in section 8.2.1.2, use local convex approximations to convert the non-convex constraints into mixed-integer convex constraints. The issue with this method is that the resultant mixed-integer convex constraints

are usually very slow to resolve as a precise approximation usually gives a large number of integer variables. On the other hand, one can convert the discrete variables into continuous ones with complementary constraints. Specifically, the binary variable $z \in \{0,1\}$ may be converted into $z \in [0,1]$ with additional constraint $z(1-z) = 0$. However, this conversion tends to give a numerically difficult nonlinear optimization problem. Simply giving such a problem to NLP solvers without a good initial guess results in an extremely low feasibility rate. In both cases, pre-solving some of the problems and using learning to provide warm-starts online dramatically helps with the feasibility rate and solving speed. In the extreme case, one can take the MIP approach and learn full integer variables offline, such that the problem becomes convex online given warm-starts. Commonly, the learner will sample several integer strategies and try them one by one until a feasible solution is retrieved. Ideally, we would like to use a minimal amount of trials to get a highly optimized solution resulting in the fastest solving speed (similar to convex MPCs). This will require a training dataset of high optimality and well-performed learning. Several learning schemes have been explored [27, 28].

Other than data-driven methods, several non-data-driven methods also exist to get local optimal solutions at a decent speed such as ADMM [132]. However, ADMM can oftentimes provide highly sub-optimal solutions. With datasets of good quality, data-driven methods could provide solutions that are close to global optimal ones with even faster solving speeds. In some cases of robotics problems, global optimal are sometimes desired as they have consistent behavior across similar scenarios and are economically beneficial (for example, they complete the tasks faster and cost less energy).

8.2.3.2 Training for locomotion with single rigid body model

To train a learner offline, we sample the problem instances. Define the problem with a parameter P. Consider a quadruped robot walking subject to disturbance forces. In this case, P describes the various initial conditions such as initial accelerations, velocities, angles, and angular rates caused by disturbances. In this paper, we simplify the problem by only considering flat terrain. Non-flat terrains can be considered by randomizing the terrain shape. The problem is then formulated as NLP with complementary constraints for gait variables and solved using solver IPOPT. As mentioned in the previous section, it is difficult to find feasible solutions without a good warm-start. For our problem, we use trot gait to warm-stat the binary gait variables which will be further optimized by the MIP formulation. After collecting a dataset from the NLP formulation, we use them to warm-start the equivalent MIP formulation by converting the non-convex continuous variables into integer variables. Without a warm-start, the MIP solver struggles to find any feasible solution. The MIP formulation can further improve the optimality of the dataset. One of the tricks is that we can *partially* warm-start the binary variables and let the MIP solver deal with the rest of the binary variables. As the initial conditions of the problem to be solved by MIP do not necessarily match with the dataset coming from NLP solutions, using the exact binary solutions from NLP dataset may lead to infeasibility. Partially warm-start most of the binary variables can give more feasible solutions, and the MIP solving time is still fast. For the walking problem, if we warm-start the binary variables for non-convex variables and let the MIP solver deal with gait variables, the problem can be solved in a few seconds with a much more dynamic gait retunred.

One feature of MIP is that the theoretical lower bound of the optimal cost can be retrieved by relaxing the binary variables into continuous variables. The optimal primal-dual gap can be defined by the ratio of primal objective bound

z_P minus the dual objective bound z_D, or the current best objective value minus the lower bound of objective value. The gap ratio $|z_P - z_D|/|z_P|$ tells how optimal the current solution is. The lower this value is, the closer the current solution is to the global optimal solution. Most of the solutions from the NLP solver have an optimal gap more than 30%. Since the single rigid body model is at a reasonable scale, the MIP solver Gurobi can improve the gap to less than 15%.

We then train a learner to provide full list of integer variables online. The general balance controller should not need the information about the terrain, and should regulate the robot body despite the change of environment. Therefore, the input of the learner is the body center of mass positions, velocities and accelerations, and the output is foot positions, body orientations and contact forces. On the other hand, the controller can be aware of the terrain if vision information is available. In this case, constraint (8.24) can be replaced by terrain observed by the robot. In this case, vision information can also be used as input to the learner which provides the warm-start allowing the MPC to make decisions of the foot steps on the terrain through convex optimization.

8.2.4 Experimental Results

We conduct several offline computation experiments using the method described in the paper. For getting an offline solution to a certain problem, we first sample the problem parameters around the targeted problem, then solve the set of problems using the nonlinear programming method with a trot gait as initial guesses. Although the chance of getting infeasible solution is high due to complementary constraints, NLPs usually finish within a few seconds. We can keep sampling until we find feasible cases. After gathering some feasible solutions using NLP, we use them to warm-start the target problem with the mixed-integer formulation. This greatly improves the solving speed of the target MIP which can oftentimes be solved to a MIP gap below 15% within a

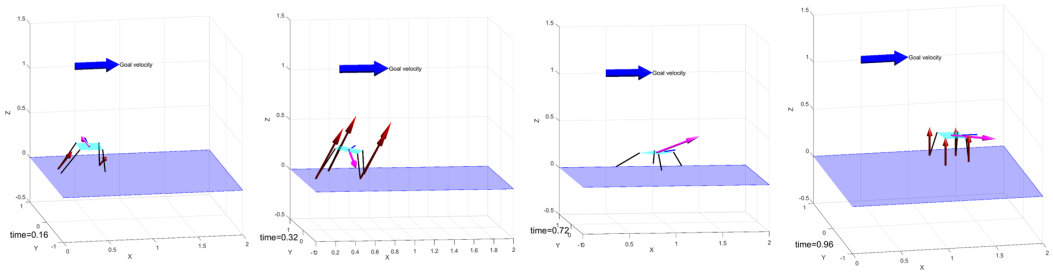


Figure 8.5: Large angle turning trajectory and contact forces. From left to right are snapshots at time 0.16s, 0.32s, 0.72s, and 0.96s. The red arrow represents the contact forces. The magenta arrow represents the current velocity. The short blue line pointing forwards represents the head position of the robot. Note at t=0.32, the robot tilts its body using gravity to cancel the moment from the contact forces.

reasonable time. This approach is used for solving forward walking, disturbance rejection, and large angle turning trajectories for a quadruped robot.

8.2.4.1 Forward walking

We first generate the control input to make the robot move forward. The objective function we use is:

$$\begin{aligned} f_{obj} &= \|\mathbf{v}[i] - \mathbf{v}_{ref}\|^2_{\mathbf{w}_v} + \|\boldsymbol{\Theta}[i] - \boldsymbol{\Theta}_{ref}\|^2_{\mathbf{w}_\theta} \\ &+ \|\mathbf{p}_z[i] - \mathbf{p}_{z,ref}\|^2_{w_h} + \sum_i \|\mathbf{p}[i+1] - \mathbf{p}[i]\|^2 \\ &+ \sum_{i,\ s\neq t} \|\mathbf{f}_s[i] - \mathbf{f}_t[i]\|^2 \end{aligned} \tag{8.30}$$

Where $\mathbf{w}_v$, $\mathbf{w}_\theta$, w_h are the weights for velocity tracking, rotation angle tracking and body height tracking. For walking forward, we first set $\mathbf{w}_v = [100, 100, 10]$, $\mathbf{w}_\theta = [10, 10, 10]$, $w_h = 10$. This set of weight favors more on the forward walking speed. As a result, the MIP solution consistently gives forward jumping gait despite that the NLP solution being handed as warm-start uses trot gait. The MIP solution, being close to the global optimal solution, significantly im-

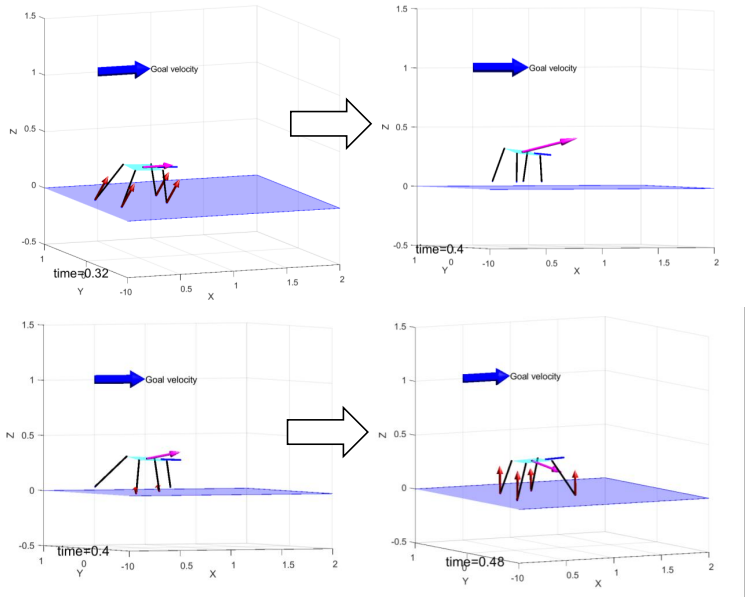


Figure 8.6: Forward walking trajectories and forces. The upper two figures represent the dynamic jumping trajectory from the MIP formulation which uses 4 legs to push simultaneously to maximize the forward velocity. The bottom two figures represent the trot walking trajectory from the NLP formulation which has a slower forward speed. The red arrow represents the contact forces. The magenta arrow represents the current velocity. The short blue line pointing forwards represents the head position of the robot.

proves the tracking for speed as shown by Fig. 8.7. This is intuitively true, as jumping gaits immediately use 4 legs to push on the ground simultaneously, hence more effective in changing speed. On the other hand, if we increase the weight of z velocity and position tracking to 1000 to minimize the z direction body vibration, the optimizer gives a gait that lifts only one leg each iteration. The forward speed tracks much slower in this case. The optimized forward trajectories are shown in Fig. 8.6.

8.2.4.2 Disturbance rejection

In this experiment, we test the controller performance when the objective is moving forward as in the previous section, but there is a large initial backward speed as a disturbance. If the backward speed is relatively smaller, The con-

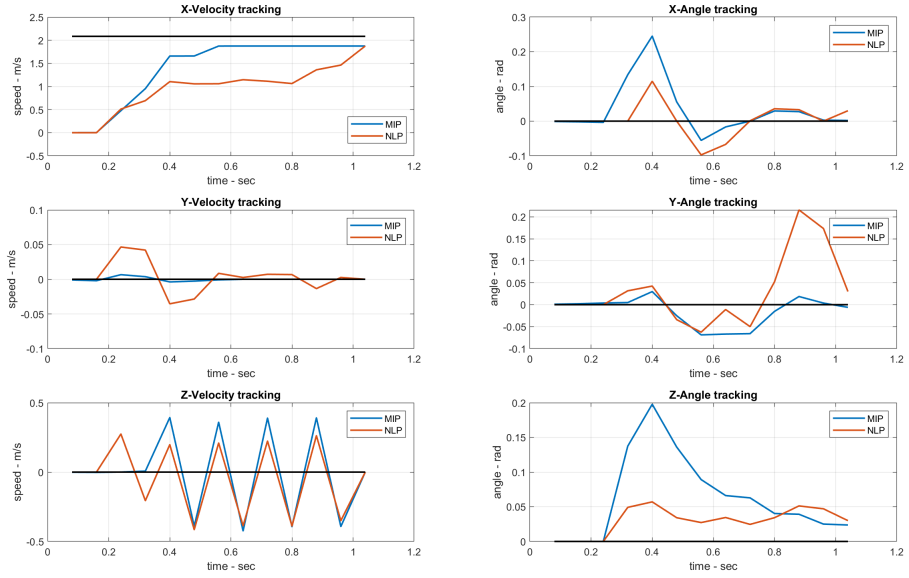


Figure 8.7: Velocity and angle trajectories of the forward walking task. Black lines are the tracking objectives. Orange curves come from the NLP solution. Blue curves come from the MIP solution with optimized gait. X is the forward direction and Y is the side direction. Simply changing the gait can make forward speed tracking much faster. Note that the angle tracking weights are relatively smaller, hence MIP solver may initially choose to sacrifice more angular tracking performance for better speed tracking.

troller commands all the legs to be on the ground for a few iterations, then resumes the forward jumping trajectory. In this situation, all legs create forward forces to cancel the back speed, effectively serving as a brake. If the backward speed is relatively larger, the leg kinematics will be infeasible due to the no-slip constraint as the body quickly goes backward. In this case, the controller generates a back trot gait for a few iterations, then resumes the forward jumping gait. The tracking performance is shown in Fig. 8.8.

8.2.4.3 Large angle turning

In this section, we validate the controller performance to make large angle turns. This serves to verify the controller's ability to select gait and make large

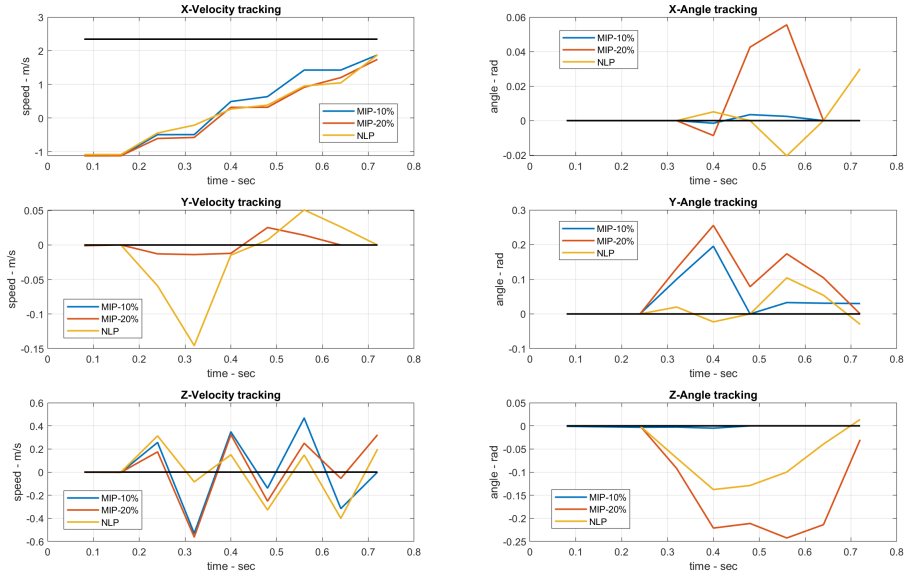


Figure 8.8: Velocity and angle trajectories of the disturbance rejection task. Black lines are the tracking objectives. The 20% gap MIP solution takes 30 min to solve, and the more optimal 10% gap MIP solution takes 45 min. X is the forward direction and Y is the side direction. As the 20% gap MIP solution still does unnecessary motions such as out of plane (Z) rotation, the 10% gap MIP solution almost does not do any unnecessary motions.

orientation changes simultaneously. The initial condition given is $1.8m/s$ forward while the target tracking speed is $1.8m/s$ but at a $90deg$ angle to the side. The NLP solution with initial guesses of trot gait remains using the trot gait to perform the turning. However, the trajectory improved by the MIP will generate a jumping-type of gait with a jump first to cancel the forward velocity. It then simultaneously walks sideways and rotates the body using a non-traditional gait. The velocity and angle tracking plots are shown in Fig. 8.9, and the snapshots of the trajectory are shown in Fig 8.5. Intuitively, the more optimized MIP solution makes the motion much more dynamic. Note that when the robot breaks to cancel the forward velocity (iteration 4, time=0.32s), it needs to generate a large breaking force on its toe. The robot shifts the center of mass backward using the moment from gravity to cancel the moment from

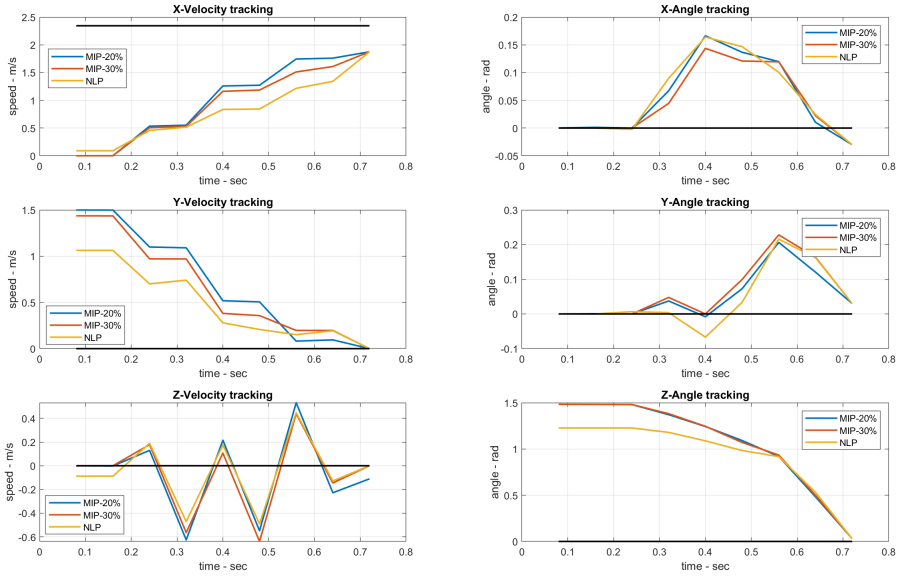


Figure 8.9: Velocity and angle trajectories of the large angle turning task. Black lines are the tracking objectives. The 30% gap MIP solution takes 30 min to solve, and the more optimal 20% gap MIP solution takes 90 min. X is the forward direction and Y is the side direction. Although the NLP solution has much better initial conditions (we cannot find a feasible solution with the exact initial condition to the MIPs), both the MIP solutions quickly catch up and track the objectives faster.

the braking force, such that a larger braking force can be used.

8.2.4.4 Approximation accuracy

In this section, we show the approximation accuracy using McCormick envelopes for bilinear and trilinear terms. Since we use envelopes to locally convexify the constraints, it is important to ensure reasonable approximation accuracy. There are several types of non-convex terms: bilinear trig multiplication terms such as $sin(\theta_0)cos(\theta_1)$, bilinear angular velocity multiplication terms such as $w_0 w_1$, bilinear moment terms such as $p_x f_y$, trilinear trig multiplication terms such as $sin(\theta_0)sin(\theta_1)cos(\theta_2)$, trilinear trig and Euler angular rate multiplication terms such as $sin(\theta_2)cos(\theta_1)\dot{\theta}_0$. We average the approximation

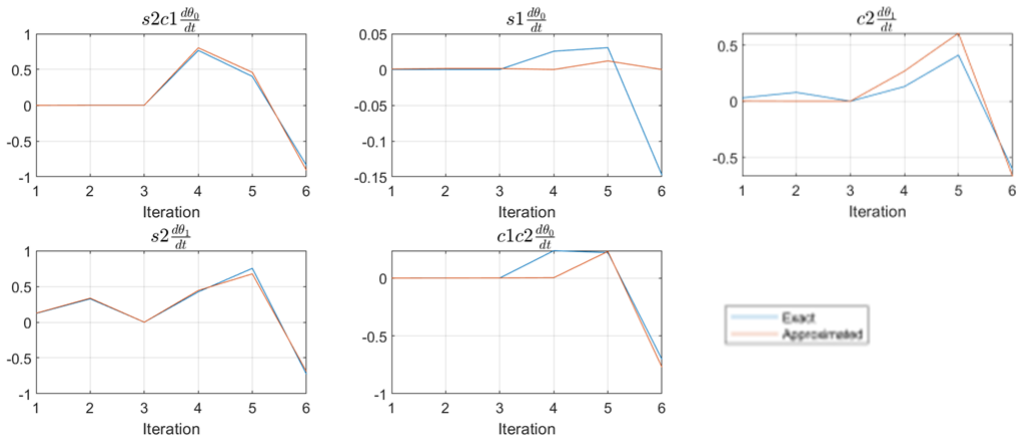


Figure 8.10: Approximation accuracy along a typical trajectory.

accuracy of those terms across the trajectory for a 200 point dataset using the following equation:

$$error_x = \frac{|x_{approx} - x_{true}|}{max(|x_{approx}|, |x_{true}|)} \tag{8.31}$$

The results are listed in Table 8.5. An actual trajectory of approximated and true values for a few non-convex terms is shown by Fig. 8.10. The results show that the approximated values clearly resemble the trend of change in the actual values, and the accuracy is generally around 10% or less. We note that since the model may not be perfect, a rough approximation leaves some room for tuning on actual hardware which is sometimes favored.

Table 8.5: Average approximation accuracy

term	$s\theta_0 c\theta_1$	$s\theta_0 s\theta_1 c\theta_2$	$s\theta_2 c\theta_1(\dot{\theta}_0)$	$w_0 w_1$	$p_x f_y$
error	10.84%	12.38%	2.86%	11.73%	7.81%

8.2.4.5 Dataset collection and hardware implementation

We implemented our proposed control framework on a position-controlled quadruped robot SCALER. The experiment is a forward walking task similar

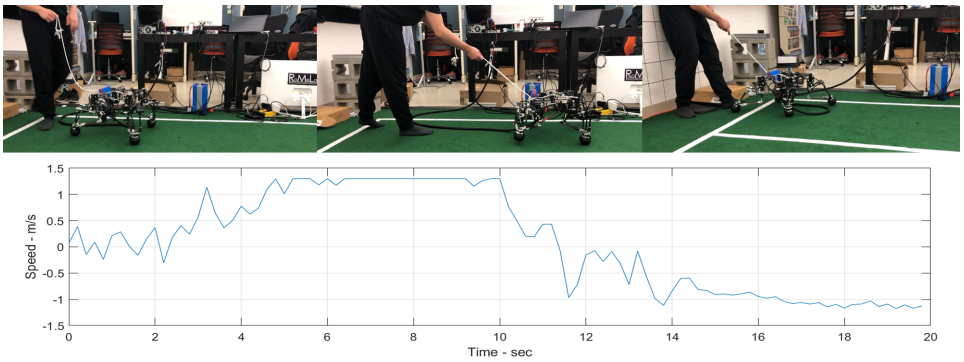


Figure 8.11: Hardware experiment for walking. Up: 3 snapshots of the robot walking forward using the proposed framework and changing gait when dragged back. Down: estimated forward speed from state estimation.

to what is described in section 8.2.4.1 and 8.2.4.2, designed to observe online gait change due to MPC results given the state of the robot. Since this robot is position controlled, we use the admittance control law with force torque sensors on its feet to track the planned position and force profile. This same controller is used throughout the trajectory. We collected 110 trajectories with 720 ms long, 9 iterations each, that begin with randomized initial conditions of $vx \in [-1.5, 1.5]$ and $ax \in [-15, 15]$. Those trajectories give information on how the robot can reach the goal velocity from different initial conditions in the most optimal approach. Since our robot cannot track the jumping trajectory, we force the optimizer to have at least 2 contact points on the ground for each iteration. The MPC plan 5 iterations ahead (400 ms) for the robot to follow the trajectories and reach the goal. Along the full trajectory, we segment out 5 iteration sections, 220 in total, and use them as data for the learner. Our previous tests on smaller scale problems [34, 148] show that the K-nearest neighbor method works decently well on problems of this scale, hence is used in this experiment. Typical force MPC algorithms only need to know the current state to plan ahead. This is because there is a pre-generated gait pattern from a high-level planner. However, for our hybrid MPC, the gait and forces are both generated online. To reason the gait which is of slower changing frequency, the

learner needs to have information of a short history of the trajectory to decide the next lifting leg. Otherwise, the learner may keep on lifting the same set of legs. We feed the current forward velocity, forward acceleration, and the toe positions in the previous 2 iterations to the KNN learner which provides the integer variables in the next 5 iterations. With the integer variables provided, convex optimization is solved for the actual contact positions, forces, body orientations, and corresponding speed and acceleration quantities. We use OSQP solver [177] for online solving. One feature of KNN is that it can provide several candidates of integer variables. If one fails, the solver can try the next one until one succeeds. For this problem, as long as the input to KNN is within the region of the randomization regime, it usually takes only 1 trial to find a feasible solution. The average solving speed is 19 ms on an Intel Core i7-12800H laptop or 53 Hz. State estimation [101] that fuses encoder and IMU information is used to estimate the forward velocity used by KNN.

The hardware experiment begins with the robot making a step forward that gains some speed, which is fed back to KNN such that the robot proceeds to follow the trajectory. The human operator then manually pulls the robot backward. With the sensed negative speed, the robot then stops trotting forward but keeps all its toes on the ground to maximize the braking force as described in section 8.2.4.2. The process is shown in the upper part of Fig. 8.11. The estimated forward velocity is shown in the lower part of the same figure.

8.2.4.6 Solving speed

For mixed-integer nonconvex problems such as locomotion planning with a single rigid body, directly solving the MIP formulation with envelope constraints is mostly infeasible. The solver can sometimes take days before it can find even one feasible solution. First sampling the problem parameters around the targeted problem and solving a small set of problems in NLP formulation,

Table 8.6: Problem sizes and solving speeds

	Offline TO (NLP)	Offline TO (MIP)	Online MPC	Online NLP benchmark
# of iterations	9 (720 ms)	9 (720 ms)	5 (400 ms)	5 (400 ms)
# of continuous variables	1143	80321	34549	579
# of binary variables	N.A.	976	488	N.A.
# of constraints	2327	103430	44478	1115
Avg. solving time	1.34 s (when feasible with trot gait initial guess)	See section 8.2.4.6	19 ms (53 Hz)	498 ms (2 Hz, when feasible with trot gait initial guess)
Solver	IPOPT	Gurobi	OSQP	IPOPT

then using them to partially warm-start the target problem in MIP formulation dramatically speeds up the MIP solving speed. For our single rigid body problem, the MIP solver with warm-start tends to make quick progress to minimize the primal-dual gap in the next 30 minutes to 1 hour, until it reaches around 15% and slows down again. For the disturbance rejection task, the solver takes 30 minutes to decrease the bound to 20%, and 45 minutes to further decrease it to 10%. For the large angle turning task, it takes 30 minutes to decrease the bound to 30%, and 90 min to further decrease it to 20%. In many cases, this process already makes the solution significantly more optimal than the original solutions from NLP solvers. If the goal is to optimize the gait, the MIP solver takes seconds to finish while keeping the large angle rotation trajectories from NLP solvers. After MIP solving, the more optimal solutions can be added to the dataset which makes the following process even faster.

Table 8.6 gives the number of variables, constraints, and solving time for both the offline and online optimizations. The online MPC with KNN learned integer variables can run more than 50 Hz on the OSQP solver which is sufficient for hybrid MPC. In comparison, solving the same problem with NLP formulation on the IPOPT solver is significantly slower. The commercialized KNITRO solver may run faster than IPOPT but we expect it not to be able to catch the convex MPC solving speed.

CHAPTER 9

Conclusion

In this dissertation, we mainly complete three things: First, we developed compliance models for a multi-limbed robot that allows it to climb up by bracing between two walls. Second, we developed combinatorial optimization based algorithms for motion planning and control with applications on robot walking and climbing, item manipulation, and self-reconfigurable robot motion generation. Third, we implemented data-driven methods to speed up the solving process and enhance the robustness of optimization solvers.

9.1 Compliance Model

The compliance model developed in this thesis can be used for gauging the robot sagdown due to compliance. This is used for our robot SCALER, and can be extended for other types of robots such as soft robots. The virtual joint method is a simplification of the more complicated finite element analysis based method. The complexity is dramatically reduced, but still very expensive to resolve online. Direct nonlinear programming may be used if solving time is not a concern. For fast solutions, data-driven methods are still good options.

9.2 Vertical Climbing

As a unique type of locomotion method, vertical climbing combines legged locomotion and grasping into the problem hence is an interesting one to investigate but has been less explored. With the assistance of grippers, robot climbers can potentially traverse any terrain including even vertical walls. Interesting

research can be done to apply robot climbers to space exploration, disaster rescue, construction, and so on. We are particularly interested in climbing applications on earth. Being able to achieve such tasks allows robot climbers to perform dangerous jobs which can save numerous human workers. To achieve climbing, reliability is required both in terms of hardware platform and planning/control algorithms.

First of all, the robot climber needs to carry a non-trivial payload such as investigation and cleaning equipment to perform practical applications. A loading capacity of at least 5kg is expected. Among the few current platforms capable of doing this, the most famous one is the JPL lemur 3 robot. However, this robot, designed for space exploration, uses highly over-designed grippers and extremely strong actuators without any contact force sensing. The strategy is to ensure that the gripper can simply be attached to the rock-based wall without any control as the large number of spines ensure a grip with a high chance. However, this strategy will fail if the graspable spot is very sparse. Additionally, the over-designed gripper is heavy which further limits the loading capacity.

Our goal along this line of work is to fundamentally change the strategy. We plan to use a gripper that only has a few fingers like a human hand and climb on discrete handholds such as rock climbing walls. This means we need to carefully reason the grasp beforehand and control the contact wrench using rich feedback signals. Moreover, to increase the loading capacity, linkage designs are being investigated. SCALER is equipped with force torque sensors on its end effector, RGB-D cameras, and other sensors allowing the implementation of motion planning and control algorithms.

9.3 Optimization Based Motion Planning

Several optimization-based motion planning schemes are developed in this thesis with the goal in mind to solve a relatively more complicated model faster. Our formulation is quasi-static but incorporates compliance models that did not appear in the previous works. This model can be solved in a decoupled 2-stage approach by mixed-integer programs, or with a nonlinear programming formulation. One significant slowdown of mixed-integer programming formulation comes from the contact selection. If the number of contact regions becomes large, mixed-integer programming can sometimes be significantly slower. The slowdown is even worse with dynamics which needs to be approximated through convex envelope relaxations. On the other hand, nonlinear programs can resolve dynamics relatively faster. With complementary formulations, it can also deal with discrete terrain shapes. However, complementary constraints are usually numerically difficult for the gradient-based solver, despite smart formulations such as [63] existing. In both cases, as the problem scale becomes larger, the solving process is too slow for practical robotics implementations that require real-time operation.

9.4 Data-driven Methods for Combinatorial Optimization

Due to the recent progress of deep learning, researchers have started to investigate data-driven approaches to speed up optimization solvers. Typical methods involve using graph neural networks [38, 191, 192]. While this direction is promising, the purpose of the current literature is to speed up the MIP solver for general purposes such as operational research. Many problems in those fields do not require real-time solving speed. The applications of those ideas and methods to robot motion planning or control problems still require major testing. Many questions are yet to be answered. Can we bound the solving

time? Is the controller stable? This thesis implements a simple data-driven idea to speed up the solver for a small-scale bookshelf organization problem. The solving speed itself is sufficient for real-time applications such as MPC. However, strict verifications to answer the questions above are required. In addition, bookshelf organization problems and other problems such as the modular robot problem studied in this thesis can be easily scaled up to real-world size, such that the current solver cannot handle within a reasonable time. Toy problems such as inverted pendulum with contact likely will give us clues and insights into those questions.

[1] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain", in *2019 IEEE-RAS 19th international conference on humanoid robots (humanoids)*. IEEE, 2019, pp. 9–16.

[2] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, "Models, feedback control, and open problems of 3d bipedal robotic walking", *Automatica*, vol. 50, no. 8, pp. 1955–1988, 2014.

[3] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics", *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.

[4] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem", *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 317–342, 2006.

[5] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot", *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[6] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[7] X. Lin, J. Zhang, J. Shen, G. Fernandez, and D. W. Hong, "Optimization based motion planning for multi-limbed vertical climbing robots", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1918–1925.

[8] M. S. Ahn, H. Chae, and D. W. Hong, "Stable, autonomous, unknown terrain locomotion for quadrupeds based on visual feedback and mixed-integer convex optimization", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3791–3798.

[9] Y. Shirai, X. Lin, Y. Tanaka, A. Mehta, and D. Hong, "Risk-aware motion planning for a limbed robot with stochastic gripping forces using nonlinear programming", *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 4994–5001, 2020.

[10] Y. Shirai, X. Lin, A. Mehta, and D. Hong, "Lto: Lazy trajectory optimization with graph-search planning for high dof robots in cluttered environments", *arXiv preprint arXiv:2103.01333*, 2021.

[11] J. Zhang, X. Lin, and D. W. Hong, "Transition motion planning for multi-limbed vertical climbing robots using complementarity constraints", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2033–2039.

[12] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning", *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[13] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real", in *Conference on Robot Learning*. PMLR, 2020, pp. 317–329.

[14] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning task space actions for bipedal locomotion", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1276–1282.

[15] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain", in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2665–2670.

[16] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning", in *2001 European control conference (ECC)*. IEEE, 2001, pp. 2603–2608.

[17] J. Zhou, R. He, Y. Wang, S. Jiang, Z. Zhu, J. Hu, J. Miao, and Q. Luo, "Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 439–446, 2020.

[18] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization", in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[19] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics", in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[20] M. Soler, A. Olivares, P. Bonami, and E. Staffetti, "En-route optimal flight planning constrained to pass through waypoints using minlp", 2011.

[21] S. Boyd and J. Mattingley, "Branch and bound methods", *Notes for EE364b, Stanford University*, pp. 2006–07, 2007.

[22] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments", in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 1934–1940.

[23] X. Lin, M. S. Ahn, and D. Hong, "Designing multi-stage coupled convex programming with data-driven mccormick envelope relaxations for motion planning", in *2021 IEEE/RSJ International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[24] X. Da, R. Hartley, and J. W. Grizzle, "Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3476–3483.

[25] R. L. H. Deits, *Convex segmentation and mixed-integer footstep planning for a walking robot*, PhD thesis, Massachusetts Institute of Technology, 2014.

[26] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization.", in *Robotics: Science and Systems*, 2014, vol. 4.

[27] A. Cauligi, P. Culbertson, E. Schmerling, M. Schwager, B. Stellato, and M. Pavone, "Coco: Online mixed-integer control via supervised learning", *arXiv preprint arXiv:2107.08143*, 2021.

[28] J.-J. Zhu and G. Martius, "Fast non-parametric learning to accelerate mixed-integer programming for online hybrid model predictive control", *arXiv preprint arXiv:1911.09214*, 2019.

[29] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control", *arXiv preprint arXiv:2109.14152*, 2021.

[30] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, "A clustering technique for the identification of piecewise affine systems", *Automatica*, vol. 39, no. 2, pp. 205–217, 2003.

[31] G. Ferrari-Trecate and M. Schinkel, "Conditions of optimal classification for piecewise affine regression", in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2003, pp. 188–202.

[32] H. Nakada, K. Takaba, and T. Katayama, "Identification of piecewise affine systems based on statistical clustering technique", *Automatica*, vol. 41, no. 5, pp. 905–913, 2005.

[33] S. Paoletti, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal, "Identification of hybrid systems a tutorial", *European journal of control*, vol. 13, no. 2-3, pp. 242–260, 2007.

[34] X. Lin, G. I. Fernandez, and D. W. Hong, "Reduce: Reformulation of mixed integer programs using data from unsupervised clusters for learning efficient strategies", *arXiv preprint arXiv:2110.00666*, 2021.

[35] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots", *Science Robotics*, vol. 4, no. 26, pp. eaau5872, 2019.

[36] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning", *arXiv preprint arXiv:2105.08328*, 2021.

[37] I. A. OpenAI, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al., "Solving rubik's cube with a robot hand", 2019.

[38] V. Nair, S. Bartunov, F. Gimeno, I. von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, et al., "Solving mixed integer programs using neural networks", *arXiv preprint arXiv:2012.13349*, 2020.

[39] A. Cauligi, P. Culbertson, B. Stellato, D. Bertsimas, M. Schwager, and M. Pavone, "Learning mixed-integer convex optimization strategies for robot planning and control", *arXiv preprint arXiv:2004.03736*, 2020.

[40] Y. Tang, S. Agrawal, and Y. Faenza, "Reinforcement learning for integer programming: Learning to cut", in *International Conference on Machine Learning*. PMLR, 2020, pp. 9367–9376.

[41] P. Beardsley, R. Siegwart, M. Arigoni, M. Bischoff, S. Fuhrer, D. Krummenacher, D. Mammolo, and R. Simpson, "Vertigo-a wall-climbing robot including ground-wall transition", *Disney Research*, 2015.

[42] M. P. Murphy and M. Sitti, "Waalbot: An agile small-scale wall-climbing robot utilizing dry elastomer adhesives", *IEEE/ASME transactions on Mechatronics*, vol. 12, no. 3, pp. 330–338, 2007.

[43] S. Kim, M. Spenko, S. Trujillo, B. Heyneman, D. Santos, and M. R. Cutkosky, "Smooth vertical surface climbing with directional adhesion", *IEEE Transactions on robotics*, vol. 24, no. 1, pp. 65–74, 2008.

[44] M. J. Spenko, G. C. Haynes, J. Saunders, M. R. Cutkosky, A. A. Rizzi, R. J. Full, and D. E. Koditschek, "Biologically inspired climbing with a

hexapedal robot", *Journal of field robotics*, vol. 25, no. 4-5, pp. 223–242, 2008.

[45] A. Parness, N. Abcouwer, C. Fuller, N. Wiltsie, J. Nash, and B. Kennedy, "Lemur 3: A limbed climbing robot for extreme terrain mobility in space", in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5467–5473.

[46] G. A. Lynch, J. E. Clark, P.-C. Lin, and D. E. Koditschek, "A bioinspired dynamical vertical climbing robot", *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 974–996, 2012.

[47] J. Clark, D. Goldman, P.-C. Lin, G. Lynch, T. Chen, H. Komsuoglu, R. J. Full, and D. E. Koditschek, "Design of a bio-inspired dynamical vertical climbing robot.", in *Robotics: Science and Systems*, 2007, vol. 1.

[48] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning", *IEEE transactions on systems, man, and cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[49] M. Posa, M. Tobenkin, and R. Tedrake, "Stability analysis and control of rigid-body systems with impacts and friction", *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1423–1437, 2015.

[50] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies", *Science Robotics*, vol. 4, no. 26, pp. eaau4984, 2019.

[51] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[52] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of a", *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 505–536, 1985.

[53] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality", *Advances in neural information processing systems*, vol. 16, 2003.

[54] A. Stentz et al., "The focussed d^* algorithm for real-time replanning", in *IJCAI*, 1995, vol. 95, pp. 1652–1659.

[55] M. Likhachev and A. Stentz, "R* search", 2008.

[56] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid", in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 629–634.

[57] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[58] S. M. LaValle et al., "Rapidly-exploring random trees: A new tool for path planning", 1998.

[59] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning", *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[60] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints", in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2859–2865.

[61] H. Dai, G. Izatt, and R. Tedrake, "Global inverse kinematics via mixed-integer convex optimization", *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1420–1441, 2019.

[62] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems", *Mathematical programming*, vol. 10, no. 1, pp. 147–175, 1976.

[63] O. Stein, J. Oldenburg, and W. Marquardt, "Continuous reformulations of discrete–continuous optimization problems", *Computers & chemical engineering*, vol. 28, no. 10, pp. 1951–1966, 2004.

[64] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact", *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[65] G. Bledt and S. Kim, "Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6316–6323.

[66] G. Bledt and S. Kim, "Extracting legged locomotion heuristics with regularized predictive control", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 406–412.

[67] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[68] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain", *Science robotics*, vol. 5, no. 47, pp. eabc5986, 2020.

[69] K. Green, Y. Godse, J. Dao, R. L. Hatton, A. Fern, and J. Hurst, "Learning spring mass locomotion: Guiding policies with a reduced-order model", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3926–3932, 2021.

[70] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7309–7315.

[71] M. Neunert, A. Abdolmaleki, M. Wulfmeier, T. Lampe, T. Springenberg, R. Hafner, F. Romano, J. Buchli, N. Heess, and M. Riedmiller, "Continuous-discrete reinforcement learning for hybrid control in robotics", in *Conference on Robot Learning*. PMLR, 2020, pp. 735–751.

[72] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement learning on variable impedance controller for high-precision robotic assembly", in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3080–3087.

[73] X. Da and J. Grizzle, "Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots", *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1063–1097, 2019.

[74] R. Tedrake, *Underactuated Robotics*, 2022.

[75] M. H. Raibert, *Legged robots that balance*, MIT press, 1986.

[76] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, "Fast trajectory optimization for legged robots using vertex-based zmp constraints", *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2201–2208, 2017.

[77] J. R. Hooks, *Real-time optimization for control of a multi-modal legged robotic system*, University of California, Los Angeles, 2019.

[78] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point", in *2003 IEEE International Conference on*

Robotics and Automation (Cat. No. 03CH37422). IEEE, 2003, vol. 2, pp. 1620–1626.

[79] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery", in *2006 6th IEEE-RAS international conference on humanoid robots*. IEEE, 2006, pp. 200–207.

[80] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control", in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[81] H. Hirukawa, S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, and M. Morisawa, "A universal stability criterion of the foot contact of legged robots-adios zmp", in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1976–1983.

[82] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization", in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 579–586.

[83] S. Caron, Q.-C. Pham, and Y. Nakamura, "Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5107–5112.

[84] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[85] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.", in *ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[86] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[87] N. J. Kong, G. Council, and A. M. Johnson, "ilqr for piecewise-smooth hybrid dynamical systems", *arXiv preprint arXiv:2103.14584*, 2021.

[88] E. D. Sontag, "A 'universal'construction of artstein's theorem on nonlinear stabilization", *Systems & control letters*, vol. 13, no. 2, pp. 117–123, 1989.

[89] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications", in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[90] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems", *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[91] M. Ahmadi, X. Xiong, and A. D. Ames, "Risk-averse control via cvar barrier functions: Application to bipedal robot locomotion", *IEEE Control Systems Letters*, vol. 6, pp. 878–883, 2021.

[92] X. Xiong and A. D. Ames, "Coupling reduced order models via feedback control for 3d underactuated bipedal robotic walking", in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[93] J.-K. Huang and J. W. Grizzle, "Efficient anytime clf reactive planning system for a bipedal robot on undulating terrain", *arXiv preprint arXiv:2108.06699*, 2021.

[94] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3821–3828.

[95] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming", in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4054–4061.

[96] A. Hereid, C. M. Hubicki, E. A. Cousineau, J. W. Hurst, and A. D. Ames, "Hybrid zero dynamics based multiple shooting optimization with applications to robotic walking", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5734–5740.

[97] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification", *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[98] P. S. Schmitt, F. Wirnshofer, K. M. Wurm, G. v. Wichert, and W. Burgard, "Planning reactive manipulation in dynamic environments", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 136–143.

[99] M. X. Grey, A. D. Ames, and C. K. Liu, "Footstep and motion planning in semi-unstructured environments using randomized possibility graphs", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4747–4753.

[100] Y. Tanaka, Y. Shirai, Z. Lacey, X. Lin, J. Liu, and D. Hong, "An underactuated whippletree mechanism gripper based on multi-objective design optimization with auto-tuned weights", in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6139–6146.

[101] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots-consistent fusion of leg kinematics and imu", *Robotics*, vol. 17, pp. 17–24, 2013.

[102] A. Schperberg, Y. Shirai, X. Lin, Y. Tanaka, and D. Hong, "Auto-calibrating admittance controller for robust motion of robotic systems", *arXiv preprint arXiv:2207.01033*, 2022.

[103] E. Moore, D. Campbell, F. Grimminger, and M. Buehler, "Reliable stair climbing in the simple hexapod'rhex"', in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. IEEE, 2002, vol. 3, pp. 2222–2227.

[104] Y. Tanaka, X. Lin*, Y. Shirai*, A. Schperberg, H. Kato, A. Swerdlow, N. Kumagai, and D. Hong, "Scaler: A tough versatile quadruped free-climber robot", *arXiv preprint arXiv:2207.01180*, 2022.

[105] S. Wang, H. Jiang, and M. R. Cutkosky, "A palm for a rock climbing robot based on dense arrays of micro-spines", in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 52–59.

[106] S. Wang, H. Jiang, and M. R. Cutkosky, "Design and modeling of linearly-constrained compliant spines for human-scale locomotion on rocky surfaces", *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 985–999, 2017.

[107] S. Wang, H. Jiang, T. Myung Huh, D. Sun, W. Ruotolo, M. Miller, W. R. Roderick, H. S. Stuart, and M. R. Cutkosky, "Spinyhand: Contact load sharing for a human-scale climbing robot", *Journal of Mechanisms and Robotics*, vol. 11, no. 3, pp. 031009, 2019.

[108] H. Jiang, S. Wang, and M. R. Cutkosky, "Stochastic models of compliant spine arrays for rough surface grasping", *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 669–687, 2018.

[109] K. Hauser, S. Wang, and M. R. Cutkosky, "Efficient equilibrium testing under adhesion and anisotropy using empirical contact force models", *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1157–1169, 2018.

[110] W. R. Provancher, J. E. Clark, B. Geisler, and M. R. Cutkosky, "Towards penetration-based clawed climbing", in *Climbing and walking robots*, pp. 961–970. Springer, 2005.

[111] X. Lin, H. Krishnan, Y. Su, and D. W. Hong, "Multi-limbed robot vertical two wall climbing based on static indeterminacy modeling and feasibility region analysis", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4355–4362.

[112] X. Lin and D. Hong, "Formulation of posture optimization for multi-legged robot via eigen decomposition of stiffness matrix", in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2016, pp. 104–108.

[113] J. K. Salisbury, "Active stiffness control of a manipulator in cartesian coordinates", in *1980 19th IEEE conference on decision and control including the symposium on adaptive processes*. IEEE, 1980, pp. 95–100.

[114] T. Bretl and S. Lall, "A fast and adaptive test of static equilibrium for legged robots", in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1109–1116.

[115] X. Gao and S.-M. Song, "A stiffness matrix method for foot force distribution of walking vehicles", *IEEE transactions on systems, man, and cybernetics*, vol. 22, no. 5, pp. 1131–1138, 1992.

[116] X. Gao, S.-M. Song, and C. Q. Zheng, "A generalized stiffness matrix method for force distribution of robotic systems with indeterminancy", 1993.

[117] V. Kumar and K. Waldron, "Force distribution in walking vehicles", *Journal of Mechanical Design*, vol. 112, no. 1, pp. 90–99, 1990.

[118] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation", in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 842–849.

[119] P. M. Castro, "Tightening piecewise mccormick relaxations for bilinear problems", *Computers & Chemical Engineering*, vol. 72, pp. 300–311, 2015.

[120] B. Acikmese and S. R. Ploen, "Convex programming approach to powered descent guidance for mars landing", *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[121] RoMeLa, "Icra2021 - multi-stage coupled convex programming with data-driven mccormick envelopes".

[122] G. Optimization, "Inc.,"gurobi optimizer reference manual," 2015", 2014.

[123] H. P. Williams, *Model building in mathematical programming*, John Wiley & Sons, 2013.

[124] X. Lin, G. Fernandez, Y. Liu, T. Zhu, Y. Shirai, and D. Hong, "Multi-modal multi-agent optimization for limms, a modular robotics approach to delivery automation", 2022.

[125] T. Zhu, G. I. Fernandez, C. Togashi, Y. Liu, and D. Hong, "Feasibility study of limms, a multi-agent modular robotic delivery system with various locomotion and manipulation modes", in *2022 19th International Conference on Ubiquitous Robots (UR)*. IEEE, 2022, pp. 30–37.

[126] C. Liu, M. Whitzer, and M. Yim, "A distributed reconfiguration planning algorithm for modular robots", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4231–4238, 2019.

[127] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu, "Miche: Modular shape formation by self-disassembly", *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 345–372, 2008.

[128] Y. Zhang, W. Wang, P. Zhang, and P. Huang, "Reinforcement learning-based task planning for self-reconfiguration of cellular satellites", *IEEE Aerospace and Electronic Systems Magazine*, 2021.

[129] N. Gandhi, D. Saldana, V. Kumar, and L. T. X. Phan, "Self-reconfiguration in response to faults in modular aerial systems", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2522–2529, 2020.

[130] T. Kataoka, "A multi-period mixed integer programming model on reconfigurable manufacturing cells", *Procedia Manufacturing*, vol. 43, pp. 231–238, 2020.

[131] A. Aydinoglu and M. Posa, "Real-time multi-contact model predictive control via admm", *arXiv preprint arXiv:2109.07076*, 2021.

[132] Y. Shirai, X. Lin, A. Schperberg, Y. Tanaka, H. Kato, V. Vichathorn, and D. Hong, "Simultaneous contact-rich grasping and locomotion via distributed optimization enabling free-climbing for multi-limbed robots", *IEEE Proc. 2022 IEEE/RSJ Int. Conf. Intell. Rob. Syst*, 2022.

[133] G. I. Fernandez, S. Gessow, J. Quan, and D. Hong, "Self-aligning rotational latching mechanisms", in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2022.

[134] J. Hooks, M. S. Ahn, J. Yu, X. Zhang, T. Zhu, H. Chae, and D. Hong, "Alphred: A multi-modal operations quadruped robot for package delivery applications", *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5409–5416, 2020.

[135] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[136] A. U. Raghunathan, D. K. Jha, and D. Romeres, "Pyrobocop: Python-based robotic control & optimization package for manipulation and collision avoidance", *arXiv preprint arXiv:2106.03220*, 2021.

[137] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers", *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[138] M. Stolle and C. G. Atkeson, "Policies based on trajectory libraries", in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 3344–3349.

[139] C. Liu, C. G. Atkeson, and J. Su, "Biped walking control using a trajectory library", *Robotica*, vol. 31, no. 2, pp. 311–322, 2013.

[140] Y. Tassa, T. Erez, and W. Smart, "Receding horizon differential dynamic programming", *Advances in neural information processing systems*, vol. 20, 2007.

[141] A. V. Fiacco, *Introduction to sensitivity and stability analysis in nonlinear programming*, vol. 165, Academic press, 1983.

[142] E. N. Pistikopoulos, V. Dua, N. A. Bozinis, A. Bemporad, and M. Morari, "On-line optimization via off-line parametric optimization tools", *Computers & Chemical Engineering*, vol. 26, no. 2, pp. 175–185, 2002.

[143] T. Gal, *Postoptimal Analyses, Parametric Programming, and Related Topics: degeneracy, multicriteria decision making, redundancy*, Walter de Gruyter, 2010.

[144] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems", *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[145] V. Dua and E. N. Pistikopoulos, "Algorithms for the solution of multiparametric mixed-integer nonlinear optimization problems", *Industrial & engineering chemistry research*, vol. 38, no. 10, pp. 3976–3987, 1999.

[146] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit mpc solutions", *Automatica*, vol. 39, no. 3, pp. 489–497, 2003.

[147] V. Dua and E. N. Pistikopoulos, "An algorithm for the solution of multiparametric mixed integer linear programming problems", *Annals of operations research*, vol. 99, no. 1, pp. 123–139, 2000.

[148] K. Hauser, "Learning the problem-optimum map: Analysis and application to global optimization in robotics", *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 141–152, 2016.

[149] G. Tang and K. Hauser, "A data-driven indirect method for nonlinear optimal control", *Astrodynamics*, vol. 3, no. 4, pp. 345–359, 2019.

[150] T. Power and D. Berenson, "Variational inference mpc using normalizing flows and out-of-distribution projection", *arXiv preprint arXiv:2205.04667*, 2022.

[151] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "An algebraic geometric approach to the identification of a class of linear hybrid systems", in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*. IEEE, 2003, vol. 1, pp. 167–172.

[152] W. Jin, A. Aydinoglu, M. Halm, and M. Posa, "Learning linear complementarity systems", in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 1137–1149.

[153] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey, "Cutting planes in integer and mixed integer", *Discrete Optimization: The State of the Art*, vol. 11, pp. 397, 2003.

[154] M. Anitescu and F. A. Potra, "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems", *Nonlinear Dynamics*, vol. 14, no. 3, pp. 231–247, 1997.

[155] E. Drumwright, "Rapidly computable viscous friction and no-slip rigid contact models", *arXiv preprint arXiv:1504.00719*, 2015.

[156] M. Kojima, N. Megiddo, T. Noma, and A. Yoshise, "A unified approach to interior point algorithms for linear complementarity problems: A summary", *Operations Research Letters*, vol. 10, no. 5, pp. 247–254, 1991.

[157] S. L. Cleac'h, T. Howell, M. Schwager, and Z. Manchester, "Fast contact-implicit model-predictive control", *arXiv preprint arXiv:2107.05616*, 2021.

[158] J. Park and S. Boyd, "A semidefinite programming method for integer convex quadratic minimization", *Optimization Letters*, vol. 12, no. 3, pp. 499–518, 2018.

[159] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey", *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[160] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, vol. 6, Athena Scientific Belmont, MA, 1997.

[161] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey, "Cutting planes in integer and mixed integer programming", *Discrete Applied Mathematics*, vol. 123, no. 1-3, pp. 397–446, 2002.

[162] C. A. Meyer and C. A. Floudas, "Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes", *Journal of Global Optimization*, vol. 29, no. 2, pp. 125–155, 2004.

[163] M. Fischetti and A. Lodi, "Local branching", *Mathematical programming*, vol. 98, no. 1, pp. 23–47, 2003.

[164] E. Danna, E. Rothberg, and C. L. Pape, "Exploring relaxation induced neighborhoods to improve mip solutions", *Mathematical Programming*, vol. 102, no. 1, pp. 71–90, 2005.

[165] S. Ghosh, "Dins, a mip improvement heuristic", in *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 2007, pp. 310–323.

[166] T. Berthold, S. Heinz, M. Pfetsch, and S. Vigerske, "Large neighborhood search beyond mip", 2012.

[167] T. Berthold, "Rens", *Mathematical Programming Computation*, vol. 6, no. 1, pp. 33–54, 2014.

[168] G. Nannicini, P. Belotti, and L. Liberti, "A local branching heuristic for minlps", *arXiv preprint arXiv:0812.2188*, 2008.

[169] T. Berthold and A. M. Gleixner, "Undercover: a primal minlp heuristic exploring a largest sub-mip", *Mathematical Programming*, vol. 144, no. 1, pp. 315–346, 2014.

[170] A. Chmiela, E. Khalil, A. Gleixner, A. Lodi, and S. Pokutta, "Learning to schedule heuristics in branch and bound", *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[171] N. Sonnerat, P. Wang, I. Ktena, S. Bartunov, and V. Nair, "Learning a large neighborhood search algorithm for mixed integer programs", *arXiv preprint arXiv:2107.10201*, 2021.

[172] D. Liu, M. Fischetti, and A. Lodi, "Learning to search in local branching", *arXiv preprint arXiv:2112.02195*, 2021.

[173] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems", *IEEE Transactions on Automatic Control*, vol. 66, no. 6, pp. 2433–2448, 2020.

[174] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming", in *Algorithmic foundations of robotics XI*, pp. 109–124. Springer, 2015.

[175] L. Dai, Q. Cao, Y. Xia, and Y. Gao, "Distributed mpc for formation of multi-agent systems with collision avoidance and obstacle avoidance", *Journal of the Franklin Institute*, vol. 354, no. 4, pp. 2068–2085, 2017.

[176] A. U. Raghunathan, D. K. Jha, and D. Romeres, "Pyrobocop: Python-based robotic control & optimization package for manipulation", in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 985–991.

[177] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs", *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[178] X. Lin, G. I. Fernandez, and D. W. Hong, "Reduce: Reformulation of mixed integer programs using data from unsupervised clusters for learning efficient strategies", in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4459–4465.

[179] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "Density-based spatial clustering of applications with noise", in *Int. Conf. Knowledge Discovery and Data Mining*, 1996, vol. 240, p. 6.

[180] D. Bertsimas and B. Stellato, "The voice of optimization", *Machine Learning*, vol. 110, no. 2, pp. 249–277, 2021.

[181] D. Bertsimas and B. Stellato, "Online mixed-integer optimization in milliseconds", *arXiv preprint arXiv:1907.02206*, 2019.

[182] X. Lin, G. I. Fernandez, and D. W. Hong, "Benchmark results for bookshelf organization problem as mixed integer nonlinear program with mode switch and collision avoidance", 2022.

[183] J. P. Vielma, "Mixed integer linear programming formulation techniques", *Siam Review*, vol. 57, no. 1, pp. 3–57, 2015.

[184] P. Bonami and J. Lee, "Bonmin user's manual", *Numer Math*, vol. 4, pp. 1–32, 2007.

[185] P. Belotti, L. Liberti, A. Lodi, G. Nannicini, A. Tramontani, et al., "Disjunctive inequalities: applications and extensions", *Wiley Encyclopedia of Operations Research and Management Science*, vol. 2, pp. 1441–1450, 2011.

[186] J. P. Vielma and G. L. Nemhauser, "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints", *Mathematical Programming*, vol. 128, no. 1, pp. 49–72, 2011.

[187] D. Avis and K. Fukuda, "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra", *Discrete & Computational Geometry*, vol. 8, no. 3, pp. 295–313, 1992.

[188] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.", *Journal of machine learning research*, vol. 9, no. 11, 2008.

[189] D. Noh, Y. Liu, F. Rafeedi, H. Nam, K. Gillespie, J.-s. Yi, T. Zhu, Q. Xu, and D. Hong, "Minimal degree of freedom dual-arm manipulation platform with coupling body joint for diverse cooking tasks", in *2020 17th International Conference on Ubiquitous Robots (UR)*. IEEE, 2020, pp. 225–232.

[190] X. Lin, F. Xu, A. Schperberg, and D. Hong, "Learning near-global-optimal strategies for hybrid non-convex model predictive control of single rigid body locomotion", *arXiv preprint arXiv:2207.07846*, 2022.

[191] Q. Cappart, D. Chételat, E. Khalil, A. Lodi, C. Morris, and P. Veličković, "Combinatorial optimization and reasoning with graph neural networks", *arXiv preprint arXiv:2102.09544*, 2021.

[192] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks", *Advances in Neural Information Processing Systems*, vol. 32, 2019.