Learning for Humanoid Multi-Contact Navigation Planning

by

Yu-Chi Lin

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in The University of Michigan
2020

Doctoral Committee:

Associate Professor Dmitry Berenson, Chair
Associate Professor Odest Chadwicke Jenkins
Associate Professor Matthew Johnson-Roberson
Associate Professor Ludovic Righetti, New York University

Yu-Chi Lin

linyuchi@umich.edu

ORCID iD: 0000-0001-5259-534X

For the freedom of mankind

ACKNOWLEDGEMENTS

First, I would like to thank Dmitry for giving me the opportunity to be a part of the Autonomous Robotic Manipulation Lab, and generously supporting me throughout my PhD study. I also would like to thank Ludovic for advice during and after my internship at Max Planck Institute. Thank you to my other committee members, Prof. Jenkins and Prof. Johnson-Roberson. Although we only had a few opportunities to meet, I am grateful for your suggestions.

Thanks to all my labmates. You have made all the difference in making my time in ARM lab a great experience. Special thanks to Dale, Andreas, Ruikun, Brad, Glen and Andrew for countless meaningful discussions which helped me in completing this dissertation. I also would like to thank other members of the lab for broadening my knowledge in robotics, software engineering, and many other subjects.

Thanks to my collaborators in Machine in Motion Lab in Max Planck Institute and New York University. I really enjoyed the 2018 summer time in Tuebingen with beer, BBQ and the world cup. I would like to thank Brahayam, Majid, Julian, Miroslav, and Andrea for our discussions about humanoid momentum optimizations and neural networks, which finally evolve into a major contribution of this dissertation.

Thanks to Robert, Jerry and other friends in IHMC to give me the opportunity to intern at the Robotics Lab in IHMC. Although I did not have enough time to make a contribution, it is an invaluable experience to me to finally see a working humanoid robot in close distance at the end of my PhD study.

Also thanks to Jessy, Ella, Denise, and all other professors, staffs and students in the Robotics Program. Thank you for building up such an amazing program. I really enjoy my time here, and appreciate all the support I got from the program and the student community.

Finally, thank you to my family, who always unconditionally support me. Without them, I would not have any chance to start this journey of PhD study, and I would not be able to overcome the self-doubt and setbacks along the path. With them, I know I will always have a home waiting for me to come back.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF ABBREVIATIONS

C-Space Configuration Space

DOF degree of freedom

PFS Planning from Scratch

RA Retrieve and Adapt

ANA* Anytime Non-parametric A*

CES Contact-consistent Elastic Strip

CoM center of mass

CoP center of pressure

ABSTRACT

Humanoids' abilities to navigate uneven terrain make them well-suited for disaster response efforts, but humanoid motion planning in unstructured environments remains a challenging problem. In this dissertation we focus on planning contact sequences for a humanoid robot navigating in large unstructured environments using multi-contact motion, including both foot and palm contacts. In particular, we address the two following questions: (1) How do we efficiently generate a feasible contact sequence? and (2) How do we efficiently generate contact sequences which lead to dynamically-robust motions?

For the first question, we propose a library-based method that retrieves motion plans from a library constructed offline, and adapts them with local trajectory optimization to generate the full motion plan from the start to the goal. This approach outperforms a conventional graph search contact planner when it is difficult to decide which contact is preferable with a simplified robot model and local environment information. We also propose a learning approach to estimate the difficulty to traverse a certain region based on the environment features. By integrating the two approaches, we propose a planning framework that uses graph search planner to find contact sequences around easy regions. When it is necessary to go through a difficult region, the framework switches to use the library-based method around the region to find a feasible contact sequence faster.

For the second question, we consider dynamic motions in contact planning. Most humanoid motion generators do not optimize the dynamic robustness of a contact sequence. By querying a learned model to predict the dynamic feasibility and robustness of each contact transition from a centroidal dynamics optimizer, the proposed planner efficiently finds contact sequences which lead to dynamically-robust motions. We also propose a learning-based footstep planner which takes external disturbances into account. The planner considers not only the poses of the planned contact sequence, but also alternative contacts near the planned contact sequence that can be used to recover from external disturbances. Neural networks are trained to efficiently predict multi-contact zero-step and one-step capturability, which allows the planner to generate contact sequences robust to external disturbances efficiently.

CHAPTER I

Introduction

Humanoid robots and other kinds of legged robots navigate by breaking and making contacts in the environment. Such motion only requires a small part of the environment being contacted by the robot, and can overcome terrains with large sudden changes in height, such as stairs. Moreover, compared to other legged robots with more legs, humanoid robots have the smallest projection area on the ground, which makes them more suitable to navigate through narrow spaces. Therefore, despite the difficulty of controlling humanoid robots, humanoid robots are still promising solutions for navigating in unstructured environment, such as a disaster site. In such situation, the robots are expected to operate in very complicated and unstructured environments with limited rescue time. Although a robot can be commanded by a human with remote control, with limited communication bandwidth, the robot still needs to plan its motion quickly. Therefore, this dissertation focuses on autonomous humanoid navigation planning in large unstructured environments (see Figure 1.1). Although the robot's sensing range may be limited to only a few meters, it is still important to construct a long-term navigation plan to ensure the robot can reach its goal. Such a plan can be constructed from a pre-generated map of the environment; e.g., using a drone to map the environment before the humanoid enters.

The most common formulation for robot motion planning is to plan in Configuration Space (C-Space). This formulation captures the robot kinematics, which allows a planner to easily check all the kinematic constraint including collision avoidance via rejection sampling. However, such a formulation is very inefficient for humanoid robots. First, humanoid robots generally have a high number of degree of freedom (DOF), which makes sampling configuration in the C-Space inefficient. Second, humanoid robots navigate by making contacts, so a feasible configuration is always limited to some lower-dimensional manifolds in C-Space. Each lower-dimensional manifold corresponds to a set of contacts the robot makes, also called a *stance* in

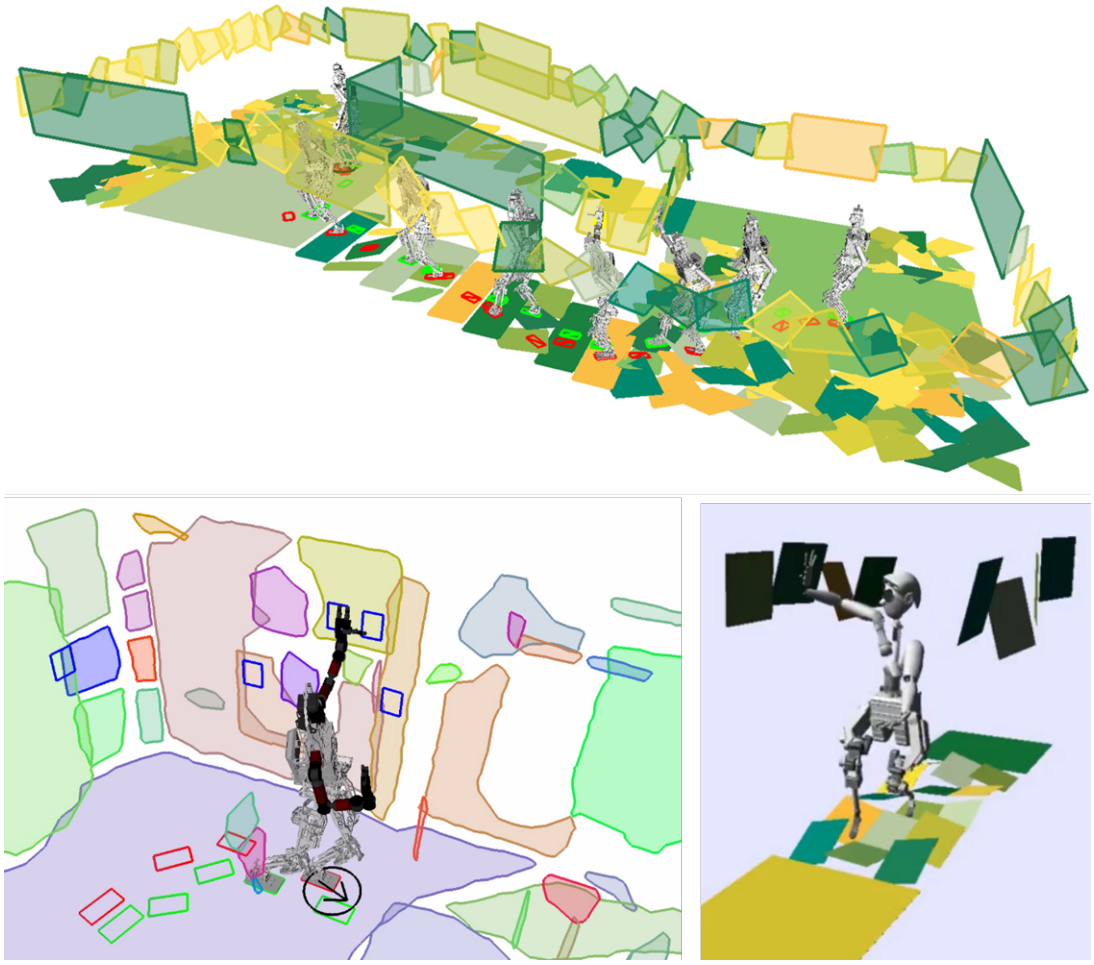


Figure 1.1: Examples of unstructured environments. Top: A large environment featuring rubble surfaces, and long stairways covered with debris. Bottom Left: Surfaces with irregular shape extracted from a ship environment. Bottom Right: A rubble corridor environment emulating a disaster scene.

this dissertation. Although projection methods can be applied to sample the configurations in lower-dimensional manifolds, the configuration samples are generally not in the same lower-dimensional manifold. Therefore, the robot cannot simply move between any two close configurations with straight lines in C-Space. Instead, the robot has to first determine if the two configurations are in the same manifold before moving from one configuration to the other.

To increase planning efficiency, many approaches (Kuffner et al., 2001; Chestnutt et al., 2003; Chung and Khatib, 2015; Ponton et al., 2016; Tonneau et al., 2018) decompose the problem into planning a sequence of lower-dimensional manifolds and then planning a trajectory within each lower-dimensional manifold. Each

lower-dimensional manifold corresponds to a set of contacts, so the planning of a lower-dimensional manifold sequence is equivalent to the planning of a contact sequence, which we call *contact planning* throughout this dissertation. This formulation can reduce the dimensionality of the problem if the robot has redundant manipulators. Furthermore, since all the C-Space path corresponding to a contact transition are represented by single edge in contact planning, the solution path in contact planning is much smaller in the number of edges than the one derived from directly planning in C-Space, which reduces the search space significantly. Finally, in each lower-dimensional manifold, the joint trajectories can be represented by some predefined motion primitive, such as a parabolic trajectory for the moving end-effector, and planning in C-Space can be avoided.

However, to plan contact sequence without C-Space still requires a simplified model which aims to capture the robot's kinematic and dynamic constraints, and this approach suffers from the trade-off between planning efficiency and accuracy of the simplified model. In structured environments (Kuffner et al., 2001; Chestnutt et al., 2003), a simplified model can easily be found to capture the robot's kinematic and dynamic constraint, but it is challenging to find a simplified model working in unstructured environments.

Therefore, we are interested in the contact planning of humanoid robot to traverse unstructured environments, represented as a set of contactable surfaces, as shown in Figure 1.1. In such environments, humanoid navigation can benefit greatly from the use of palm contacts for more robust balance and control, but it poses important computational challenges. First, adding palm contacts requires multiple non-coplanar contacts, which prevents the use of some popular simplified dynamics models, such as the linear inverted pendulum model, and the support polygon for balance checking. Second, in multi-contact motion scenarios, the robot can use any combination of the end-effectors, which increases the planning complexity, and emphasizes the need for a fast evaluation of contact pose feasibility. Finally, it is also difficult to consider dynamic robustness of multi-contact motions in contact planning because it requires expensive computation to solve optimization problems.

In this dissertation, we summarize the above challenges as the following two questions:

- How do we efficiently generate a feasible contact sequence?
- How do we efficiently generate contact sequences which lead to dynamically-robust motions?

The keys to both questions are the high dimensionality and branching factor in humanoid contact planning using multi-contact motions. Fortunately, we can utilize previous experience of the robot navigating in different environment and train models to help improve the efficiency of contact planning. Therefore, we propose multiple learning-based approaches to plan contact sequences more efficiently by sampling contact poses more intelligently with a learned heuristic and approximating expensive dynamic robustness evaluation with neural networks. We show that the proposed learning-based approaches significantly improves existing search-based contact planner in its efficiency and solution quality.

The rest of this dissertation is organized as follows. We introduce the related work in Chapter II. Chapter III briefly describes the baseline search-based contact planner, the Planning from Scratch (PFS) approach. The subsequent chapters feature the proposed learning approaches, and are intended to be mostly self-contained.

In Chapter III, to solve the contact planning problem, we first construct a baseline approach based on Kuffner et al. (2001) by formulating the problem as a graph search. We call this approach PFS. Given an initial stance and a goal region in the environment, PFS produces a sequence of contact transitions, including both palm and foot contacts, to move the robot from the start stance to the goal region. In its basic form, PFS aims to find a sequence of contact transitions with shorter end-effector traveling distance and fewer number of steps. By adjusting the edge cost definition in the graph search problem, PFS can produce solutions with different objectives.

In Chapter IV, we propose a humanoid robot navigation planning framework that reuses previous experience to decrease planning time. In the proposed framework, an experience-retrieval module is added in parallel to PFS. This module collects previously-generated motion plans and clusters them based on contact pose similarity to form a motion plan library. To retrieve an appropriate plan from the library for a given environment, the framework uses a distance between the contact poses in the plan and environment surfaces. Candidate plans are then modified with local trajectory optimization until a plan fitting the query environment is found.

In Chapter V, we build on the finding from Chapter IV that using library-based methods to help the planner solve difficult navigation planning problems requiring palm contacts, but such methods are not efficient when navigating an easy-to-traverse part of the environment. To maximize planning efficiency, we would like to use PFS when an area is easy to traverse and switch to the library-based method only when traversal becomes difficult. We present a method that 1) Plans a guiding torso path

which accounts for the difficulty of traversing the environment as predicted by learned regressors; and 2) Decomposes the guiding path into a set of segments, each of which is assigned a motion mode (i.e. a set of feet and hands to use) and a planning method. Easily-traversable segments are assigned to PFS, while other segments are assigned a library-based method that fits existing motion plans to the environment near the given segment.

In Chapter VI, we propose an approach to consider the dynamics of the robot motion in contact planning. Traditional contact planning approaches assume a quasi-static balance criterion to reduce the computational challenges of selecting a contact sequence in unstructured environment. However, this limits the applicability of the approach when dynamic motions are required, such as when walking down a steep slope or crossing a wide gap. In this work, we go beyond current approaches by learning a prediction of the dynamic evolution of the robot centroidal momenta, which can then be used for quickly generating dynamically-robust contact sequences.

In Chapter VII, we propose a footstep planner which takes external disturbances into considertation to generate robust contact sequences. Most existing contact planners only consider kinematic constraints, and few, including the contact planner proposed in Chapter VI, take dynamic constraints into consideration. However, the robot motion robustness depends on not only the existing contacts and centroidal momenta, but also how many alternative contacts around the robot that can used by the robot to recover from external disturbances. Therefore, We improve the contact planner proposed in Chapter VI by explicitly modeling the external disturbances, and finding contact sequences that maximize the robot's success rate to reach the goal without falling under a disturbance. To achieve this, we propose a learning approach to approximate the zero-step and one-step capturability during multi-contact motion, and use the capturability prediction to inform the contact planner. The result shows that the capturability estimate help generate contact sequences that are robust to external disturbances.

CHAPTER II

Related Work

2.1 Energy-based Foot Placement Selection

Energy-based contact placement selection has been widely adopted in gaited legged robot locomotion. These approaches assumes flat terrain, decide foot placement based on the kinetic energy of the simplified robot dynamics model. Hodgins and Raibert (1991) modeled legged robot as a spring-loaded inverted pendulum, and decide the foot placement by controlling the speed of the robot. Similar idea has been applied to many legged system including robots developed in Boston Dynamics. Pratt et al. (2006) proposed the capture point approach which decides the next foot placement based on the instantaneous robot momentum to keep the robot stable. This approach and its variation are widely adopt in humanoid community to develop walking controllers. Nguyen et al. (2020) interpolated the sampled offline-computed motion trajectories to achieve fast online computation of motion trajectories of different step length and width. However, these approaches cannot generalize to multi-contact motions without gaits and only consider terrain variations as execution errors which are dealt by the controller reactively.

2.2 Footstep Planning

To deal with more unstructured terrain, footstep planning approaches shift the computation of contact placement from the controller to the planner to make higher-level decision, such as colliison avoidance. Footstep planning for humanoid robot has been studied extensively Kuffner et al. (2001); Chestnutt et al. (2003); Michel et al. (2005); Baudouin et al. (2011); Hornung et al. (2012); Kanoun et al. (2009); Deits and Tedrake (2014). In these works, the planner plans a footstep sequence to avoid obstacles on the ground and remain inside the specified contact regions on a flat

or piecewise-flat ground. To increase the likelihood of success, they incorporate an approximation of robot balance and kinematic reachability into the contact transition model, and do not explicitly perform balance check online.

2.3 Contact Planning using Multi-contact Motion

There are works addressing contact planning in unstructured environment using both palm and foot contacts. Escande et al. (2009) used optimization to find contacts in the neighborhood of a "rough" trajectory. However, its planning time is prohibitively long. Hauser et al. (2006) proposed a humanoid robot planner that used learned motion primitives. Chung and Khatib (2015) combined discrete-search-based contact space planning with a local trajectory optimizer. They generated an initial trajectory only obeying the reachability constraint, and locally optimize it to be feasible. Tonneau et al. (2018) use a sampling-based planner to first plan the torso path based on the kinematic reachability heuristic using a sampling-based planner, and then plan contacts around the torso path. However, these approaches assumes quasi-static motions, and drops solutions involving dynamic motions.

2.4 Reuse Previous Motion Plans

Reusing pre-computed plans has been studied in computer animation (Lau and Kuffner, 2006) and trajectory optimization (Myung et al., 2007). However, using path libraries for high-dimensional humanoid locomotion planning with balance and collision constraints and both hand and foot contact has not yet been explored.

Robot motion libraries have been used to speed up motion planning in C-Space (Berenson et al., 2012). However, the distance metric of (Berenson et al., 2012) is not adequate in our context because it does not consider contact with the environment, which is key for humanoid robot navigation. Recently Coleman et al. (2015) improved on (Berenson et al., 2012) by storing the experience in a sparse roadmap spanner. However, humanoid navigation involves multiple contact switches, necessitating that the robot travel through manifolds of differing dimension, which cannot be done with this sparse roadmap spanner. Jetchev and Toussaint (2013) also proposed a motion plan library framework by learning the mapping between environments and plans. However, this approach may overlook plans that come from an environment which is not similar to the query environment, but are nevertheless a good fit.

2.5 Traversability Estimation

There has been work proposing traversability estimation algorithms for mobile robots (Suger et al., 2015; Cunningham et al., 2017; Shneier et al., 2008). These methods learn models to estimate the terrain types based on visual, range or thermal inertia sensor data. The goal is to avoid certain types of terrain which may cause the mobile robot to slip or be stuck. In our work, the traversability does not measure the effect of the texture of the terrains for navigation, instead, it measures the richness of the space for humanoid robot contact placement.

Researchers have investigated predicting traversability for quadruped robots (Chilian and Hirschmuller, 2009; Wermelinger et al., 2016). They computed traversability features such as slope, terrain roughness and step height from visual data. Those features are combined in a weighted-sum cost function, which guides the robot. In our approach, we not only capture features from the environment, but also use simulation to learn a model to predict the actual traversability of the robot in the environment.

2.6 Contact Planning Combined with Different Planners

There has been work addressing humanoid locomotion planning using different planners or action types. Grey et al. (2017) proposed a probabilistic planner to plan humanoid locomotion on flat ground with doorways and small obstacles on the ground. The planner saves computation by generating periodic footstep motions on open flat ground, and plans for whole-body motion only when an obstacle is close by. Dornbush et al. (2018) proposed an approach to plan with adaptive dimensionality. The planner plans for multiple tasks, such as walking or climbing a ladder, in a low-dimensional representation with multi-heuristic A*, and computes high dimensional plans for each task. While this work is promising for planning a sequence of tasks, it is not clear how well it can perform if the task involves acyclic motions that require fine planning for the contact placements, such as traversing rubble.

2.7 Contact Planning Involed with Dynamics Constraints

Approaches to synthesize dynamically feasible multi-contact motions have also been extensively studied (Herzog et al., 2016; Carpentier et al., 2016; Dai and Tedrake, 2016; Caron and Kheddar, 2016; Audren et al., 2014). However, it is not trivial to include planning of contact poses in these approaches because contacts planning in general involves discrete or non-convex constraints for the contact poses. Deits and

Tedrake (2014) addresses the non-convexity by decomposing the environment into a set of convex regions and approximating the rotation using piecewise affine functions. The problem is then formulated as a mixed integer convex program and solved to global optimality. Although Deits and Tedrake (2014) only uses foot contact, and does not consider dynamics, it points a direction to include contact planning in an optimization problem.

Extensions of (Deits and Tedrake, 2014) for dynamic planning of a contact sequences are proposed in Ibanez et al. (2014); Ponton et al. (2016), which extend (Deits and Tedrake, 2014) with the selection of contact timings or hand contacts respectively. More recent works (Aceituno-Cabezas et al., 2017, 2018) use the same concept to plan gait sequences for quadruped robots and produce dynamically robust motions. However, mixed-integer approaches scale poorly against the number of integer decision variables. For instance, their applicability is limited to online contact generation in environments with few convex terrain regions, and short planning horizons.

Fernbach et al. (2017) proposes a kinodynamic sampling-based contact planner to plan kino-dynamically feasible contact sequences. They use a simplified robot model to dynamically plan smooth CoM trajectories based on convex optimization and then search for kinematically feasible contact poses around it. It shows a unified planning framework to consider dynamics and kinematics constraints, but it suffers from long planning time. Fernbach et al. (2018) proposes an efficient dynamic feasibility check by conservatively reformulating the problem as a linear program. While the check guarantees to reject dynamically infeasible motions, they do not address dynamical robustness in the stability check. Kim et al. (2013) learns quadratic dynamics objective of humanoid walking motion, and applies this learned model to select steps in a search-based footstep planner. However, their dynamics model assumes flat contact, and does not consider palm contacts, which limits the applicability of the approach.

2.8 Capturability Analysis

Capturability analysis of linear inverted pendulum (LIP) model is first proposed by Koolen et al. (2012). Since then, it is widely used to determine footstep placement in planning and control of robot dynamic walking (Sugihara, 2009; Takenaka et al., 2009; Morisawa et al., 2012; Englsberger et al., 2015; Griffin et al., 2017). There are also works address the capturability analysis for more complex variable-height inverted pendulum (VHIP) model to account for the height changes of the CoM. Pratt

and Drakunov (2007); Ramos and Hauser (2015); Koolen et al. (2016) address the balance control of humanoid robot using VHIP model for planar motions. Caron et al. (2019) further extends it to consider 3D movements, and develops analytical tool to determine capturability in VHIP model. Del Prete et al. (2018) proposes efficient analytical tool to compute zero-step capturability for multi-contact configuration using centroidal dynamics model. However, it has strong assumptions on using zero angular momentum, and cannot generalize to use more steps.

CHAPTER III

Planning from Scratch: Humanoid Contact Planning by Solving a Graph Search Problem

3.1 Contact Planning Problem Statement

Given an environment represented as a set of contactable surfaces, we wish to output a feasible path from the start stance to a goal region in the workspace. The path is defined as a series of stances, and consecutive stances in the path differ by one foot or palm contact pose. When executing this path, the robot must obey balance and collision constraints at all times. We assume that the robot can generate sufficient torque to balance itself. We also assume the friction coefficients are given. Our goal is to compute a feasible path for the robot as quickly as possible.

3.2 Contact Planning by Solving a Graph Search Problem

In PFS, the contact planning problem includes both palm and foot contacts, and is formulated as a graph search problem. The state of the planner is defined as the stance of the robot, the set of the contact poses corresponding to each end-effector. The end-effectors should be on one of the surfaces, and free from collision with all the surfaces except for the contact surface. The robot should also be in static balance in every configuration.

An action in the planner is defined as the robot switching one of the end-effectors' contact poses. Given a state, the possible next actions are described as a pre-defined transition model relative to the current end-effector poses, as shown in Figure 3.1. We do not specify the order of end-effector transitions other than requiring that the same end-effector is not used in consecutive actions. To determine the next foot contact, we first project the standing foot contact pose to the XY plane along the global Z

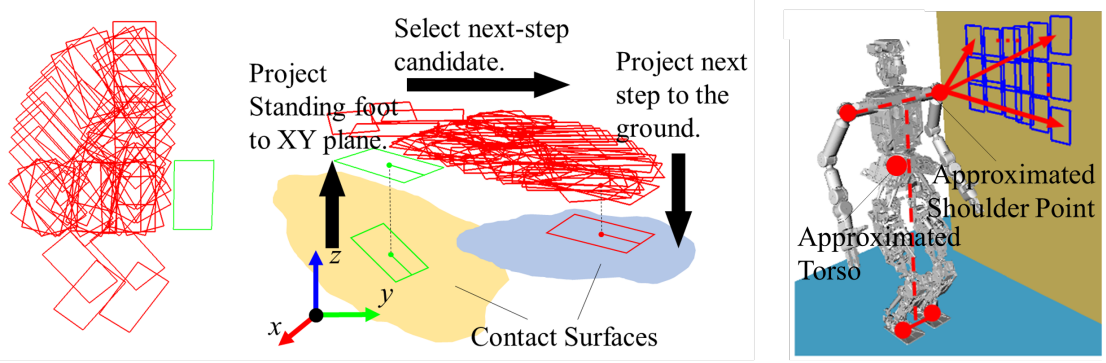


Figure 3.1: Left: Foot contact transition model (57 steps); Middle: The projections of foot contact to get the next step pose; Right: An example of palm contact transition model

axis, use the transition model to find the next step in the XY plane, and then project the pose to the ground to get the next foot contact pose, as shown in Figure 3.1.

For palm contacts, we first approximate the torso pose p_t based on the poses of the feet with the following equations:

$$p_t = \left[\frac{x_{lf}+x_{rf}}{2} \quad \frac{y_{lf}+y_{rf}}{2} \quad \frac{z_{lf}+z_{rf}}{2} + z_{\text{offset}} \quad 0 \quad 0 \quad \frac{\theta_{lf}+\theta_{rf}}{2}\right]^T \tag{3.1}$$

where $[x_{lf}, y_{lf}, z_{lf}]$ and $[x_{rf}, y_{rf}, z_{rf}]$ are the left and right foot positions, respectively, θ_{lf} and θ_{rf} are the rotations of each foot about the z axis, and z_{offset} represents the nominal body height relative to the feet. Given the approximated torso pose in each state, we can derive the approximated shoulder points of the robot. The potential palm contacts are then computed by ray-casting from the approximated shoulder points, as shown in Figure 3.1.

For each edge from state s to s', the edge cost $\Delta g(s, s')$ is defined as:

$$\Delta g(s, s') = d_e(s, s') + w_\theta d_\theta(s, s') + w_s \tag{3.2}$$

where d_e is the translation of the moving end-effector, d_θ is the difference in robot orientation (defined as the mean of the two feet's rotation about the Z axis), and w_θ and w_s are the weight for robot orientation difference and the step cost, respectively. With this formulation, the contact planner will try to find shorter path by reducing the total distance the end-effectors travel, and avoid changing direction. Adding step cost w_s helps reduce the number of steps used in the plan. The heuristic for each

state s used in the planner is:

$$h(s) = w_\theta h_\theta(s) + \sum_{i}^{|end\text{-}effectors|} h_{e,i}(s) + w_s \frac{h_{e,i}(s)}{d_{e,i,max}} \tag{3.3}$$

where h_θ is the difference between the current and goal robot orientations, $h_{e,i}$ is the Euclidean distance between the pose of end-effector i and the goal, and $d_{e,i,max}$ is the maximum possible translation for end-effector i in one action. The above formulation of $\Delta g(s, s')$ and $h(s)$ focuses on reducing the number of steps and the total traveling distance of the end-effectors. In Chapter V and VI, we will modify the definition of $\Delta g(s, s')$ and $h(s)$ to consider other objectives.

In PFS, we solve the contact planning problem with Anytime Non-parametric A* (ANA*). To traverse through unstructured environment, the planner needs a fine discretization of possible contact pose transitions, as shown in Figure 3.1. However, this would significantly increase the branching factor, and could have a case where multiple contact transitions are similar in edge cost, which cause a conventional A* planner to waste time on evaluating similar contact transitions. To deal with this problem, we solve the contact planning problem in PFS using ANA*. ANA* is an anytime planning algorithm, which performs depth-first search in the beginning to quickly find a feasible solution, and then refines the solution overtime once the first one is found. This is achieved by exploring the state with highest priority function value$e(s)$ (Lines 1 to 3 and 8 to 10 in Algorithm 1). Initially, when G, the total cost of the current best solution, is high, $g(s)$ has little impact on $e(s)$. Therefore, the algorithm will favor states with lower $h(s)$, which behaves like a depth-first search. ANA* updates G when a solution is found (Line 12 in Algorithm 1). As G goes down, $g(s)$ becomes more important in $e(s)$, which shifts the searching behavior from depth-first search gradually toward breadth-first search. In this way, the planner can avoid early comparison between similar contact transitions, and reduce the time required to find a feasible solution.

3.3 State Feasibility Check

The contact planner targets planning in unstructured environments. When expanding the search tree, the discretized foot and palm contacts are projected to environment surfaces to get the contact poses. Therefore, we must decide an state's feasibility online.

When considering each candidate contact pose, the planner checks if the contact

end-effector link is free from collision with all the surfaces except for the contact surface, and reject contact poses that does not pass this check. Besides obeying collision constraints, the robot must be able to reach the specified contact poses and maintain balance when moving from the parent state to the current state. A typical approach to address reachability in contact planning is to derive a contact transition model in which all possible moves are within a conservative bound of reachability. Since we expect the robot to navigate in an unstructured environment, such a boundary is hard to derive without sacrificing significant reachability. To better describe the robot's reachability, we filter out impossible transitions with a loose bound based on the length of the manipulators, and then use Jacobian-based IK to directly check reachability. Joint limit and self-collision constraints are also checked.

To ensure the robot remains in balance, we use the method described in Caron et al. (2015) as the balance checker which is treated as a constraint in the inverse kinematics solver. To speed up the process, we approximate the balance check for the entire transition by checking two critical configurations: the beginning of the contact transition where the moving end-effector has just broken contact and the end of the contact transition where the moving end-effector is about to make contact.

Algorithm 1: The Anytime Nonparametric A* algorithm (Reproduced from van den Berg et al. (2011))

```
1  e(s):
2      return (G − g(s)) / h(s)
3
4  ImproveSolution():
5      while OPEN ≠ ∅ do
6          s ← argmax_{s∈OPEN} {e(s)}
7          OPEN ← OPEN \ {s}
8          if e(s) < E then
9              E ← e(s)
10         end
11         if IsGoal(s) then
12             G ← g(s)
13             return
14         end
15         foreach successor s' of s do
16             if g(s) + Δg(s, s') < g(s') then
17                 g(s') ← g(s) + Δg(s, s')
18                 predecessor(s') ← s
19                 if g(s') + h(s') < G then
20                     Insert of update s' in OPEN with key e(s')
21                 end
22             end
23         end
24     end
25
26 ANA*():
27     G ← ∞; E ← ∞; OPEN ← ∅; ∀s : g(s) ← ∞; g(s_start) ← 0
28     Insert s_start into OPEN with key e(s_start)
29     while OPEN ≠ ∅ do
30         ImproveSolution()
31         Report current E-suboptimal solution
32         Update keys e(s) in OPEN and prune if g(s) + h(s) ≥ G
33     end
```

CHAPTER IV

Retrieve and Adapt Previously Generated Motion Plan

4.1 Introduction

In Chapter III, we describe the formulation of the contact planning problem as a graph search problem, and solved with ANA*. Although ANA* could speed up the search by compromising on optimality and find a path quickly in a contact-rich environment, it is difficult to compute which state is closer to the goal in the search tree with a heuristic function when the environment is difficult to traverse. As a result PFS may become stuck in a cul-de-sac, i.e. a region with scarce contacts. The approach proposed in this chapter adds an experience-retrieval module in parallel to PFS, which reduces the planning time by reusing previous experience.

In this work, there are two modules running in parallel: the PFS module and the Retrieve and Adapt (RA) module, as shown in Figure 4.1. The PFS module first plans a contact sequence without considering collision with the environment (except for the end-effector links), as described in Chapter III. The resulting contact sequence is then interpolated, inverse kinematics is computed, and the entire sequence of configurations is optimized locally to avoid obstacles. We call the output joint trajectory and the corresponding contact sequence of this process a "motion plan."

RA, on the other hand, provides solutions by retrieving motion plans from a library. RA stores and clusters motion plans generated by the framework to form a motion plan library based on the contact poses of each plan. Given a new environment, RA queries the library to find an appropriate plan based on its contact poses and modifies it to fit the environment. Both modules start planning simultaneously and the one that finishes first stops the other one. Finally, the generated motion plan is added to the library if it differs significantly from other plans in the library.

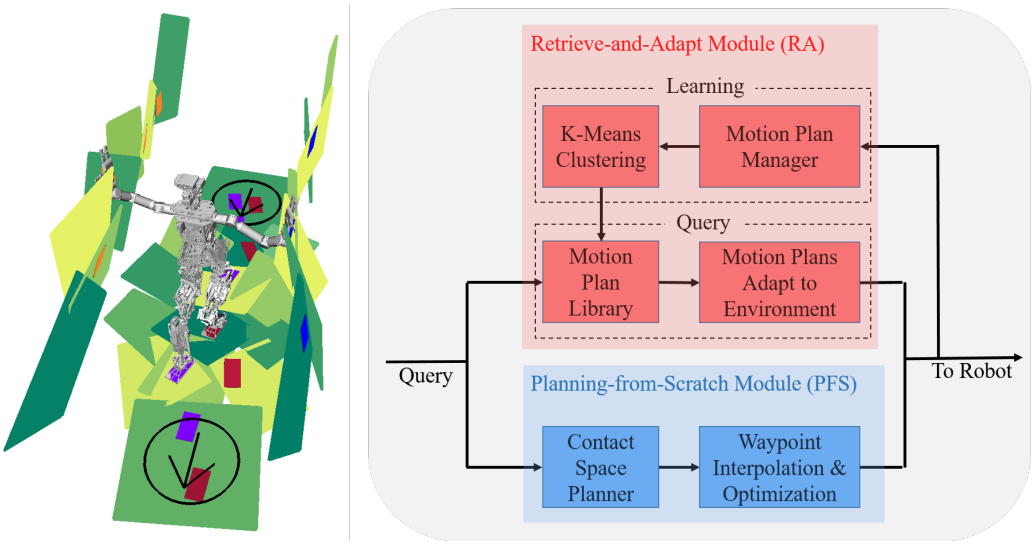


Figure 4.1: Left: A humanoid follows a planned sequence of contact poses to navigate in a complex unstructured environment modeled as a set of contact regions. Right: The structure of the proposed framework

The main contributions of this work are: (1) A framework for building humanoid robot motion plan libraries based on contact pose sequences; (2) A fast distance function for retrieving feasible plans from the library for a new environment.

Our experiments show that the proposed framework achieves a higher success rate in unstructured environments compared to planning-from-scratch. Additionally, the framework is agnostic to the PFS module, so new developments in navigation planning can be integrated easily.

4.2 Problem Statement

We address the humanoid navigation planning problem. Given an environment represented as a set of contactable surfaces, we wish to output a feasible trajectory from the start configuration to a goal region, defined as an area the feet must be within. We are interested in using the hands to help balance the robot against potential disturbances. Therefore, a feasible path should have at least three end-effectors in contact at any time, and must obey balance and collision constraints. We assume (as in Caron et al. (2015)) that the robot can generate sufficient torque to balance itself. We also assume the friction coefficients are known.

4.3 The PFS Module

The PFS module first plans a sequence of contacts using ANA* and then uses inverse kinematics and CES to construct a feasible trajectory from the contacts. The PFS contact planner follows the definition in Chapter III.

To obtain the final robot trajectory, the contact sequence returned by the contact planner is interpolated with parabolic trajectories for each contact transition. IK is computed for the interpolated poses and the sequence of resulting configurations is then optimized with the CES algorithm to avoid obstacles in the environment.

4.4 Learning Part of the RA Module

To efficiently retrieve a feasible motion plan in a new environment, the RA module needs to identify promising plans in the library quickly. This is achieved by clustering the motion plans. Motion plans inside each cluster are represented by a cluster representative, so that the RA module can find promising motion plans by checking these cluster representatives instead of the entire library. We denote the motion plan library as L, which can also be represented as K clusters of motion plans: $L = [C_1, C_2, ..., C_K]$.

Each new motion plan P_{new} generated by the proposed framework is first examined by the motion plan manager. If the motion plan's distance to other motion plans in the library is above a user-defined threshold d_{min}, it will be added to the library. The algorithm used in the learning part of the RA module is shown in Algorithm 2.

4.4.1 Motion Plan Feature Extraction

To find promising motion plans, the RA module should measure how close the contacts of the motion plan are to the surfaces in the query environment. The set of contact poses $\mathcal{C}(\pi)$ is extracted from each motion plan π:

$$\mathcal{C}(\pi) = \{\langle \mathbf{c}_k, e_k \rangle \,|\mathbf{c_k} = \langle \mathbf{X_k}, \mathbf{Q_k} \rangle \in SE(3); k = 1, 2, ..., N_c\} \tag{4.1}$$

where N_c is the number of contact poses, e_k is an index which indicates the corresponding end-effector, and $\mathbf{X_k}$ and $\mathbf{Q_k}$ are the translation vector and the rotation quaternion, respectively. As suggested by Kuffner (2004), the distance between two

Algorithm 2: Learning Part of the RA Module

```
close_to_existing_motion_plan ← False;
for i in 1 to K do
    C_i ← L[i];
    for j in 1 to |C_i| do
        π_{i,j} ← C_i[j];
        if d(π_{i,j}, π_new) < d_min then
            close_to_existing_motion_plan ← True;
        end
    end
end
if not close_to_existing_motion_plan then
    L ← L ∪ π_new;
    K ← 1;
    do
        [C_1, C_2, ..., C_K] ← K-means(L, K);
        L' ← [C_1, C_2, ..., C_K];
        [d_{C_1}, d_{C_2}, ..., d_{C_K}] ← Get_In-Cluster_Dist(L');
        d_C ← max([d_{C_1}, d_{C_2}, ..., d_{C_K}]);
        K ← K + 1;
    while d_C > d_max;
    L ← L';
end
return L;
```

contact poses is:

$$\eta(\langle \mathbf{c}_i, e_i \rangle, \langle \mathbf{c_j}, e_j \rangle) = \begin{cases} |\mathbf{X_i} - \mathbf{X_j}| + w_r \cdot (1 - |\mathbf{Q_i} \cdot \mathbf{Q_j}|), w_r > 0 & , e_i = e_j \\ \infty & , e_i \neq e_j \end{cases} \tag{4.2}$$

To calculate the distance between motion plans, the motion plans need to be aligned. We define the start and goal point of the motion plan as the mean position of the feet in the first and last configurations of the trajectory. We then align the start points of the two plans and subsequently rotate the $\mathcal{C}(\pi)$ of one plan about the global Z axis to align the start and the goal points of both motion plans on the same line.

The distance between a pair of motion plans π_1 and π_2 is defined as the Hausdorff

Algorithm 3: Query part of the RA Module

```
L_sorted ← Env_Match_and_Sort(L) ;
for i in 1 to K do
    C_i ← L_sorted[i];
    C_{i,sorted} ← Env_Match_and_Sort(C_i);
    for j in 1 to |C_{i,sorted}| do
        π_j ← C_{i,sorted}[j];
        π_optimized ← CES(π_j);
        if π_optimized is feasible then
            return π_optimized;
        end
    end
end
return Failure
```

distance of the between their sets of contact poses:

$$d(\pi_1, \pi_2) = max\{ \sup_{\langle \mathbf{c}_i, e_i \rangle} \inf_{\langle \mathbf{c_j}, e_j \rangle} \eta(\langle \mathbf{c}_i, e_i \rangle, \langle \mathbf{c_j}, e_j \rangle), \sup_{\langle \mathbf{c_j}, e_j \rangle} \inf_{\langle \mathbf{c}_i, e_i \rangle} \eta(\langle \mathbf{c}_i, e_i \rangle, \langle \mathbf{c_j}, e_j \rangle)\} \tag{4.3}$$

where $\langle \mathbf{c}_i, e_i \rangle \in \mathcal{C}(\pi_1)$ and $\langle \mathbf{c}_j, e_j \rangle \in \mathcal{C}(\pi_2)$. Hausdorff distance allows comparisons between motion plans with different numbers of contacts, automatically separating motion plans with different lengths. The drawback of Hausdorff distance is its sensitivity to outlying data. However, it is not an issue in our context because the contact poses are bounded by the reachability and balance constraints.

4.4.2 K-Means Clustering

The motion plans in the library are clustered with the K-means algorithm using the Hausdorff distance described in Eq. 4.3. K is determined by running K-means with iteratively increasing K until the maximum distance between every motion plan pair in each cluster is below a user-defined bound d_{max}. Lowering the bound can increase the similarity in each cluster, but it also increases the number of clusters and lengthens the time to evaluate all clusters.

In each cluster the plan with the minimum sum-of-squared distance to every other motion plan is selected as the cluster representative. The cluster representative is used to estimate how well the plans in this cluster fit the query environment.

4.5 Query Part of the RA Module

In the query part, the input is a combination of a goal and a query environment modeled as a set of contact regions. The objective is to generate a feasible motion plan as soon as possible. The RA module first calculates the distance of each cluster representative to the query environment. The clusters are then sorted by the distance, and searched in that order. The plans in the searched cluster are also sorted in the same manner, and then deformed by the Contact-consistent Elastic Strip (CES) algorithm (Chung and Khatib, 2015) to adapt to the query environment one-by-one until a feasible motion plan is found (see Algorithm 3).

4.5.1 Contact Region Extraction

The environment can be viewed as a union of possible contact poses for each end-effector. In this representation, the distance between any end-effector pose to the environment is simply the distance between the end-effector pose and the nearest contact pose in the environment. Therefore, it is important to convert the environment into the contact pose union representation in order to define the distance between a motion plan and an environment.

We adopt the idea in (Chung and Khatib, 2015) to express the environment as a union of circular contact regions. We sample multiple circle origins with a pre-defined density, and sequentially grow circular regions from samples not covered by other regions. The circular regions aim to cover all possible contact poses of the environment. If the radius of a region is smaller than the density, the process will iteratively increase the sampling density in its neighborhood until reaching a density bound $d_{cr,min}$. We denote the set of contact regions as CR. Since the contact region does not consider rotation about the contact normal, this representation is a conservative estimation of the available contact poses in the environment, as shown in Figure 4.2, but it can be generated automatically. Note that end-effector size is considered in the generation of contact regions. The start and goal regions are modeled as circles centered at the specified start/goal position and bounded by the nearest obstacles or surface boundaries.

4.5.2 Environment to Motion Plan Cluster Matching

In order to find a promising plan to navigate through the query environment, we define a distance between a motion plan and an environment. Based on the assumption that a motion plan is more likely to be modified to become feasible in the

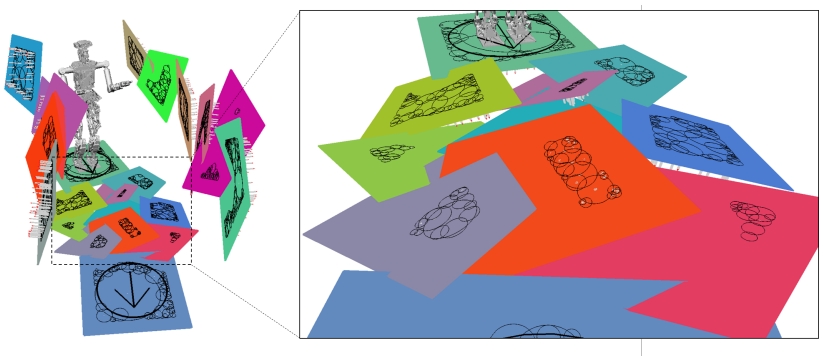


Figure 4.2: Contact region sampling

environment if its contact poses are closer to the contact regions in the environment, we match a motion plan to an environment based on the distance between contact poses in the motion plan and the contact regions in the environment. This is done in the *Env_Match_and_Sort* function in Algorithm 3 as follows:

We define a contact region's frame by aligning the Z axis to the contact region normal, and X axis to an arbitrary vector on the surface. The distance between a pose $\mathbf{c}$ and a contact region $cr \in CR$ is defined as:

$$\begin{aligned} \xi(\mathbf{c}, cr) &= \sqrt{d_{xy}^2 + d_z^2} + w_r d_{ori} \\ d_{xy} &= max\left(0, |(x'_c, y'_c)| - r_{cr}\right), \; d_z = |z'_c| \\ d_{ori} &= 1 - \mathbf{n}'_c \cdot [0, 0, 1]^T \end{aligned} \tag{4.4}$$

where (x'_c, y'_c, z'_c) and $\mathbf{n}'_\mathbf{c}$ are the contact position and normal in the contact region frame, $w_r \in \mathbb{R}^+$ is a weighting factor, and r_{cr} is the radius of the contact region. Furthermore, we can define the projection of the contact pose to the contact region by shifting the contact pose to the closest point inside the contact region, and rotate the pose to align the contact pose normal to the contact region normal, as shown in Figure 4.3.

Since a motion plan starts and stops inside circular regions, there exist an infinite number of alignments of the plan before deformation of the contact poses. If we treat the whole contact series of a motion plan as a rigid body with 4 degrees of freedom: translation in the X, Y and Z directions, and rotation about the Z axis, expressed as $(x_{rp}, y_{rp}, z_{rp}, \theta_{rp})$, the initialization problem is then to find a transform of the entire plan that minimizes the distance between the plan and the environment. We call this representation of a plan as a rigid body a *rigid plan*. Finding a globally-optimal alignment of the rigid plan is costly so we find a local solution using a Jacobian-based

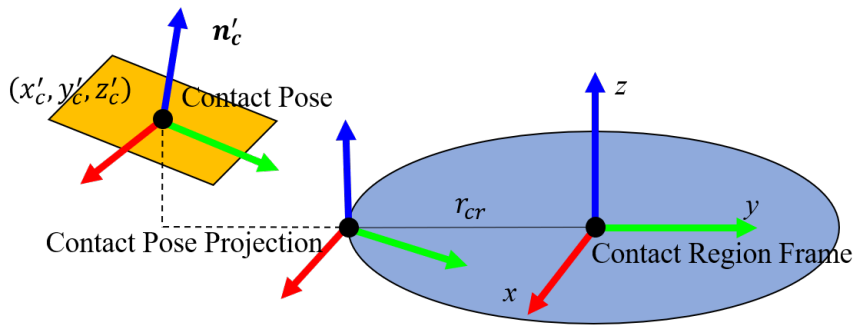


Figure 4.3: Contact pose vs. contact region distance.

approach. This approach "snaps" the rigid plan to the nearest set of contact surfaces.

Given a query environment, the start region is at (x_s, y_s, z_s) with radius r_s and the goal region is at (x_g, y_g, z_g) with radius r_g. The distance between the start and goal poses is l_{sg}, and the distance between the first and last poses of the rigid plan is l_{rp}. The algorithm initializes the rigid plan pose $\mathbf{T}_{rp} = (x_{0,rp}, y_{0,rp}, z_{0,rp}, \theta_{0,rp})$ as:

$$\begin{aligned} x_{0,rp} &= x_s + (l_{sg} - l_{rp}) \frac{r_s}{r_s + r_g} \frac{|x_g - x_s|}{l_{sg}} \\ y_{0,rp} &= y_s + (l_{sg} - l_{rp}) \frac{r_s}{r_s + r_g} \frac{|y_g - y_s|}{l_{sg}} \\ z_{0,rp} &= (z_s + z_g)\,/2 \\ \theta_{0,rp} &= atan2(y_g - y_s, x_g - x_s) \end{aligned} \tag{4.5}$$

This initialization guarantees the rigid plan's first and last poses will be inside the start and goal regions, respectively, if $l_{sg} - r_s - r_g \leq l_{rp} \leq l_{sg} + r_s + r_g$.

After initialization, we iteratively update $\mathbf{T}_{rp}$ to move the rigid plan's $\mathcal{C}(\pi)$ closer to their nearest contact regions. At each iteration, we find $cr_{min,i}$, the closest contact region to $\langle \mathbf{c}_i, e_i \rangle \in \mathcal{C}(\pi)$. To ensure that the motion plan connects the start and the goal, the foot poses of the start and the goal configurations are matched to the start and the goal regions, respectively. Jacobian J_i relates $\dot{\mathbf{T}}_{rp}$, the change in the rigid plan pose, to $\dot{\mathbf{c}}_i$, the desired change in the pose of contact $\mathbf{c}_i$. We can then combine the Jacobians for all $\mathbf{c}_i$:

$$\begin{bmatrix} \dot{\mathbf{c}}_1 \\ \dot{\mathbf{c}}_2 \\ \vdots \\ \dot{\mathbf{c}}_{N_c} \end{bmatrix} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_{N_c} \end{bmatrix} \dot{\mathbf{T}}_{rp} = J \dot{\mathbf{T}}_{rp} \tag{4.6}$$

We then use the pseudo-inverse J^+ to arrive at a $\dot{\mathbf{T}}_{rp}$ that takes into account the desired motion of all $\mathbf{c}_i$:

$$\dot{\mathbf{T}}_{rp} = J^+ \left[\dot{\mathbf{c}_1}^T, \quad \dot{\mathbf{c}_2}^T, \quad \ldots, \quad \dot{\mathbf{c}_{N_c}}^T \right]^T \tag{4.7}$$

The rigid plan pose will converge to a local minimum $\mathbf{T}'_{rp}$ through iterative application of Eq. 4.7. The distance between a rigid plan and the query environment is then defined as:

$$\Xi(\mathcal{C}(\pi), CR) = \frac{1}{N_c} \sum_{i=1}^{N_c} \xi \left(\mathbf{c}'_i, cr_{min,i} \right) \tag{4.8}$$

where $\mathbf{c}'_i$ is the ith contact pose in $\mathcal{C}(\pi)$ transformed by $\mathbf{T}'_{rp}$. The clusters are sorted by this distance, and searched in this order. Motion plans inside the searched cluster are also sorted in the same manner.

4.5.3 Local Trajectory Optimization

The motion plan, expressed as a sequence of configurations, is modified and optimized to fit the query environment with CES (Chung and Khatib, 2015). Each configuration of the trajectory will move its contact toward the nearest contact region, remain balanced, and avoid obstacles simultaneously. Although each contact pose converges to the nearest contact region according to the contact constraint, the contact pose is also affected by balance constraints and collision avoidance during each iteration. Therefore, contact poses may not end in the initial nearest contact region. In our setup the task priority of CES was (1) obey joint limits; (2) three tasks in parallel: maintain contact, remain in balance, avoid collision; and (3) "internal forces" used to smooth the trajectory, as described in Brock and Khatib (2002).

4.6 Experiments and Results

We test on the ESCHER humanoid robot. ESCHER had 33 DOF in our setup: two 7-DOF arms, two 6-DOF legs, one waist joint, and a 6-DOF base transform. The robot is to be in contact with at least 3 of its manipulators at any given time. We implemented our algorithms and test examples in OpenRAVE (Diankov, 2010) and also tested in the Gazebo physics simulator (Koenig and Howard, 2004). All experiments were run on an Intel Core i7-4790K 4.40 GHz CPU with 16GB RAM. The values of the parameters used in the experiments are the following: $w_\theta = 0.3\text{m/rad}, w_s = 10\text{m}, d_{cr,min} = 0.01\text{m}, w_r = 0.5, d_{min} = 0.1, d_{max} = 0.5$. Time

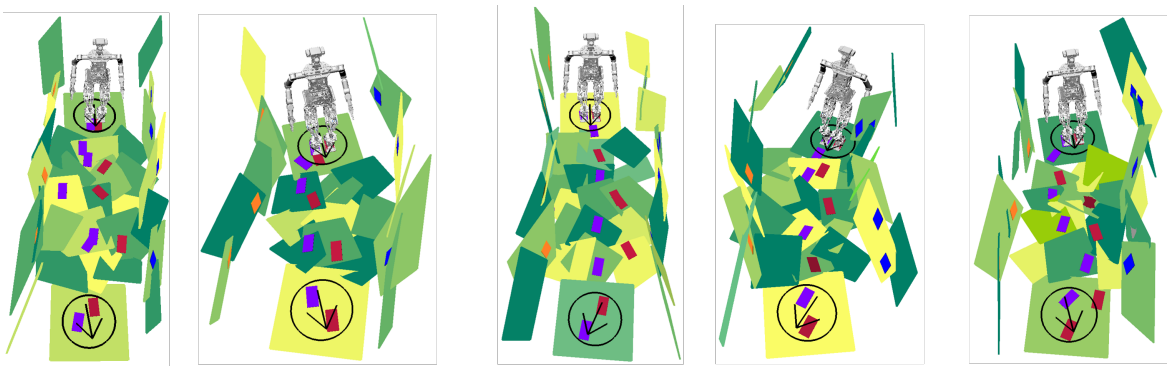


Figure 4.4: Examples of plans in rubble-like environments. Planned contacts for left foot (red), purple (right foot), blue (left palm), and orange (right palm).

limit for each trial is 5 minutes.

4.6.1 Random Surface Environment Test

We set up complex random surface test environments. We generate the environments with randomly tilted quadrilateral surfaces, as shown in Figure 4.4. Each environment is between 2m and 4m long. The surfaces may cover or intersect with each other, and leave part of the surface non-contactable, which causes the environment to be very challenging. Since this environment is extremely complex, the recall rate of the reachability and balance databases in these test environments are 10.3% and 30.2%, respectively with 1.5 million entries in the database. This does not provide a significant improvement in planning time, further motivating the need for the RA module.

To evaluate the proposed framework, we generated 100 random surface environments, and record the performance of PFS alone (the baseline) vs. the proposed framework with different sizes of motion plan libraries. If a trial's runtime exceeds 5 minutes, it is counted as a failure. For the PFS module, the failure cases also include optimization failure: the case when the local optimization after contact planning cannot find a feasible trajectory. For the RA module, the case when no feasible motion plan can be found in the library is counted as a failure. Examples of the test environments and plans generated by our framework are shown in Figure 4.4.

Figure 4.5 shows the success rate of the proposed framework with different library sizes. Even with a small library size, the proposed framework significantly improves the success rate. One of the major reason is that CES used in the RA module can shift contact poses in continuous space to arrive at small contact regions. However, PFS

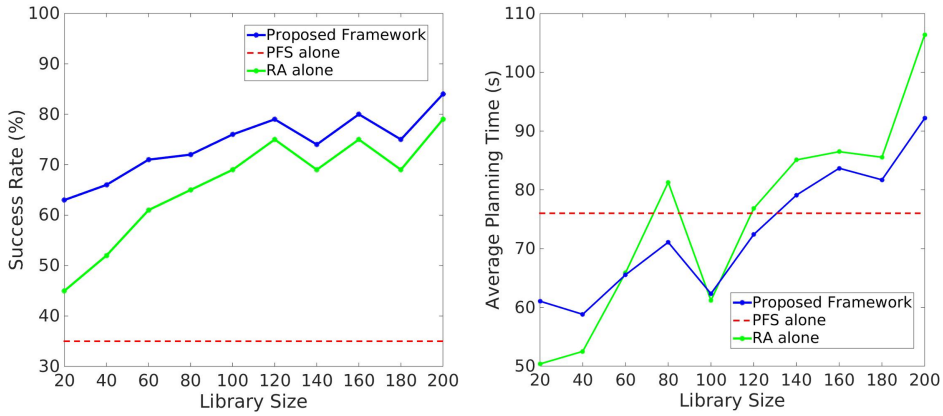


Figure 4.5: Left: Success rate and Right: Average planning time of **successful trials** for the PFS module, the RA module and the proposed framework with different sizes of libraries

may not find feasible next contacts in the transition model, and needs redundant contacts in order to adjust the standing foot to find feasible contacts at the next transition. Those redundant contacts increase the search depth and the planning time.

Furthermore, to navigate in such a complex environment, PFS requires a large transition model that densely discretizes the reachable space of the end-effectors. This entails a higher branching factor. When the heuristic is not accurate, this high branching factor slows down the planner.

Figure 4.5 shows the average planning time of the **successful trials**. Although the RA module takes more time to find a solution with a larger library, the increase in planning time of the successful trials is partly because the RA module with a large library can find solutions in difficult cases which cannot be solved within the time limit using a small library. For a library with 20 entries, RA outperforms the PFS success rate by 10%, and the combined framework outperforms PFS by 28%. For a library with 200 entries, RA outperforms the PFS success rate by 44%, and the combined framework outperforms PFS by 49%.

In Figure 4.6, we can observe that the number of trials in which the RA module finishes first increases as the size of the motion plan library grows. However, the trend saturates after the size exceeds 100. This is a mixture of two effects: (1) The RA module can solve more problems with a larger library. (2) The RA module requires more time to find a feasible motion plan in a larger library. This effect can be observed in Figure 4.6 as the successful and timeout cases both increase with a larger library.

	Plan found	Opt. Fail	Time out	Lib. Exhaust
PFS	35	21	44	-
RA(20)	45	-	0	55
RA(40)	52	-	1	47
RA(60)	61	-	2	37
RA(80)	65	-	6	29
RA(100)	69	-	6	25
RA(120)	75	-	4	21
RA(140)	69	-	10	21
RA(160)	75	-	7	18
RA(180)	69	-	11	20
RA(200)	79	-	10	11

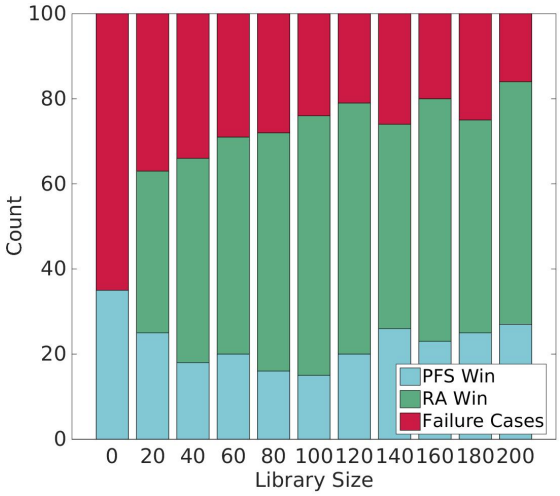


Figure 4.6: Left: Results with different library sizes; Right: Number of trials in which PFS or RA finishes first for different library sizes.

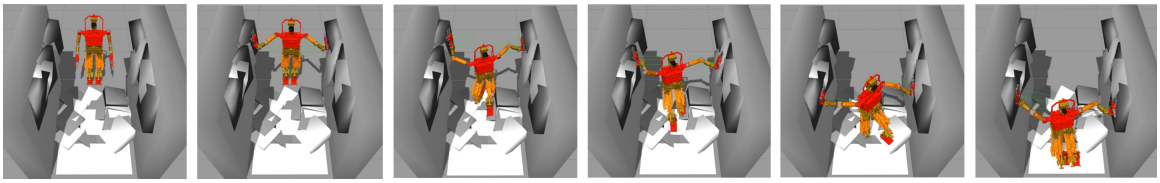


Figure 4.7: Gazebo simulation of robot navigating through a rubble-like environment.

4.6.2 Testing in Physics Simulation

To verify the feasibility of trajectories produced by the framework, we executed the plans in a random surface environment in the Gazebo simulator. The robot can walk through the "rubble" while using palm contacts for balance, as shown in Figure 4.7 and the attached video.

4.7 Conclusion

In this work, we proposed a humanoid navigation planning framework running two modules in parallel: Planning from Scratch (PFS) and Retrieve and Adapt (RA). PFS is a discrete-search-based contact planner. RA stores and clusters previously generated motion plans based on the Hausdorff distance of the contact poses. When the robot encounters a new environment, the module matches each cluster representative to the environment, and searches motion plan clusters based on the distance

between the motion plan cluster representative and the environment. Each plan in the searched cluster is then sorted by distance to the environment and then modified by CES algorithm to fit the environment until a valid plan is found. The results show that the proposed framework outperforms the baseline planning-from-scratch algorithm in success rate in difficult unstructured environments.

CHAPTER V

Humanoid Contact Planning in Large Unstructured Environments Using Traversability-Based Segmentation

5.1 Introduction

Disaster response is an important potential application for humanoid robots because of their abilities to navigate stairs and uneven terrain, such as rubble. This work focuses on constructing navigation plans for a humanoid in such large unstructured environments (see Figure 5.1). Even though the robot's sensor range may be limited to only a few meters, it is still important to construct a long-term navigation plan to ensure the robot can reach its goal. Such a plan can be constructed from a pre-generated map of the environment; e.g., using a drone to map the environment before the humanoid enters.

In such environments, humanoid navigation can benefit greatly from the use of palm contacts. Palm contacts provide additional support to allow the robot to make larger steps to avoid obstacles, cross gaps, or help with balance. However, considering palm contact in graph-search navigation planning algorithms (Kuffner et al., 2001; Chestnutt et al., 2003; Michel et al., 2005) greatly increases the branching factor of the search, resulting in impractical planning times for large environments. The planning is also difficult because palm contacts may not be available in all locations and sometimes they may be unnecessary, so the robot needs to decide when and where to use its palms. In previous work we explored using library-based methods to address difficult navigation planning problems requiring palm contacts Lin and Berenson (2016), but such methods are not efficient when navigating an easy-to-traverse part of the environment. To maximize efficiency, we would like to use graph-search, PFS, to traverse easy areas and switch to the library-based method, RA, when

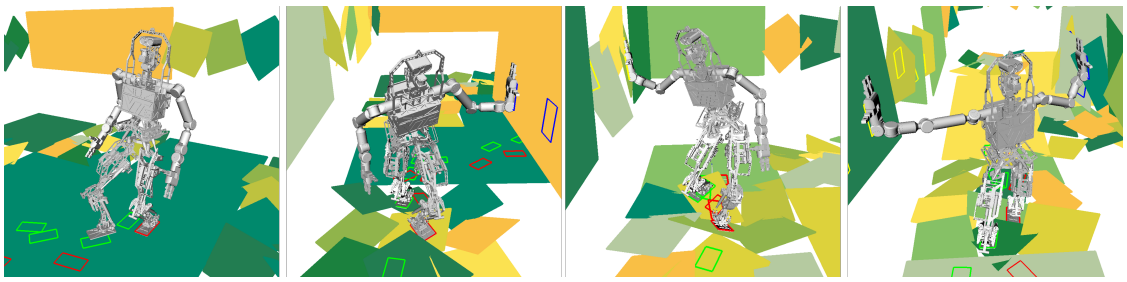


Figure 5.1: Using different motion modes to traverse unstructured environments.

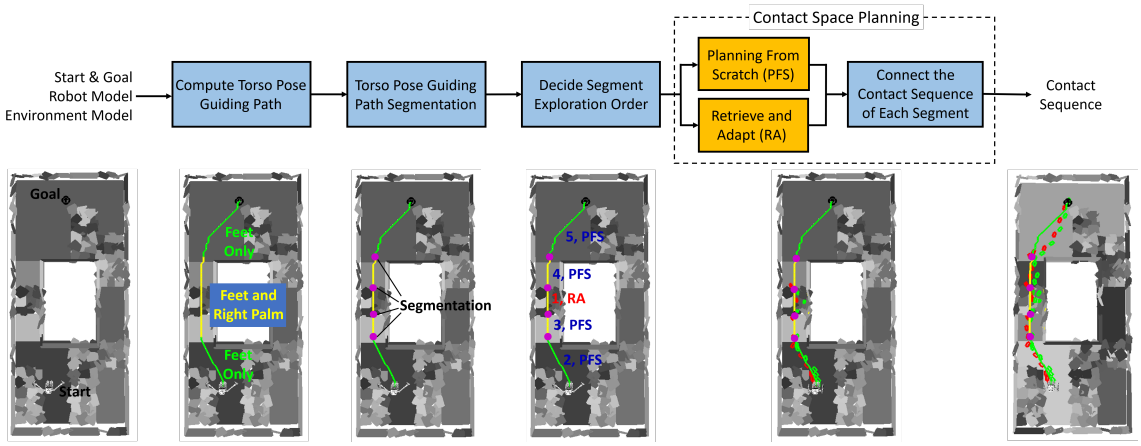


Figure 5.2: The data flow of the proposed framework. Yellow denotes that the blocks operate on segments of the torso pose guiding path.

traversal becomes difficult.

Thus, to plan a contact sequence in a large unstructured environment we present the framework shown in Figure 5.2. This framework relies heavily on the concept of humanoid *traversability*, which we introduced in previous work Lin and Berenson (2017). Traversability is defined as the time PFS will require to traverse a given area of the environment. Computing it is computationally expensive, so we have developed a way to learn a traversability estimator from data. In this work, we extend this process to consider multiple predefined motion modes (i.e. different combinations of palms and feet).

The key novel contribution of our framework is the method to segment the guiding torso path to minimize planning time. We first segment the guiding path into motion modes based on traversability predictions for each mode. We then further segment each segment based on the average traversability within the segment. This process results in segments that have either high or low average traversability. Based on the

motion mode and the traversability of each segment we then assign a planning method to use: either PFS, when the segment is easy to traverse, or RA, when it is difficult. In addition to this contribution, we also improve on the computational overhead of our RA method and generalize it to consider motion plans of widely-varying length.

Our results on randomly-generated environments with rubble suggest that our segmentation approach greatly outperforms standard graph search planning in terms of success rate. We also confirm that using the RA method for more difficult segments gives a benefit over using PFS.

5.2 Problem Statement

We address the humanoid contact navigation planning problem. Given an environment represented as a set of contactable surfaces, we wish to output a feasible sequence of stances from the start stance to a goal region in the workspace as quickly as possible. The sequence is as a series of stances, and consecutive stances in the sequence differ by one foot or palm contact pose. When executing this sequence, the robot must obey balance and collision constraints at all times. We assume that the robot can use any sequence of motion modes (from a predefined set) to traverse the environment. The motion modes are defined in terms of which end-effectors to use. The robot should always use the foot contacts, but can choose to use either or both palms to help it navigate. We assume that the robot can generate sufficient torque to balance itself. We also assume the friction coefficients are given.

5.3 Method Overview

Our framework is depicted in Figure 5.2. The process starts by computing a guiding path for the torso of the robot by planning a path in an $SE(2) \times M$ grid using the A* algorithm, where M is the set of motion modes (feet only, feet and left palm, feet and right palm, and all end-effectors). This planner uses estimates of traversability from our learned regressors to find a path that is as easy as to traverse as possible while also being biased to reduce the number of motion mode changes.

Given the torso pose guiding path found by A*, we then segment the path in two phases: first by motion mode, and then further by the traversability. This process produces segments which have either high or low average traversability. High traversability segments tend to be contact-rich, i.e. there are many viable options for contact placement. In these cases it is appropriate to use PFS to plan a contact

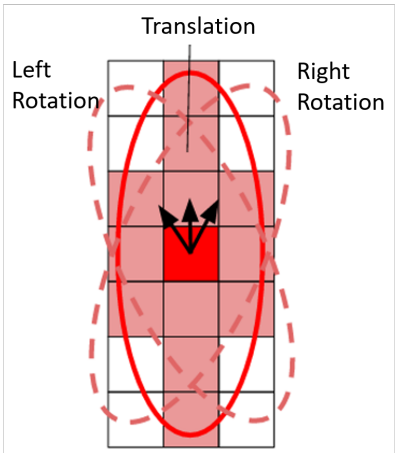


Figure 5.3: Torso pose transition model in the torso pose grid. Note that we only show the translation for one torso orientation here. To generate translation for other torso orientation, we rotate the ellipse, which represents the moving range of the torso, to align with the torso orientation and count all cells inside the ellipse as possible translations for the torso orientation.

sequence because the planner is likely to quickly find feasible contact placements. For low-traversability segments PFS is unlikely to find a solutions quickly, so we use RA, which searches a library of previously-computed motion plans for one that is appropriate for a current segment and locally-deforms the plan to the given environment. If the library is exhausted before finding a fitting plan, we default to PFS for this segment. Because PFS and RA have different start/goal specifications (RA: regions only, PFS: stance or region), before initiating planning for each segment, we order them so that connecting the segments becomes easier. Finally, when we have planned a valid contact pose sequence for all segments, we connect them with a PFS planner to produce the final result.

In the following sections, we first describe how we compute the torso pose guiding path, and how traversability for different motion modes is estimated. We then introduce the segmentation algorithm and describe how segments are ordered. Finally, we describe the PFS and RA approaches used in this work to generate the contact sequences and how sequences are connected.

5.4 Torso Pose Guiding Path

The purpose of computing a torso pose guiding path with a simplified model is to guide the higher-dimensional contact planning search. In this work, we discretize the robot torso pose in x and y, and the rotation about the z axis, θ, and call the

resulting grid the *torso pose grid.* In this work, we assume that the robot is traveling on a surface, so z is uniquely defined by the x and y coordinates. Thus we do not include z in the grid. The grid cells in which there is no contactable surface or the torso collides with the environment will be marked as invalid by the torso planner. The possible transitions of the robot torso for one step are shown in Figure 5.3. The ellipse shape captures the fact that the robot can travel farther with a forward or backward step than a lateral step.

A torso pose guiding path P_{tp} is a sequence of torso poses:

$$P_{tp} = \{p_{t,1}, p_{t,2}, \ldots, p_{t,N_p} \,|p_{t,1}, \ldots, p_{t,N_p} \in SE(2)\} \tag{5.1}$$

where N_p is the number of torso poses in P_{tp}. Note that P_{tp} is defined on a grid, so the values of each torso pose is discretized based on the density of the grid. To introduce the motion mode into the torso pose grid, we append the motion mode indicator m to each cell in the grid. m represents the motion mode of the action used to reach the cell. Based on this definition of a torso pose grid, we can rewrite P_{tp} as:

$$P_{tp} = \{(m_1, p_{t,1}), (m_2, p_{t,2}), \ldots, \left(m_{N_p}, p_{t,N_p}\right)\} \tag{5.2}$$

where m_i is the motion mode used to reach torso pose i, (note that m_1 can be any motion mode). This change in the torso pose grid will quadruple the number of cells. Although it is possible to only include motion mode information in the edges of the graph and allow the nodes to remain in $SE(2)$, we use the information of the motion mode at each node to avoid frequent changes in motion modes along the path. We do this by assigning a penalty for changing motion modes (see below). It is important to minimize the number of motion mode changes because each segment of the path is assigned a single motion mode. Frequent changes in motion mode will create many segments, and thus create many subgoals along the torso pose guiding path. This adds additional (possibly unnecessary) constraints to the original problem as well as increasing the number of calls to PFS and RA, so we would like to reduce the number of segments by reducing the number of motion changes. The algorithm to find an optimal P_{tp} is discussed below.

5.4.1 Torso Pose Guiding Path Planning

To find an optimal P_{tp}, we formulate the search problem as a graph search problem, and solve it with the A* algorithm. The edge cost Δg_{tp} between two cells $(m_i, p_{t,i})$

and $(m_j, p_{t,j})$ is defined as

$$\begin{aligned}
&\Delta g_{tp}\left(\left(m_i, p_{t,i}\right), \left(m_j, p_{t,j}\right)\right) = \\
&l_{p_t,j}^{p_{t,i}} + w_s + w_{tr}\Delta g_{tr}\left(p_{t,i}, p_{t,j}, m_j\right) + M\left(m_i, m_j\right) \\
&M\left(m_i, m_j\right) = \begin{cases} 0, & m_i = m_j \\ w_m, & m_i \neq m_j \end{cases}
\end{aligned} \tag{5.3}$$

where w_s is a fixed cost of taking a step, w_m is a fixed motion mode changing cost, $\Delta g_{tr}(0 \leq \Delta g_{tr} \leq 1)$ is the traversability cost associated with the transition from $p_{t,i}$ to $p_{t,j}$ using motion mode m_j (described in Section 5.5), and w_{tr} is a weighting factor for Δg_{tr}. Possible actions are the combination of torso pose transitions shown in Figure 5.3 and the motion modes used. The heuristic function for planning the torso pose guiding path is

$$h_{tp}\left(\left(m_i, p_{t,i}\right)\right) = d_{goal}^{t}(p_{t,i}) + w_s \frac{d_{goal}^{t}(p_{t,i})}{d_{t,max}} \tag{5.4}$$

where $d_{goal}^{t}(p_{t,i})$ is the Euclidean distance of the torso pose $p_{t,i}$ to the goal, and $d_{t,max}$ is the maximum traveling distance of the torso pose in one transition. The first and the second term are the admissible estimates of the remaining distance to the goal and the remaining transitions needed to go to the goal, respectively. Since we do not know what regions of the environment we need to traverse to reach the goal and which modes will be used in the future, the heuristic function does not contain any information related to motion mode change and traversability.

5.5 Learning Traversability

Traversability describes how quickly the contact planner can find a contact sequence to traverse through a region. If the planner knows which region has a higher traversability before planning, it can bias its search to avoid difficult regions, and generate a contact sequence more quickly. However, the true traversability will only be known after the contact planner has found a path. Therefore, we propose a learning approach to quickly estimate traversability.

For a given torso pose p_t in an environment $\mathbf{E}$, a traversability estimator is defined as $|\Gamma_+| : \{\mathbf{v}, m\} \rightarrow \mathbb{R}_+$, where $\mathbf{v}$ is a 2D torso pose translation in the XY plane, and $\mathbf{E}$ is expressed as the set of planar contact surfaces. We use a finite set of $\mathbf{v}$, as shown in Figure 5.3, and train an estimator for each $\{\mathbf{v}, m\}$ pair. Given a transition between

two torso pose $p_{t,i}$ and $p_{t,j}$ using motion mode m, we can compute $\mathbf{v}$, and then use the estimator with the matching m and closest $\mathbf{v}$.

To estimate the traversability, we start by finding a set of feasible footstep combinations Γ at p_t. To compute Γ, we first use the footstep transition model shown in Figure 3.1 and Eq. 3.1 to find possible footstep combinations given $p_t = (x_t, y_t, \theta_t)$. For example, if a transition is that the right foot moves to $\left(x_{rf}^{lf}, y_{rf}^{lf}, \theta_{rf}^{lf}\right)$ relative to the left foot, then the left foot and right foot pose can be computed in the world frame as:

$$\begin{gathered} \theta_{lf} = \theta_t - \frac{1}{2}\theta_{rf}^{lf} \; ; \; \theta_{rf} = \theta_t + \frac{1}{2}\theta_{rf}^{lf} \\ \begin{bmatrix} x_{rf} \\ y_{rf} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \cos\theta_{lf} & -\sin\theta_{lf} \\ \sin\theta_{lf} & \cos\theta_{lf} \end{bmatrix} \begin{bmatrix} x_{rf}^{lf} \\ y_{rf}^{lf} \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \end{bmatrix} \end{gathered} \tag{5.5}$$

The calculation is analogous for the left foot moving. These 3D poses of the feet will then be projected to the environment to obtain the full 6D pose. If there exists a valid projection on the environment for both feet, this footstep combination is feasible. The above computation corresponds to GetFeasibleFootstepCombination function in Algorithm 4. We denote the set of all foot and palm contact transitions as $\mathbf{FC}_{\mathbf{\Delta}}$ and $\mathbf{PC}_{\mathbf{\Delta}}$, respectively.

Given an environment, a $\{\mathbf{v}, m\}$ pair and a starting footstep combination $\gamma \in \Gamma$, if the contact planner can generate a contact sequence which applies the palm contacts specified by the motion mode m and moves the torso to the cell to which $p_t + \mathbf{v}$ belongs in the torso pose grid, we call such γ a *useful* footstep combination and denote its set Γ_+. To limit the search space, we limit the number of palm contact poses that the planner can explore, denoted $n_{\text{p,lim}}$. Therefore, the planner will return failure only when the search tree is exhausted. The number of useful footstep combinations, $|\Gamma_+|$ serves as an indicator for the traversability. The process computing the ground truth is summarized in Algorithm 4. We randomly generate rubble corridor environments to collect training data.

To compute the feature vector to estimate $|\Gamma_+|$, we first discretize each surface frame into a set of contact points $C_{p,i}$ which form a grid. We denote the set of all contact points, which is also the union of all $C_{p,i}$s from all surface, as C_p. For each contact point $c_p \in C_p$, we cast a ray from each contact point along the normal of each surface to check if the contact point is collision-free. The distance of each contact point to the closest obstacle, denoted as $\delta\left(c_p\right)$, is approximated as the closest distance

Algorithm 4: Compute Traversability Ground Truth Label

Input : $p_t, \mathbf{E}, \mathbf{v}, m, \mathbf{FC}_{\mathbf{\Delta}}$;
$\Gamma \leftarrow$ GetFeasibleFootstepCombinations $(p_t, \mathbf{FC}_{\mathbf{\Delta}}, \mathbf{E})$;
$\Gamma_+ \leftarrow \{\ \}$;
for γ **in** Γ **do**
    if *ContactSequenceExists* $(\mathbf{E}, \mathbf{v}, m, \gamma)$ **then**
        $\Gamma_+ \leftarrow \{\gamma\} \cup \Gamma_+$;
    end
end
return $|\Gamma_+|$;

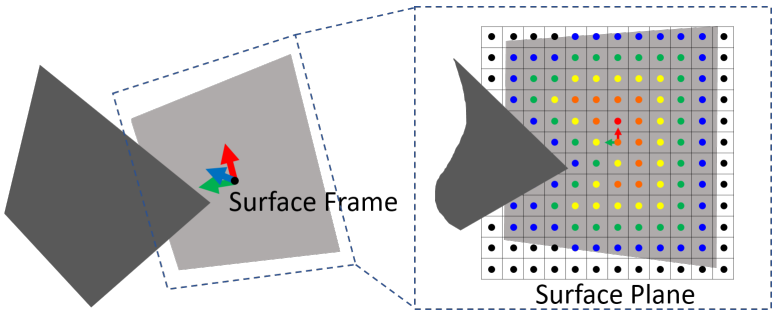


Figure 5.4: The grid of contact points on a surface plane. The distance of each contact point to the closest obstacle or surface boundary is marked in color spectrum order. Note that the light gray surface is covered by the dark gray surface, which causes part of its contact points to be infeasible.

to any contact point in collision. Figure 5.4 shows an example contact point grid of a surface. We can define the following scoring function to represent the clearance of each contact point:

$$S\left(c_p\right) = \begin{cases} 0 & \delta\left(c_p\right) < r_{\text{ins}} \\ \frac{\delta(c_p) - r_{\text{ins}}}{r_{\text{cir}} - r_{\text{ins}}} & r_{\text{ins}} \leq \delta\left(c_p\right) < r_{\text{cir}} \\ 1 & \delta\left(c_p\right) \geq r_{\text{cir}} \end{cases} \tag{5.6}$$

r_{ins} and r_{cir} are the radius of the inscribed and circumscribed circle of the contact end-effector shape, respectively. For each contact, if $\delta\left(c_p\right)$ is larger than r_{cir}, there must exist enough free space for any contact pose at the contact point c_p. However, if $\delta\left(c_p\right)$ is lower than r_{ins}, it is impossible to make contact at c_p regardless of the orientation of the contact.

S describes how likely a contact pose is feasible given its corresponding contact point c_p. In other words, each collision check is turned into a table lookup, which speeds up the process. For foot contacts, based on the footstep transition projection

shown in Figure 3.1, we can further project all the contact points on ground surfaces to a 2D grid on XY plane so that S can be queried with only the X and Y coordinate of the foot contact.

For each feasible foot combination $\gamma \in \Gamma$, we expand the contact planning search tree based on the transition model shown in Figure 3.1 for only one step. For each expansion, we can define a footstep translation tuple α as $\alpha = \left\{\mathbf{t}_{lf}^t, \mathbf{t}_{rf}^t, \mathbf{t}_{ex}^t\right\}$. $\mathbf{t}_{lf}^t$ and $\mathbf{t}_{rf}^t$ are the 2D translation of the left foot and right foot in the torso frame in the XY plane, and $\mathbf{t}_{ex}^t$ is the 2D translation of the expanded footstep in the torso frame in the XY plane. Following the definition of torso pose p_t in Eq. 3.1, each α corresponds to a translation $\mathbf{v}$ in the torso grid. Since each position in α is in the XY plane, we can pre-compute all α, and label each α by its corresponding nearest $\mathbf{v}$. We denote the set of α for each $\mathbf{v}$ as $A(\mathbf{v})$.

Lines 4 to 11 in Algorithm 5 describe the process of computing the footstep score S_f. We first iterate through each α labeled as moving in the direction $\mathbf{v}$. For each foot placement in the tuple, we find its nearest contact point, and the corresponding score using Eq. 5.6. The multiplication shown in Line 9 captures the idea that the feasibility of each footstep transition α requires all three foot contacts to be collision-free. Finally we sum the scores for each α to obtain the footstep score S_f.

For palm contacts, we project palm contacts with a given torso pose p_t as shown in Figure 3.1. Each projection returns a nearest contact point c_p on one of the surfaces. We divide the palm scores into the four quadrants of the torso frame: $\mathbf{S}_p[i]$ for the ith quadrant. This process corresponds to Lines 12 to 17 in Algorithm 5. Therefore, we define the feature vector $\mathbf{S}(p_t, \mathbf{v}, m, \mathbf{E})$ as:

$$\mathbf{S}(p_t, \mathbf{v}, m, \mathbf{E}) = \begin{cases} [S_f], & m = \text{feet only} \\ [S_f, \mathbf{S}_p[1], \mathbf{S}_p[2]], & m = \text{feet and left palm} \\ [S_f, \mathbf{S}_p[3], \mathbf{S}_p[4]], & m = \text{feet and right palm} \\ [S_f, \mathbf{S}_p[1], \mathbf{S}_p[2], \mathbf{S}_p[3], \mathbf{S}_p[4]], & m = \text{all end-effectors} \end{cases} \tag{5.7}$$

To train the estimator for each motion mode, we generate multiple environment with randomly tilted surfaces, and collect ground truth data for the difficulty in planning using the PFS approach. We then learn each estimator using Support Vector Regression (SVR) with an RBF kernel. We then define the traversability cost $\Delta g_{tr}(\mathbf{v}, m) = e^{-|\Gamma_+|(\mathbf{v},m)}$, where $|\Gamma_+|$ is the appropriate traversability estimator for that transition. With this definition, higher traversability implies a lower traversabil-

Algorithm 5: Generate the Feature Vector to Estimate Traversability

Input : $p_t, \mathbf{v}, \mathbf{E}, A(\mathbf{v}), \mathbf{PC}_{\mathbf{\Delta}}, C_p$;
$T_{p_t} \leftarrow \text{GetTransformaionMatrix}(p_t)$;
$S_f \leftarrow 0,\ \mathbf{S}_p \leftarrow [0,0,0,0]$;
for α **in** $A(\mathbf{v})$ **do**
    $S_\alpha \leftarrow 1$;
    for $\boldsymbol{t}$ **in** α **do**
        $c_{\text{Nearest}} \leftarrow \text{GetNearestContactPoint}(T_{p_t}\mathbf{t}, C_p)$;
        $S_\alpha \leftarrow S_\alpha S(c_{\text{Nearest}})$;
    end
    $S_f \leftarrow S_f + S_\alpha$;
end
for pc **in** $\mathbf{PC}_{\mathbf{\Delta}}$ **do**
    $p_{palm} \leftarrow \text{GetPalmPose}(p_t, pc)$;
    $c_{\text{Neareset}} \leftarrow \text{GetNearestContactPoint}(p_{palm}, C_p)$;
    $i_q \leftarrow \text{GetPalmQuadrant}(p_t, p_{palm})$;
    $\mathbf{S}_p[i_q] \leftarrow \mathbf{S}_p[i_q] + S(c_{\text{Neareset}})$;
end
return $[S_f, \mathbf{S}_p]$;

ity cost, and vice versa.

5.6 Torso Pose Guiding Path Segmentation

As mentioned in Section 5.3, we would like to segment the torso pose guiding path based on the motion modes and the traversability of each transition to use appropriate motion modes and planning methods (PFS or RA) for each segment. To segment the torso pose guiding path P_{tp}, we first define the torso pose transition sequence. Given a torso pose guiding path P_{tp} defined in Eq. 5.2, we can extract the torso pose transition sequence $T_\delta(P_{tp})$ defined as:

$$\begin{aligned} T_\delta(P_{tp}) &= \{\delta_1, \delta_2, \ldots, \delta_{N_\delta}\} \\ \delta_i &= (\mathbf{v}(p_{t,i}, p_{t,i+1}), \Delta\theta(p_{t,i}, p_{t,i+1}), m_{i+1}) \end{aligned} \tag{5.8}$$

where $N_\delta = N_p - 1$ is the number of transitions in P_{tp}. To solve the segmentation problem, we are looking for a partition of T_δ such that each subset in the partition contains torso pose transitions with continuous indices. For example, $T_\delta = \{\{\delta_1\}, \{\delta_2\}, \{\delta_3\}\}$, $\{\{\delta_1, \delta_2\}, \{\delta_3\}\}$ and $\{\{\delta_1, \delta_2, \delta_3\}\}$ are valid segmentations, but $\{\{\delta_1, \delta_3\}, \{\delta_2\}\}$ is not. We denote the set of all valid partitions of T_δ as $\Psi(T_\delta)$.

We segment the torso pose transition sequence using a two-stage approach. First,

we segment at every motion mode change point in T_δ, and denote this segmentation as ψ_{mm}. We then further segment each segment of ψ_{mm} based on the traversability. However, we would like to avoid segments that are too short. Therefore, if the number of transitions in a segment is less than a threshold N_{seg}, we do not segment it further; otherwise, we solve the following optimization problem to further decompose each segment of ψ_{mm}:

$$\begin{aligned} \underset{\psi \in \Psi(\psi_{mm}[k])}{\text{argmax}} \quad & \sum_{i=1}^{|\psi|} \left| \sum_{\delta_j \in \psi[i]} \Delta g_{tr} \left(\mathbf{v}\left(\delta_j\right), m\left(\delta_j\right) \right) - |\psi[i]| \, T_{tr} \right| \\ \text{subject to} \quad & |\psi[i]| \geq N_{seg} \end{aligned} \tag{5.9}$$

where ψ is a segmentation of the kth torso pose transition sequence, $\psi[i]$ is the ith segment in that segmentation, and T_{tr} is a traversability cost threshold which serves as a way to decide which method (PFS or RA) to use to generate the contact sequence. This optimization will try to generate segments whose average Δg_{tr} is above or below T_{tr} as much as possible. We also add a constraint to exclude segments that are too short. Again, it is important to reduce the number of segments for the reasons described in Section 5.4.

To solve the optimization problem we could apply existing segmentation methods, however we found that the space of segmentations was relatively small and the objective function was very fast to evaluate, thus instead we enumerate all segmentations, compute the cost of each, and choose the one that is optimal.

After the segmentation, the contact sequence generation method $\mu(\psi^*[k]) \in \{\text{PFS}, \text{RA}\}$ for each segment $\psi^*[k] \in \psi^*$ can be decided using the threshold T_{tr}. In this work, we tested two ways to make the decision. The first is to decide based on the average Δg_{tr} in the segment. If Δg_{tr} is above T_{tr}, that means the region around this torso pose path segment is more difficult, so we use RA to generate the contact sequence. We use PFS for other segments. The second approach is based on the observation that a segment may have low average Δg_{tr}, but contain some spikes in Δg_{tr}, and cause the PFS to be stuck in that part of the segment. Therefore, the second approach compares the maximum of Δg_{tr} with T_{tr}. We compare the performance of these methods in the Results section.

5.6.1 Decide Segment Exploration Order

After the segmentation is complete, each segment is planned for using either PFS or RA separately. To better connect motion plans in each segment, if a segment using

Algorithm 6: Decide Segment Exploration Order

```
Input : ψ*;
ψ_explore ← { };
ψ_PFS ← { };
for ψ*[k] in ψ* do
    if μ(ψ*[k]) = PFS then
        ψ_PFS ← ψ_PFS ∪ ψ*[k];
    end
    else
        if μ(ψ*[k]) = RA then
            ψ_explore ← ψ_explore ∪ ψ*[k] ∪ ψ_PFS;
            ψ_PFS ← { };
        end
    end
end
ψ_explore ← ψ_explore ∪ ψ_PFS;
return ψ_explore;
```

RA directly follows a segment using PFS, we can generate the contact sequence of the latter segment first, and set the first stance in the latter segment as the goal for PFS in the previous segment. Similarly, if two neighboring segments both use PFS, we will always explore the previous one first, so that the latter segment can use the last stance of the previous segment as the initial state. By doing this, we automatically connect these two segments using PFS. The only exception is the connection between two segments both using RA. In this case, we will run another PFS starting from the last stance in the previous segment, and set the first stance in the latter segment as goal. Algorithm 6 shows the procedure used to decide the segment exploration order.

5.7 The Planning From Scratch (PFS) Approach

In PFS, we follow Chapter III to formulate the contact planning problem as a graph search problem, and solve it with ANA*. Since PFS is a search-based planner, a cost is required for each action. For the foot contact transition between state s and s', we define the cost function as $\Delta g_f(s, s') = d_t(s, s') + w_s$, where d_t is the distance the approximated torso, defined in Eq. 3.1, travels in this action. For the palm contact transition, the cost is $\Delta g_p(s, s') = d_p(s, s') + w_s$, where d_p is the distance the palm travels in this action. We use a heuristic function to allow the planner to explore transitions in a goal-biased way. To compute the heuristic for each state, similar to the approach for planning the torso pose guiding path, we first plan on the torso pose grid. Our purpose here is to find the expected cost from each cell on the torso pose

grid to the goal cell. Therefore, we adopt the edge cost definition in Eq. 5.3, but use Dijkstra's algorithm to find the cost of each cell to the goal cell. This algorithm outputs a torso policy (i.e. a direction to move for each cell) in the form of a tree. PFS queries the policy using the approximated torso pose p_t defined in Eq. 3.1 to get the corresponding cost of each cell g_{tp} for use in the contact planner's heuristic function:

$$\begin{aligned} g_{tp}\left(p_t(s), m\right) &= \sum \Delta g_{tp}\left(p_i, p_{i,parent}, m\right) \\ &= l^t_{goal}(p_t(s)) + w_s N_s + w_{tr} g_{tr}\left(m\right) \end{aligned} \tag{5.10}$$

where $p_{t,\text{parent}}$ is the parent cell of the cell containing p_t in the torso policy tree, l^t_{goal} is the length of the path from the cell containing p_t to the goal cell, and N_s is the number of steps taken along that path. Note that we do not include the M term because there is only a single mode per segment.

The torso policy above will be queried as part of the heuristic for PFS. However, since the torso policy does not include palm contact, we add a component to estimate the cost of palm contact transitions along the path to the goal. We define the left and right palm component of the contact planner's heuristic as:

$$\begin{aligned} h_{p,lp}\left(p_t(s)\right) &= l_{lp}(P_{p_t}) + w_s \frac{l_{lp}(P_{p_t})}{d_{lp,max}} \\ h_{p,rp}\left(p_t(s)\right) &= l_{rp}(P_{p_t}) + w_s \frac{l_{rp}(P_{p_t})}{d_{rp,max}} \end{aligned} \tag{5.11}$$

where P_{p_t} is the path from the cell containing p_t to the goal in the torso policy, l_{lp} is the length of the portion of P_{p_t} where it is possible to make left and palm contact with the environment, and likewise l_{rp} for right palm contact. $d_{lp,max}$ and $d_{rp,max}$ are the maximum distances each palm contact can travel in one action. For a given mode, we define the palm heuristic $h_p\left(p_t(s), m\right)$ as the sum of the heuristics for all palms in that mode (0 for feet only).

To evaluate the heuristic for each state in PFS we find the grid cell containing p_t, which is estimated by taking the mean pose of foot contacts. We then combine that cell's cost g_{tp} from the torso policy with the palm component h_p to arrive at the heuristic: $h\left(p_t(s), m\right) = g_{tp}\left(p_t(s), m\right) + h_p\left(p_t(s), m\right)$.

5.8 The Retrieve and Adapt (RA) Approach

In Chapter IV, we showed that deforming an existing contact sequence to fit to the environment is an efficient approach to solve difficult contact planning problems. We constructed a motion plan library, sorted the motion plans based on how well the contacts matched to the environment, and finally deformed motion plans one-by-one until a matching motion plan was found. In this work, we keep the motion plan contact sequence matching process presented in Chapter IV, but modify it to have less computational overhead in selecting a motion plan from the library. We also generalize the original approach to allow extraction of partial motion plans in order to fit a longer plan to a closer goal. In addition we allow connecting multiple plans to reach a distant goal by making multiple queries to the library for a single segment.

We first sort the motion plan library offline based on its length. When given a query environment, we evaluate if a motion plan is promising for the given segment by measuring the distance of its contact poses to the environment surfaces after an alignment process. RA will deform the motion plan if those checks are passed; otherwise, it will skip the motion plan, continuing until either a suitable motion plan is found or the library is exhausted. We describe the library construction and query processes below.

5.8.1 Constructing the Motion Plan Library

We construct a motion plan library for each motion mode, with each mode's library containing N_{mp} motion plans. For each motion mode, we collect a library of motion plans by planning with the PFS method in randomly tilted surface environments with and without stairs. Figure 5.5 shows some examples. Each motion plan π consists of a joint trajectory, the corresponding contact sequence $\mathcal{C}(\pi)$, and the motion plan torso path $P_t(\pi)$. $\mathcal{C}(\pi)$ is defined as

$$\mathcal{C}(\pi) = \{\langle \mathbf{c}_k, e_k \rangle \,|\mathbf{c_k} \in SE(3); k = 1, 2, ..., N_c\} \tag{5.12}$$

where $\mathbf{c}_k$ is the pose of contact k in the motion plan, e_k is an indicator of which end-effector the contact k belongs to, and N_c is the number of contacts. Given the foot contact poses, we can find all approximated torso poses along the path by taking the mean of the foot contacts, and then project each approximated torso pose on the torso pose grid to form a torso path:

$$P_t(\pi) = \{p_k | p_k \in SE(2); k = 1, 2, ..., N_p\} \tag{5.13}$$

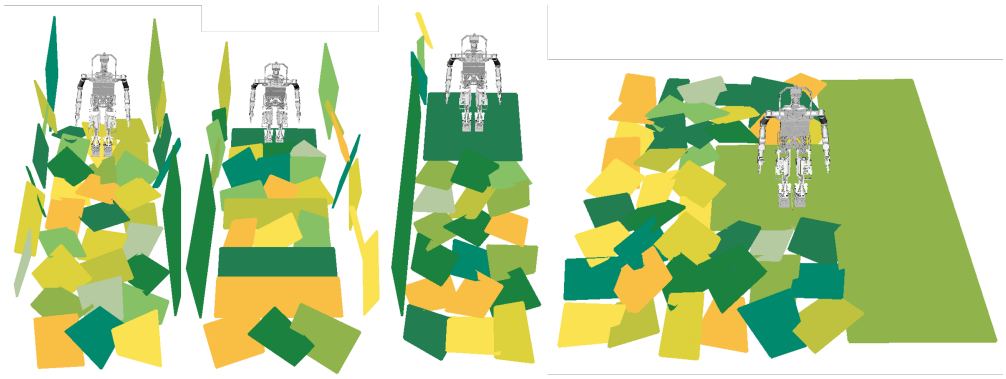


Figure 5.5: Several example environments used to collect the motion plans to construct the motion plan library.

When matching a motion plan to a torso pose guiding path segment, P_t provides a mapping between the contact sequence and its location on the torso pose grid. Therefore, $P_t(\pi)$ can help extract partial contact sequences from π to move the robot to the goal. We then extract a set $D(\pi)$ from the torso path $P_t(\pi)$ as the set of Euclidean distance in the XY plane between the start torso pose p_0 and all torso poses $p_k \in P_t(\pi)$.

$$D(\pi) = \left\{ d_k | d_k = d\left(p_1, p_k\right), d_1 \leq, \ldots, \leq d_{N_p}, p_k \in P_t \right\} \tag{5.14}$$

We force $d(p_k)$ to be monotonically increasing with k in every motion plan. If a motion plan does not follow this assumption, it can be further decomposed and stored in the library separately. We call the longest distance in D the motion plan length, l_{mp}. When searching through the library we check plans with larger l_{mp} first because, if successful, they will make the most progress toward the goal.

5.8.2 Querying the Motion Plan Library

Given a segment of the torso pose guiding path, denoted as $P_{tp,i}$, we define the start and the goal at the first and the last torso pose in $P_{tp,i}$. We denote the Euclidean distance between the start and the goal as l_{sg}. Since it is unlikely to find a motion plan to move the torso pose exactly to the goal, we define a goal radius r_g to form a circular region around the goal. When matching a motion plan to the environment, if the motion plan length l_{mp} is greater than $l_{sg} - r_g$, the motion plan has the potential to move the robot from the start to the goal region. We then check if there exist $d_j \in D(\pi)$, such that $l_{sg} - r_g \leq d_j \leq l_{sg} + r_g$. If such d_j exists, the partial motion plan corresponding to the torso path segment between the 1st and the jth torso pose can

move the robot from the start to the goal region. We extract this part of the motion plan as the effective segment of the motion plan π_e.

When $l_{mp} < l_{sg} - r_g$, the motion plan cannot move the robot from the start to the goal region. In this case, the motion plan can only cover part of $P_{tp,i}$, and stop in the neighborhood around a torso pose $p_{tp} \in P_{tp,i}$. The part of $P_{tp,i}$ after p_{tp} will then be used to query the library again. To find p_{tp}, we search from the initial torso pose in $P_{tp,i}$ toward the end of $P_{tp,i}$, and stop at the first torso pose such that $d(p_{tp,1}, p_{tp,k}) - g_r \leq l_{mp} \leq d(p_{tp,1}, p_{tp,k}) + g_r$. In this case, the effective segment of the motion plan would be the whole motion plan, so we let $\pi_e = \pi$.

In both cases, if we cannot find a π_e to meet the distance requirement, we reject this motion plan. If π_e is found, we would like to deform its joint trajectory to move the contact poses in $\mathcal{C}(\pi_e)$ to the surface patches in the environment so that the robot can make contact with the environment. To do this we apply the plan deformation process in Chapter IV, which aligns the plan to the environment using an iterative Jacobian to reduce the distance from the plan's contacts to the environment and then deforms the plan. We summarize the process below (see Chapter IV for details). This process first treats the plan as a rigid object and is initialized in the following way: Given a query environment with the start $(x_s, y_s, z_s(x_s, y_s), \theta_s)$ and the goal $(x_g, y_g, z_g(x_g, y_g), \theta_g)$, the algorithm initializes the rigid plan pose $\mathbf{T}_{rp} = (x_{0,rp}, y_{0,rp}, z_{0,rp}, \theta_{0,rp})$ as:

$$\begin{aligned} &x_{0,rp} = x_s;\ y_{0,rp} = y_s;\ z_{0,rp} = z_s \\ &\theta_{0,rp} = atan2(y_g - y_s, x_g - x_s) \end{aligned} \tag{5.15}$$

After iteratively updating $\mathbf{T}_{rp}$ until convergence, we check if the plan's contacts are too far from their nearest surfaces, and if so we reject the plan. If not, the motion plan, now expressed as a sequence of configurations, is modified and optimized to fit the query environment with CES (Chung and Khatib, 2015). Each configuration of the trajectory will moves contacts toward the nearest contact region. To speed up the process, we do not check the balance constraint in the loop of CES. Instead, we check if the resulting contact sequence follows the end-point balance constraints. If not, we reject the motion plan. Furthermore, to ensure connection between the motion plans generated in each segment of the torso pose guiding path, we force the first and last torso pose in the motion plan to be close in orientation to its corresponding pose in the torso pose guiding path segment, which means the final motion plan should obey

these constraints after the deformation:

$$|\theta_s - \theta_{\pi_e,1}| \leq \theta_\Delta, |\theta_g - \theta_{\pi_e,N_e}| \leq \theta_\Delta \tag{5.16}$$

where $\theta_{\pi_e,1}$ and θ_{π_e,N_e} are the orientation of the first and last torso pose of the motion plan π_e, respectively. θ_Δ is the orientation threshold. If all the checks have been passed, RA will output this final motion plan as the result.

5.9 Connecting the Contact Sequences

As discussed in Section 5.6.1, except for the case that the previous segment uses PFS and the latter segment uses RA, the planner for the previous segment will lead the robot to a goal region around the goal of the previous segment. If the motion modes of the two segments are different, it is possible that the last stance of the previous segment is not close enough to the next segment to make the contacts required by the motion mode of the next segment, which causes the search to fail. Furthermore, to connect two segments both using RA, the connecting planner, which uses PFS, has to find a contact sequence in the neighborhood of the connecting torso pose to the first stance of the latter segment. In a contact-scarce region, this could be difficult to plan.

We solve both of the above issues by broadening the search space. We use PFS to plan the connection sequence and allow it to use any motion mode near the connecting torso pose. This approach has a high branching factor but the connection region (which is the same size as a goal region) is very small, so the computation-time impact is limited.

5.10 Experiment on a Real Robot Platform - A Mobile Manipulator on a Steep Ramp

In this experiment, we demonstrate the motion of a real robot executing the contact sequence generated by the proposed contact planner. Since it is still an open problem to control humanoid robots to perform multi-contact motions, we use a mobile manipulator to demonstrate a real robot motion based on a planned contact sequence in a disaster-response scenario. Figure 5.6 shows the mobile manipulator used in this experiment. It is an HDT Adroit dual-arm manipulator mounted on a Clearpath Husky robot. The dual-arm manipulator is equipped with two 7-DOF

arms, and a 2-DOF (pitch and yaw) torso.

In our testing scenario the robot traverses an earthquake disaster site. A fallen ceiling forms a ramp which is so steep that the robot will tip over when driving on the ramp unless it braces itself with its hands (Figure 5.7). The robot has to plan contact sequences on the cracked and tilted wall. It can take one of two paths, either above or under the window, to reach the goal. We set the goal to be slightly higher than where the robot starts, so the path above the window is slightly shorter in distance, although it is more difficult to traverse. We compare the proposed PFS planner using our traversability estimates with a standard PFS planner which does not consider traversability ($w_{tr} = 0$). We use the following parameter values for the proposed PFS planner: $w_s = 3, w_{tr} = 10$.

The contact planning for the mobile manipulator is analogous to the formulation used in humanoid contact planning shown in Section 5.7. We transform the ground descrbied in Section 5.7 to be the wall in the mobile manipulator experiment, and the mobile manipulator is viewed as "walking" on the wall, as shown in Figure 5.7. To check quasi-static balance for each contact transition in contact planning, we set the base position to always align with the standing contact in the x direction, and follow the end-point balance constraint checking described in Section 5.7. The contact planner plans palm contacts using the transition model in which the new contact is $[0.1, 0.4]$ meters in the x axis and $[-0.2, 0.2]$ meters in the z axis from the standing contact with discretization resolution of 0.1 meter in both axes. The robot uses circular contacts, so the contact orientation remains 0 degree throughout the planning. To simplify the balance check, the support region is approximated with a conservative square contact inside the circular contact. The torso pose transition model is an 8-connected transition model in the torso pose grid in the XZ plane as shown in Figure 5.7. Since the contact orientation is always 0 degree, the torso orientation also remains 0 degree in the XZ plane.

Although this experiment uses a different platform which is not a humanoid robot, we can still use the same approach described in Section 5.5 to learn traversability estimates. For each torso translation, we collect data over sampled randomly-tilted surface environments, and train a traversability estimator with the mobile manipulator's contact transition model. The result in Figure 5.8 shows that the proposed contact planner has a much shorter planning time, but the resulting path takes more steps. Since the standard planner does not consider traversability, the planner will explore the slightly-shorter path above the window first. However, the gap created by the pipe make the path above the window require more steps than the path under

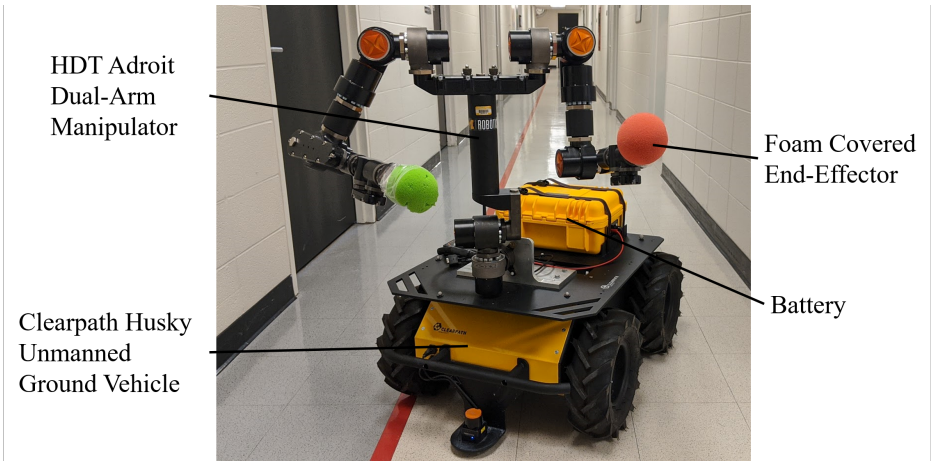


Figure 5.6: The mobile manipulator used in the experiment. The end-effectors are padded with foam covers to reduce damage on the surface of the end-effector when making contacts.

the window and the standard planner, misled by the heuristic, spends a large amount of time rejecting states around the gap before searching below the window. Because our heuristic is not admissible, we do not find a plan that is as short as the standard planner's, however we note that a difference of two steps in this context is not very large.

Although the proposed framework is originally designed for humanoid robots, we demonstrated that the application of the traversability estimates is not limited to humanoids. The experiment on the mobile manipulator shows potential extension of the proposed approach to reduce the planning time for different robot platforms which require contact planning. With the real robot experiment, we also show that the planned contact sequence is executable by a real robot.

5.11 Experiments on the Proposed Framework

We evaluate the performance of the proposed framework in planning to navigate through two types of environments and we compare the proposed framework with two baselines: The first one is the standard contact planning approach: PFS with all motion modes possible (PFS only). Since this planner is not required to use any palm contacts, it uses the feet-only motion mode heuristic to estimate the cost-to-go. The second baseline (Segmentation+PFS) uses our segmentation approach but only uses PFS to plan motion plan in each segment. Since it only uses PFS, we segment the torso pose guiding path only when motion mode changes. For the

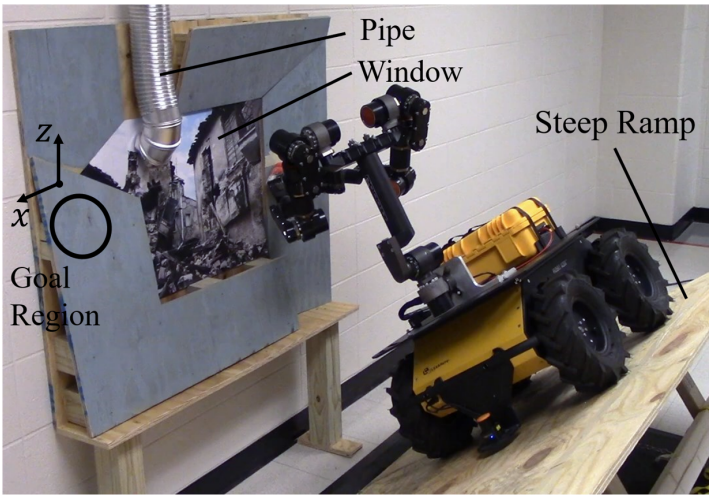


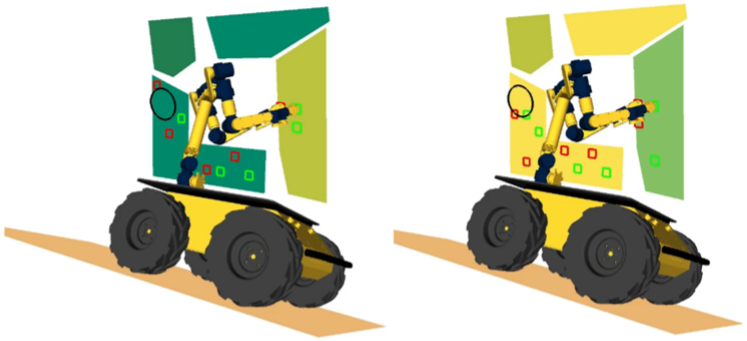
Figure 5.7: The experiment setting: The mobile manipulator moves along a steep ramp while using palm contacts to stabilize itself. The robot has to find a contact sequence to move across the window, and can only make contact to the four cracked wall surfaces showing in grey.

Table 5.1: The Result for Two-Corridor Test Environment and Two-Stair Test Environment

Environment	Approach	Success Rate	Average Number of Segments (PFS/RA/Total)	Planning Time (sec.)				
				Torso Path Planning	Torso Path Segmentation	Contact Space Planning	Segment Contact Sequence Connection	Total Time
Two-Corridor Environment	PFS Only	17/50	1/0/1	24.05	0	114.05	0	138.10
	Segmentation+PFS	27/50	4.11/0/4.11	24.81	0.01	138.95	0	163.77
	Our Framework-Mean	38/50	4.08/1.63/5.71	25.63	0.03	96.91	1.76	**124.33**
	Our Framework-Max	**42/50**	2.78/2.93/5.71	26.16	0.03	109.03	17.73	152.95
Two-Staircase Environment	PFS Only	23/50	1/0/1	24.74	0	146.05	0	170.79
	Segmentation+PFS	35/50	5.72/0/5.72	23.91	0.01	115.62	0	139.54
	Our Framework-Mean	**39/50**	4.31/3.18/7.49	24.74	0.70	117.66	1.34	144.44
	Our Framework-Max	38/50	2.74/4.75/7.49	24.67	0.71	108.11	5.19	**138.68**

proposed framework, we also implemented two versions using different decision criteria to decide whether to plan with PFS or RA for a given segment: mean(Δg_{tr}) (Our Framework-Mean) and max(Δg_{tr}) (Our Framework-Max).

We implemented our algorithms in OpenRAVE (Diankov, 2010), and tested on the Escher (Knabe et al., 2015) robot model. All experiments were run on an Intel Core i7-4790K 4.40 GHz CPU with 16GB RAM. We use the following parameter values: $N_{mp} = 50, T_{tr} = 0.3, N_{seg} = 5, w_s = 3, w_m = 2, w_{tr} = 10, r_g = 0.2m, \theta_\Delta = 30°$. The torso path grid is discretized to 0.15m resolution in x and y, and 30° in θ.



	The Standard Planner	The Proposed Planner
Planning Time (Sec.)	3.52	**0.11**
Number of Steps	**8**	10

Figure 5.8: Left: The planned contact sequence using the standard planner. Right: The planned contact sequence using the proposed PFS planner. Left and right palm contact are shown in red and green, respectively.

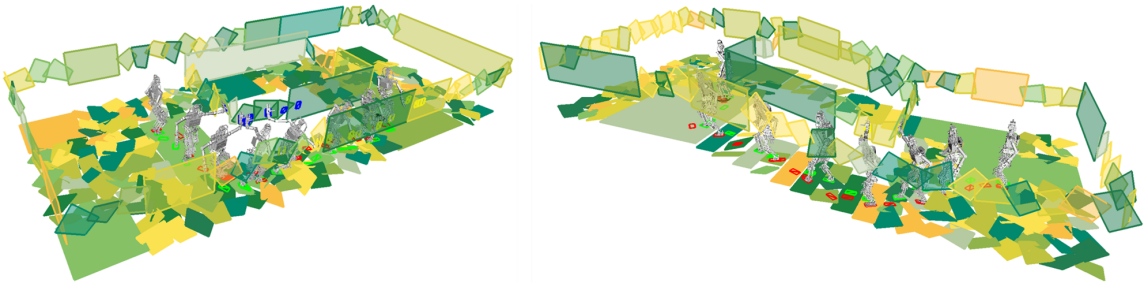


Figure 5.9: Executing a planned contact sequence in Left: a two-corridor environment. Right: a two-staircase environment.

5.11.1 Two-Corridor Environment Test

In the two-corridor environment, we construct the environment as two wide rooms connected with two parallel corridors (see Figure 5.9). The environment is formed with 1.5m by 1.5m patches, each of which is randomly generated as either flat ground or rubble with 50% probability. The rubble patches are formed with quadrilateral surfaces whose roll and pitch are sampled from a uniform distribution in $[-20°, 20°]$. The walls are also generated in the same manner. We set the start and the goal to be a random location in the lower and the upper room, respectively. We set a 500 second time limit. If the planner finds a contact sequence within the time limit in a trial, the trial is counted a success. We run on 50 testing environments, and compare the performance of the different approaches in terms of success rate and planning time for the successful trials (see Table 5.1).

The results show that the segmentation based on motion modes improves success rate by 20%. Using our full framework (i.e. introducing RA to plan for difficult segments) outperforms the other approaches in terms of success rate, while keeping the planning time low. Setting the method decision criterion based on the max traversability cost improves the success rate at the cost of higher average planning time. This is because it uses RA in some regions that can be solved quickly using PFS. The time required to connect segments also increases because there are more segments which use RA, so we require additional time to connect those segments.

5.11.2 Two-Staircase Environment Test

A two-staircase environment is shown in Figure 5.9. Testing in this environment confirms that the framework can be applied to environments with large height changes even though the torso pose guiding path is defined in $SE(2)$. In this environment, we let the upper room to be elevated by a random amount between 1m and 1.5m, and the height difference is equally distributed over 9 stairs. As in the two-corridor environment, each stair could be a flat surface or rubble with 50% probability. We use the same timeout and number of test environments as in the previous test. In this test, we again see that using segmentation gives a large performance improvement over the standard planning approach (24% increased success rate). We also see that our full framework (i.e. including RA) slightly outperforms the approach using only PFS. The improvement from using RA may be limited here because the stairs in the staircase are relatively small, so even if some stairs are rubble, there tend to be many flat steps, which are easy for PFS to traverse.

5.12 Conclusion

In this work we proposed a framework to plan humanoid navigation in unstructured environments using four predefined motion modes. The framework jointly considers the motion mode and the traversability of the environment to segment a guiding path for the torso into easy- and difficult-to-traverse segments and assigns the appropriate planning method to each. The results suggest that the proposed framework greatly outperforms standard planning without segmentation and that including library-based planning methods can also improve performance in some environments.

CHAPTER VI

Efficient Humanoid Contact Planning using Learned Centroidal Dynamics Prediction

6.1 Introduction

Humanoid robots keep balance and navigate unstructured environments by controlling the contact interaction wrenches applied at selected end-effector contact poses. In this work, we are interested in the efficient planning of such sequences of contact poses that can be used by a robot with arms and legs to optimally traverse highly dynamic, large and unstructured environments, as shown in Figure 6.1. However, as stated in Chapter I, it is computationally costly to plan dynamically feasible contact sequence. To cope with this challenge, previous approaches trade-off different factors. For instance, on the one hand, some approaches Escande et al. (2009); Hauser et al. (2006); Tonneau et al. (2018); Chung and Khatib (2015) use a quasi-static balance criteria Bresler and Frankel (1950), which lowers computational complexity but does not consider dynamic planning of contacts Bretl (2006); Caron et al. (2015); Prete et al. (2016). On the other hand, for more dynamic motions, such as when crossing a wide gap or walking down a steep slope, contact planners based on mixed-integer programming Ibanez et al. (2014); Aceituno-Cabezas et al. (2016); Aceituno-Cabezas et al. (2018) that can account for dynamics are better suited, but still suffer from the high branching factor of the search, which in large environments still remains computationally demanding for online contacts planning.

In this work, we incorporate motion dynamics within a search-based contact planner. We formulate the contact planning problem as a graph search problem where each edge corresponds to a contact transition, and the motion dynamics are evaluated for each edge. Considering motion dynamics enables the contact planner to not only plan contact sequences for dynamic motions, but also select new contacts based on a

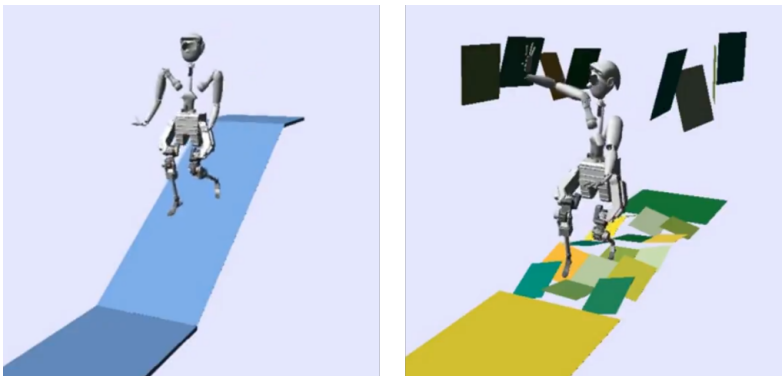


Figure 6.1: Left: The robot goes down a steep slope where quasi-static motions are not available. Right: The robot goes through a rubble corridor using both palm and foot contacts.

measure of "dynamical robustness" to achieve robust locomotion. To deal with the computationally heavy optimization of motion dynamics within the contact planning loop, we train neural networks to predict the dynamic evolution of optimal robot momentum over contact transitions, and query the networks in the planning loop to inform the contact planner how to produce contact sequences which are likely to be dynamically-robust. Using a learned approximation of optimal momentum evolution allows us to consider dynamic feasibility of transitions without paying the high computational cost of solving a dynamics optimization problem for each considered edge in the graph. The generated contact sequence is then used by a centroidal momentum dynamics optimizer Ponton et al. (2018) to produce a time-optimal dynamically feasible motion plan. To the best of our knowledge, this work is the first attempt where a learned dynamics model is used for online planning of contact sequences for a humanoid robot involving both foot and palm contacts.

In our experiments, we compare our method to a quasi-static search-based and a mixed-integer contact planner. Our results suggest that our approach produces more dynamically robust motions compared to the quasi-static planner which allows us to traverse dynamically challenging environments, and can be orders of magnitude more efficient than mixed-integer based planners in large unstructured environments.

6.2 Problem Statement

In this work, we focus our efforts on the dynamic planning of contact sequences for humanoid robots. Given an environment specified as a set of polygonal surfaces, a start stance, and a goal region, we seek to produce a dynamically-feasible contact

sequence along with a dynamics sequence, which includes centroidal momentum trajectories and contact wrenches at each time step of the trajectory, to move the robot from the start stance to the goal region within a specified planning time. The robot always uses feet contacts, but can also optionally use palm contacts when they are available. As considering variable transition times significantly increases the branching factor of the search, we assume fixed timing for each contact transition. We also assume the friction coefficient of the environment is given and fixed.

6.3 Centroidal Momentum Dynamics Optimization

The momentum dynamics have been widely adopted to plan dynamically feasible motions for floating base robots (Orin et al., 2013; Kajita et al., 2003). In this work, we use the fixed-time formulation of the centroidal dynamics optimizer proposed by Ponton et al. (2018). In the following, we briefly summarize them and explain how we use them to generate robust motion plans. The dynamics of a floating-base robot with n degrees of freedom is

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T\tau + \mathbf{J}_e^T\lambda \tag{6.1}$$

where $\mathbf{q} = \left[q^T, x^T\right]^T$ denotes the generalized robot states including joint positions $q \in \mathbb{R}^n$, and floating base frame $x \in SE(3)$. $\mathbf{H} \in \mathbb{R}^{(n+6)\times(n+6)}$ is the inertia matrix, and $\mathbf{C} \in \mathbb{R}^{n+6}$ stands for the Coriolis, centrifugal, and gravity forces. $\mathbf{S} = [\mathbf{I}^{n\times n}\ 0]$ is a selection matrix, $\tau \in \mathbb{R}^n$ is the torques vector, J_e is the endeffector jacobian, and $\lambda = \left[\cdots \mathbf{f}_e^T\ \tau_e^T \cdots\right]^T$ comprises the force $\mathbf{f}_e$ and torque τ_e of each endeffector contact. We can then decompose Eq. (6.1) to actuated parts (Eq. (6.2a)), and unactuated parts (Eq. (6.2b))

$$\mathbf{H_a}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C_a}(\mathbf{q}, \dot{\mathbf{q}}) = \tau + \mathbf{J_{e,a}^T}\lambda \tag{6.2a}$$

$$\mathbf{H_u}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C_u}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J_{e,u}^T}\lambda \tag{6.2b}$$

Under the assumption that enough torque can always be generated by the robot, if there exist robot states $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$, and the external forces λ that satisfy Eq. (6.2b), Eq. (6.2a) is also satisfied. With the assumption and decomposition, Eq. (6.2b) verifies the dynamic feasibility, and Eq. (6.2a) is only required to verify torque limits and kinematic constraints. Eq. (6.2b) is equivalent to the Newton-Euler equations of the robot (Wieber, 2006), which means that the momentum rate equals the applied

external contact wrenches. The centroidal dynamics expressed at the robot CoM is

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{l}} \\ \dot{\mathbf{k}} \end{bmatrix} = \begin{bmatrix} \frac{1}{M}\mathbf{l} \\ M\mathbf{g} + \sum \mathbf{f}_e \\ \sum (T_e(\mathbf{z}_e) - \mathbf{r}) \times \mathbf{f}_e + \tau_e \end{bmatrix} \tag{6.3}$$

$\mathbf{r}$ is the CoM position. $\mathbf{l}$ and $\mathbf{k}$ are linear and angular momenta, respectively. M is the robot mass. $\mathbf{z}_e$ is the CoP of each contact in the contact frame. $\mathbf{f}_e$ and τ_e are the contact force and torque at the CoP of each end-effector and finally, T_e is a coordinate transform in the CoM frame. In addition to Eq. (6.3), contact forces need to be inside friction cones, and CoPs inside the support regions of each contact, to prevent the contact from sliding and tilting.

To compute a dynamically robust motion we follow Ponton et al. (2018) to minimize the weighted sum of the square norm of $\mathbf{l}$, $\dot{\mathbf{l}}$, $\mathbf{k}$, $\dot{\mathbf{k}}$, $\mathbf{f}_e$, and τ_e. Lower $\mathbf{l}$ and $\dot{\mathbf{l}}$ help improve dynamic stability (Wieber, 2008). Reducing $\mathbf{k}$ and $\dot{\mathbf{k}}$ help the robot perform more natural motion (Herr and Popovic, 2008). $\mathbf{f}_e$ and τ_e terms encourage a more even distribution of forces and torques over all the contacts, which increase controllability of the robot. Additionally, we append two terms, the lateral contact forces $\mathbf{f}_l$ in the contact frame

$$\mathbf{f}_l = [\mathbf{f}_c[x], \mathbf{f}_c[y]]^T, \mathbf{f}_c = T_e^{-1}(\mathbf{f}_e) \tag{6.4}$$

and the weighted CoP position $\mathbf{z}_w$ in each contact frame

$$\mathbf{z}_w = [\frac{\mathbf{z}_e[x]}{l_{e,x}}, \frac{\mathbf{z}_e[y]}{l_{e,y}}]^T \tag{6.5}$$

where $l_{e,x}$ and $l_{e,y}$ are the lengths of the support region in X and Y direction of the contact frame. These two additional terms capture the robustness of the contact. A lower lateral contact forces favor forces away from the friction cone limits and therefore decrease the chances of sliding while a CoP position closer to the contact center decreases chance of contact tilting during execution.

Here, the dynamics optimization does not have a CoM position goal and we do not specify the final CoM position as part of the objective. Instead, a final CoM position bound is enforced as a constraint based on the mean position of the last pair of feet contacts. In the final time step of the whole contact sequence, we also constrain the CoM velocity to zero to ensure the robot can finally come to a stop.

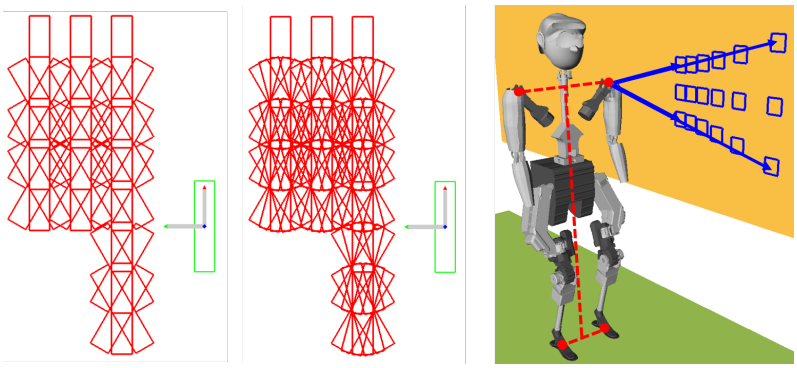


Figure 6.2: Left: The foot contact transition model used in training data collection. (38 steps) Middle: The foot contact transition model used in the experiment. (60 steps) Right: The palm contact transition model, expressed as the projections from the approximated shoulder to a wall.

6.4 Anytime Graph-Search Contact Planner

We build on the PFS contact planner described in Chapter III with modification on the state definition. In this work, in addition to the contact poses, each state further includes a CoM position and a CoM velocity to represent the centroidal dynamics. An action is either moving one end-effector to a new contact pose, or breaking one palm contact. The contact transitions are based on a predefined discrete transition model, shown in Figure 6.2, and we adopt the same contact projection scheme shown in Figure 3.1. The edge cost of each action from a state s to a state s' is defined as

$$\Delta g(s, s') = w_{xy} d(s, s') + w_s + w_{dyn} d_{dyn}(s, s') \tag{6.6}$$

where $d(s, s')$ is the XY distance the contact end-effectors' mean position travels in the contact transition, w_{xy} is the weight corresponding to $d(s, s')$, $w_s \in \mathbb{R}^+$ is a fixed cost of a contact transition, d_{dyn} is the dynamics cost, which captures the dynamical robustness of the contact transition. The dynamic cost is the optimal objective value of the dynamics optimization, discussed in Section 6.3, for the contact transition. Running the optimization in the planner is too time consuming, and we will describe how to estimate such a cost in Section 6.6. $w_{dyn} \in \mathbb{R}^+$ captures how much emphasis a user wants to put on minimizing the total dynamics cost of the path. In practice, robust contact sequences may contain more steps, and the user can adjust w_{dyn} to trade-off between the number of steps and dynamic robustness.

We solve the contact planning problem with ANA*. To guide the search, we define the heuristic function by computing the distance to reach the goal with a

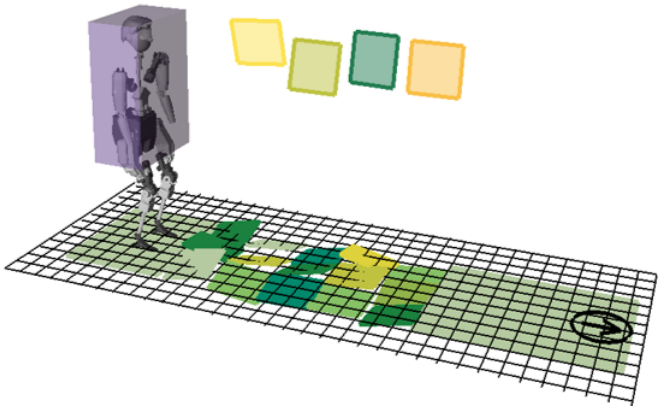


Figure 6.3: The simplified robot model, shown as the purple box, and the environment overlayed with the SE(2) grid.

simplified robot model, a floating box traveling on an SE(2) grid, as shown in Figure 6.3. We use an 8-connected grid transition model, and prune out cells where there is a collision between the box and the environment. We then plan on this grid from the cell containing the goal to every other cell in the environment using Dijkstra's algorithm. The result is a policy giving a motion direction for every cell, which can also be used to estimate the amount of motion needed to reach the goal, which we terms $d_{Dijkstra}(s)$. During contact planning, the planner queries this policy with the contacting end-effectors' weighted mean position on the XY plane, and the mean feet rotation about Z axis to compute the heuristic

$$h(s) = w_{xy} d_{Dijkstra}(s) + w_s \frac{d_{Dijkstra}(s)}{\Delta d_{max}} \tag{6.7}$$

where Δd_{max} is an overestimate of the maximum length the weighted mean contact pose can travel in one transition. The above heuristic is an example implementation for our application. It can be swapped with other heuristics, such as a Euclidean distance heuristic, or a simplified robot model policy in a discretized SE(3) space, depending on the application.

The heuristic function in Eq. 6.7 depends on the distance of the current contact poses to the goal, and does not contain any information about future dynamics cost. While ANA* will improve the solution over time, the time needed to improve the solution relies on the accuracy of the heuristic estimating future cost. The planner may be stuck in a cul-de-sac, and can only escape when the states in the cul-de-sac are exhausted. Since ANA* behaves like a depth-first search in the beginning, a cul-de-sac is especially hard for it to escape.

Index	Initial Contacts	Contact Transition	Dim.
1	Only foot contacts	Move a foot contact	24
2		Add a palm contact	24
3	Foot contacts and a palm contact	Move the inner foot contact	30
4		Move the outer foot contact	30
5		Break the palm contact	24
6		Move the palm contact	30
7		Add the other palm contact	30
8	All foot and palm contacts	Move a foot contact	36
9		Move a palm contact	36
10		Break a palm contact	30

Figure 6.4: Left: All categories of the contact transitions. The inner or outer foot means the foot in the same or opposite side of the palm contact. Each dimension includes all the initial contact poses, the new contact pose (if there is any), and initial CoM position and velocity. Right: An example environment to collect the training data. The tilting angle of each surface, the wall orientation, and wall distance to the robot are randomly sampled.

To ease the problem, we adopt the ϵ-greedy strategy Valenzano et al. (2014): With probability $1 - \epsilon$ ($0 \leq \epsilon < 1$), the planner expands a node using the same rule as ANA*, and with probability ϵ, it randomly explores a node in the priority queue. Since the random exploration does not prune out any nodes in the priority queue, and can only find new nodes or lower-cost paths to reach existing nodes, this variation does not affect the guarantees of ANA*. This strategy helps the planner escape cul-de-sacs faster by enabling the planner to explore nodes outside the cul-de-sac before exhausting it.

6.5 Evaluation of the Dynamics of Contact Transitions

To precisely evaluate the dynamics cost d_{dyn} of a contact transition to a new state, a dynamics optimization from the initial state to the new state is required. However, it is not only time consuming to compute, but also difficult to learn because the input dimension can be arbitrarily high depending on the depth of the new state in the search tree. Therefore, we approximate the dynamics evaluation as only the dynamics optimization of the contact transition. Only after the contact sequence is returned by the planner, we then apply dynamics optimization on the whole contact sequence to finally output the dynamics sequence. However, even with this simplification, running dynamics optimization for every contact transition in a search tree is still too time

consuming (in the order of 100 ms) for practical use. Therefore, we propose to learn the prediction of the results of the dynamics optimization of each contact transition using neural networks. In our test, each query to the network takes about 0.1 ms, which is 3 orders of magnitude faster than the original dynamics optimization.

6.6 Learning the Result of the Dynamics Optimization of Contact Transitions

For each contact transition, the dynamics optimizer needs to decide if it is dynamically feasible, compute the objective value as part of the edge cost, and output the CoM position and velocity of the child state. To capture the function of the dynamics optimizer in contact planning, we train two kinds of neural networks:

- A classifier to predict the dynamic feasibility
- A regressor to estimate the objective value, and the CoM position and velocity after the contact transition

The classifier has 1D binary output, which represents the feasibility of the transition, and the regressor has 7D continuous value outputs, which includes 1D objective, 3D CoM position, and 3D CoM velocity. The inputs of the neural networks are all the contact poses in the contact transition, and the initial CoM position and velocity, as same as the dynamics optimizer. To simplify the problem , we ignore CoM angular velocities in the input/output vectors, and encode the angular momentum in the objective function. We train separate neural network for each kind of contact transition using different end-effectors. Since most of the humanoid robots have symmetric kinematic structure, we further exploit this symmetry to define 10 categories of contact transition, and show its corresponding input dimensions in Figure 6.4.

The training data are collected by running the planner which calls the dynamics optimizer in each new branch in randomly tilted surface environments, as shown in Figure 6.4. The environments allow us to collect contact transitions with various contact locations and orientations. Each contact pose is encoded as a $\mathbb{R}^6$ vector with position and orientation in Tait-Bryan angles. Each angle is set to be in $[-\pi, \pi)$ to avoid the confusion of other coterminal angles. To capture the spatial relationship of the orientation data which contain angles near π and $-\pi$, we duplicate those samples with $\pm 2\pi$ in the training data, but always query the neural network with angles within $[-\pi, \pi)$.

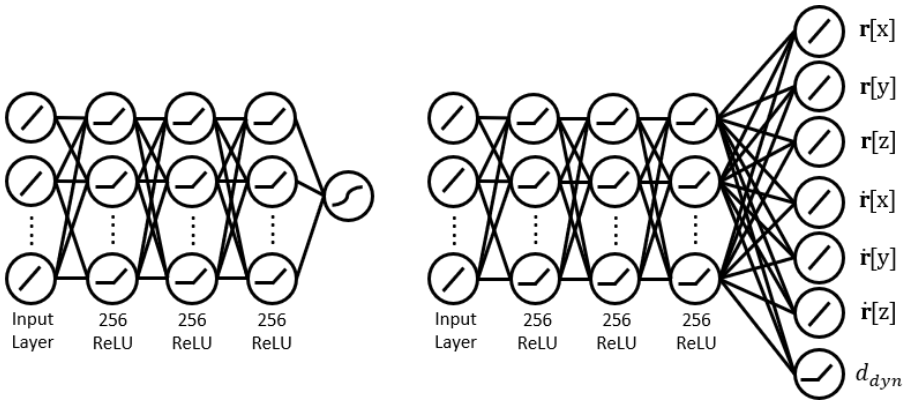


Figure 6.5: Left: the classification network. Right: the regression network.

The neural networks used in this work are shown in Figure 6.5. Although it is possible to find the best-performing network structure for each category of contact transitions, we find out that using the same structure for all categories performs reasonably well, and is much simpler in implementation. For the classifier network, the output layer uses softmax activation function, which makes the network a logistic regressor. For the regression network, the output layer is a combination of linear functions for CoM position and velocity, and ReLU for the objective value. ReLU ensures the network to output positive objective values. The hidden layers for both networks are the same, which are 3 layers of 256 fully-connected nodes using ReLU activation function.

6.7 Experiments and Results

We evaluate the performance of the proposed approach in four environments in simulation: a wide gap, a steep slope, a rubble field, and a rubble corridor, as shown in Figure 6.6. For each test, we set $w_{xy} = 1$, $w_s = 3$, $w_{dyn} = 0.1$, $\epsilon = 0.1$, and 30 seconds time limit for the proposed approach. The contact planner will keep improving solutions within this time limit, and outputs all solutions during the improvement process. With all the contact sequences returned by the ANA*, we run a complete dynamics optimization to generate a full motion sequence, from the latest to the first contact sequence until a dynamically feasible one is confirmed. For all the dynamics optimization, we fix the time step to be 0.2 second. We also fix the timing for each contact transition: 1 second in original contact (shifting CoM) and 1 second for moving the end-effector. The friction coefficient is 0.5. The weights of each term in the objective function are: $\mathbf{l}$:0.2, $\dot{\mathbf{l}}$:0.01, $\mathbf{k}$:1, $\dot{\mathbf{k}}$:0.3, $\mathbf{f}_e$:0.01, τ_e:1, $\mathbf{f}_l$:10,

and $\mathbf{z}_w$:1. All parameters are chosen empirically to help generate kinodynamically feasible motion. The dynamics optimization used in our approach is solved using the Ipopt solver Wächter and Biegler (2006). The neural networks are trained offline with Keras 2.1.6 Chollet et al. (2015) with Tensorflow 1.10.1 backend Abadi et al. (2016) for 100 training epochs, and are queried online using frugally-deep Hermann et al. (2016). All experiments were run on an Intel i7-6700 8-core 3.4GHz CPU. The proposed approach only uses a single thread. The robot has 30 DOF; 7 DOF in each manipulator and 2 torso DOF. We show the generated trajectories in the visualizer provided by the SL simulator Schaal (2009) in the attached video.

In the following experiments, we compare our approach with a baseline quasi-static contact planner, which tries to find the shortest quasi-static contact sequence to the goal. The quasi-static contact planner follows the formulation shown in Section 6.4, but it does not consider any dynamics, and only verifies the static balance of the robot stance at each state using Caron et al. (2015). We also impose the 30-second time limit, and use dynamics optimization on the contact sequence generated by the quasi-static contact planner to find its dynamics sequence. In addition to the quasi-static contact planner, we also compare the proposed planner with a mixed-integer contact planner Ponton et al. (2016) in the rubble field environment to show the advantage of the proposed approach in a non-trivial environment.

6.7.1 Wide Gap Environment Test

In this test, we show that the proposed approach can plan dynamically feasible contact sequence to cross a 0.5 meter wide gap on the ground. Including the length of the robot feet, the robot has to make a 0.72 meter stride to cross the gap, which is impossible to achieve by quasi-static walking. We use a dedicated foot contact transition model for making large step in this test, but query the same neural networks. Figure 6.6 shows the contact plan, and CoM trajectory returned by the proposed approach. It took 0.143 seconds to find the contact sequence, and 1.23 seconds for dynamics optimization over the contact sequence.

6.7.2 Steep Slope Environment Test

In this test, the robot is required to go down a 3 meter long 30° slope. The robot cannot maintain static balance on the slope, so the quasi-static contact planner is not able to find any solution. Our approach finds the first solution in 0.702s, and generating the contact sequence shown in Figure 6.6 takes 10.617s. The dynamics

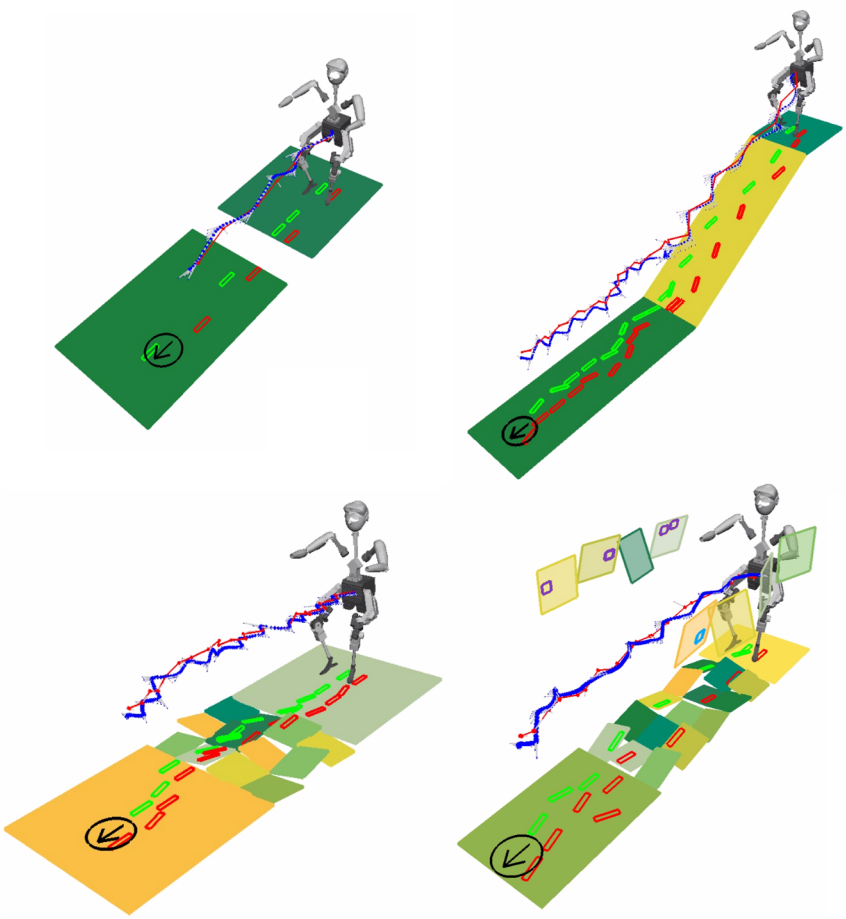


Figure 6.6: Planning examples of the proposed approach for wide gap (top left), steep slope (top right), rubble field (bottom left) and rubble corridor (bottom right) environments. The red line and blue line mark the predicted CoM trajectory, and the CoM trajectory returned by the dynamics optimizer, respectively. Contact sequences include left foot(red), right foot(green), left palm(cyan), and right palm(magenta) contacts.

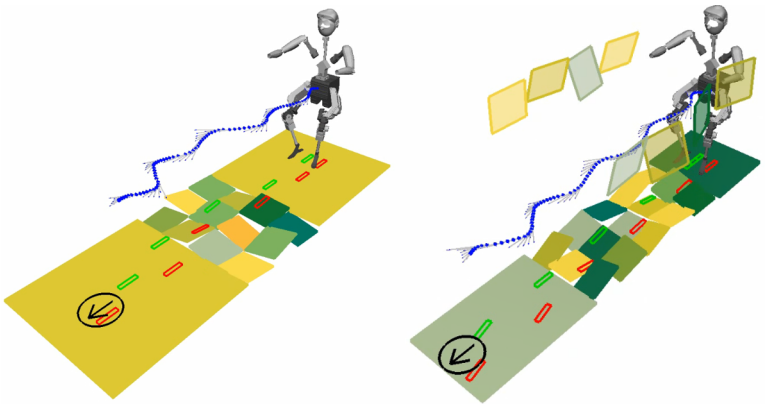


Figure 6.7: Planning examples of the quasi-static contact planner for rubble field (left) and rubble corridor (right) environments.

The Proposed Approach	Mixed integer contact planner with simplified dynamics model		
	12 Contacts	18 Contacts	24 Contacts
0.098 ± 0.037	85.93 ± 56.41	33.93 ± 18.54	46.40 ± 20.30

Figure 6.8: Time required to find dynamically feasible contact sequence in rubble field environments (Unit: second)

Test	Approach	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
Rubble Field Environment	Quasi-static Contact Planner	47/50	47/47	1.17	3498.5	12.0	12.7	**0.472**	0.706	0.767	0.012	**0.062**	0.0068
	The Proposed Approach	**50/50**	50/50	**1.02**	**3334.2**	**6.00**	**9.68**	0.581	**0.559**	**0.763**	**0.010**	0.079	**0.0079**
Rubble Corridor Environment	Quasi-static Contact Planner	44/50	44/44	**1.05**	3418.7	11.5	11.16	**0.523**	0.618	0.768	**0.013**	**0.082**	0.0069
	The Proposed Approach	**50/50**	49/50	1.59	**2392.5**	**4.38**	**4.83**	1.378	**0.349**	**0.631**	0.025	0.088	**0.0173**

Figure 6.9: Results for the rubble field corridor environments: (1) Contact planning and (2) Dynamics optimization success rates (3) Average number of tested contact sequence to find a dynamically feasible sequence (4) Mean dynamics objective of the whole contact sequence (5) Mean lin. momentum norm (kg·m/s) (6) Mean lin. momentum rate norm (kg·m/s^2) (7) Mean angular momentum norm (kg·m^2/s) (8) Mean angular momentum rate norm (N·m) (9) Mean RMS contact force norm (10) Mean contact torque (N·m) (11) Mean lateral contact force norm (12) Mean CoP distance to contact boundary (m). Contact forces are normalized by the robot weight and are unitless. In (5)-(12), means are computed over all time steps of all dynamically feasible trials.

optimization takes 6.32s to generate the dynamics sequence which contains 31 contacts.

6.7.3 Rubble Field Environment Test

The rubble field environment (Fig. 6.6), simulates a common disaster-relief scenario. The robot dynamically walks over a rubble to reach a goal about 3.4 meter away. Contact surfaces are randomly tilted in X and Y axes in $[-20°, 20°]$. The environment contains 14 convex contact surfaces.

In this test, we compare the performance of the proposed approach with the mixed integer contact planners. We first compared to a custom implementation of a mixed integer contact planner that internally solves the dynamics optimization problem as in Ponton et al. (2018), which is also used for training our neural networks. After 7 hours of planning time, it was not able to find a feasible solution. We then used a simplified dynamics model Ponton et al. (2016) and assumed that the contacts are all point contacts, which fixes each CoP to one point, and neglects the contact orientations. We solved it with state-of-the-art mixed integer solver, Gurobi 8.0 Gurobi Optimization (2018), using 8 threads. As shown in Figure 6.8, the mixed integer contact planner using the simplified model still takes much longer than the proposed approach to find a feasible solution. Furthermore, the mixed integer contact planner requires the user to specify the number of contacts used in the plan. Since the planning is in unstructured environments, it is not trivial to decide how many contacts are needed, and different number of contacts can have a great impact on the planning time (Figure 6.8).

Compared to the quasi-static contact planner, the proposed approach produces contact sequences with similar dynamics objective. However, as shown in Figure 6.9, the proposed approach generates motion with lower linear momentum and rates of linear and angular momenta. The angular momentum of the proposed approach is higher because it does not always produce straight walking motion as the quasi-static contact planner normally does, instead it may take a detour to achieve more robust locomotion using our approach.

6.7.4 Rubble Corridor Environment Test

In this test, we set up the rubble corridor environment, where palm contacts are available, and test the planner's ability to find dynamically robust contact sequence in such environment. The surfaces are randomly tilted as in Section 6.7.3. Without

any user specification, the proposed approach is able to discover palm contacts in the search, as shown in Figure 6.6. The quasi-static contact planner, on the other hand, does not consider the dynamics, and favors path with shorter traveling distance and fewer number of contacts. Therefore, it outputs solutions without palm contact, as shown in Figure 6.7. Compared to the quasi-static contact planner, the proposed approach generates motion with lower linear momentum, rates of the linear and angular momenta, and higher CoP clearance to the contact boundary, as shown in Figure 6.9. Although the angular momentum of the motion generated by the propose approach is much higher, the robot momenta rates are much lower, which results in a much lower dynamics objective of the whole contact sequence.

6.7.5 Prediction of Dynamics Optimizer Results

Here, we analyze the performance of the neural network in predicting useful information to guide the planner to find dynamically robust contact sequences. Figure 6.10 summarizes the networks' performance on predicting the results of the dynamics optimization over each contact transition. For each motion category, we use 10^5 training data, and tested with another 1000 data. The proposed approach estimates the dynamics objective of the whole contact sequence with the sum of dynamics objective in each contact transition of the contact sequence. As shown in Figure 6.11, this estimates is not accurate as it neglects previous and later contact poses in each optimization over a contact transition. However, the estimates and the actual dynamics objective are highly correlated, which makes the estimates a suitable edge cost function to select branches which lead to lower dynamics objective of the whole contact sequence.

6.8 Conclusion

We proposed a contact planner which finds dynamically robust contact sequence involving both foot and palm contacts. Costly dynamics optimization is replaced by a learned prediction of dynamic feasibility and edge cost. The planner can leverage these learned functions to efficiently evaluate contact options in the planning loop. In the future, we would like to extend the contact planner to further consider timing of each contact transition Ponton et al. (2018), so that the contact planner can generate a wider variety of dynamic motions.

Contact Transition Category Index	Dynamic Feasibility Prediction Accuracy	Mean Actual Dynamics Objective	Mean Absolute Error in Regression		
			Dynamics Objective	Final CoM (mm)	Final CoM Velocity (mm/s)
1	90.3%	1436.10	62.45	7.5	6.6
2	97.0%	740.85	40.38	6.0	5.4
3	95.3%	164.96	20.70	9.0	5.4
4	93.5%	119.85	11.07	6.7	4.7
5	94.3%	516.53	45.06	7.1	4.1
6	95.2%	87.80	12.39	9.1	4.1
7	98.1%	53.47	8.10	7.3	4.1
8	96.6%	50.66	17.28	8.1	2.4
9	96.1%	88.00	15.18	9.0	3.0
10	98.3%	62.40	7.56	8.1	3.7

Figure 6.10: Performance of the neural networks to predict dynamic feasibility, dynamics objective, final CoM and CoM velocity of a contact transition. Refer to Figure 6.4 for the meaning of each contact transition category index.

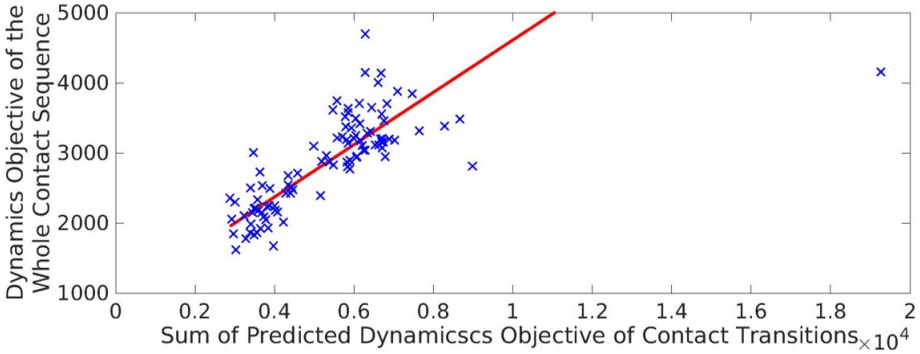


Figure 6.11: Relationship between the sum of the predicted dynamics objective of contact transitions and the actual dynamics objective of the whole contact sequence. Data taken from the rubble field and rubble corridor environments. The linear model showing the correlation is fit with robust regression Holland and Welsch (1977).

CHAPTER VII

Robust Humanoid Contact Planning with Learned Zero- and One-Step Capturability Prediction

7.1 Introduction

Algorithms efficiently computing contact sequences to traverse complex terrains are a fundamental building block for multi-contact behaviors of legged robots, in particular humanoids. In order to reduce computational complexity, most contact planners generate contact sequences considering solely quasi-static constraints (Tonneau et al., 2018; Hauser et al., 2006; Escande et al., 2009; Chung and Khatib, 2015; Lin and Berenson, 2018). However, a static stability criterion significantly decreases the set of possible contact transitions, which quickly leads to planning failure when attempting to traverse complex environments. More recently, in addition to the method proposed in VI, efficient planners using more general dynamic feasibility constraints have also been proposed (Fernbach et al., 2018; Ponton et al., 2016). Nevertheless, all these approaches assume fixed, deterministic environments and do not consider the robustness of contact sequences to potential environmental disturbances. In Figure 7.1, we show an example where the robot walks over rubble. There is a wall in the environment, and the robot can use palm contacts to capture itself against potential disturbances. However, without considering this information in the planner, a conventional contact planner could take the shortest feasible path which does not have access to the wall, and may cause the robot to fall down when a disturbance occurs.

In this capter, we propose a computationally efficient footstep planner that explicitly takes into account disturbances to increase motion robustness. In particular, we consider zero-step and one-step capture motions using either foot or palm contacts. Testing the existence of capture motions in multi-contact scenarios necessitates the solution to a kino-dynamic optimal control problem (Del Prete et al., 2018; Wang and

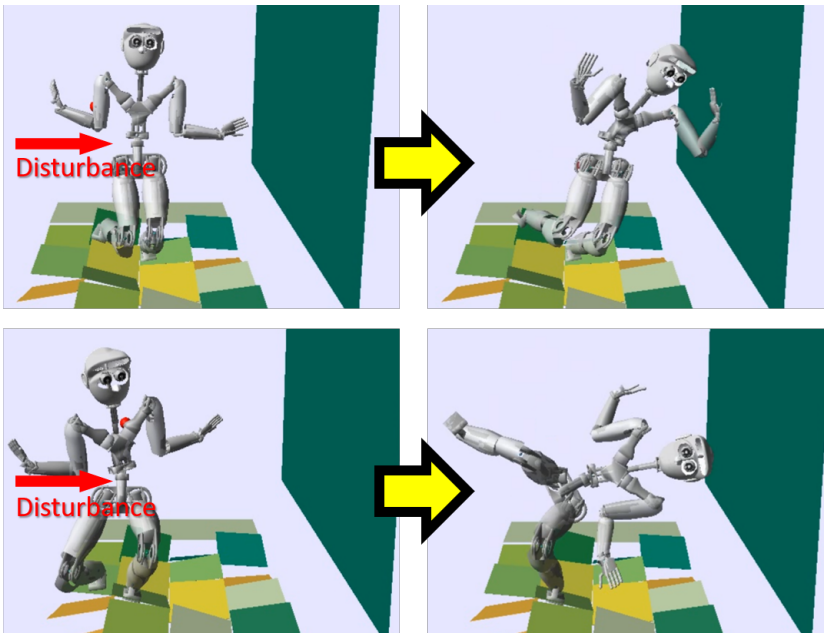


Figure 7.1: The robot walks over a rubble, and is impacted by a disturbance. Top: The robot walks close to the wall, and capture itself using a palm contact on the wall. Bottom: The robot cannot reach the wall, and falls down under the disturbance.

Hauser, 2018). However, it is prohibitively long to directly solve such problem in a footstep planner, as every candidate contact transition requires such a test. Instead, we propose to train neural networks to predict the existence of a dynamically feasible capture motion using data generated offline with a kino-dynamic optimizer. The networks predict both zero-step and one-step capturability for a full-body dynamic model using both foot and palm capture motions. We then query these networks in the footstep planning loop to inform the Anytime Non-parametric A*(ANA*) planner (van den Berg et al., 2011) about which footstep transitions are most robust to disturbances by measuring how many sampled contact poses can reject the disturbances. To the best of our knowledge, this work is the first footstep planner to use a learned model that predicts robot capturability under disturbances to produce more robust footstep sequences.

Our experiments first show that our neural networks achieve high accuracy in predicting robot capturability. We then compare our planning approach to a conventional distance-based footstep planner. Our results show that our approach generates footstep sequences that are more robust to external disturbances than the conventional method in four challenging scenarios.

7.2 Problem Statement

We focus on the problem of planning humanoid footstep sequences considering the effect of external disturbances. Given an environment specified as a set of polygonal surfaces, an initial stance (set of poses of contacting end-effectors), a goal region, and a distribution of potential disturbances in the environment, we aim to output a dynamically feasible footstep sequence to move the robot from the initial stance to the goal region. In the planning, we consider not only where the robot can create contacts to achieve dynamically feasible motions, but also how well the robot can capture itself with existing and nearby contact locations, using **both feet and hands**, to reject disturbances sampled from the distribution of potential disturbances. While it is important and desirable to generate those capture motions in real time, it is still an open problem and beyond the scope of this work. Our goal is to find a footstep sequence that maximizes the probability of the robot reaching the goal successfully without falling as a result of a disturbance. Notice that only feet are used in locomotion, but both feet and hands are available for rejecting potential disturbances. We assume that the friction coefficient is given, as well as a fixed timing for each contact transition. In this work, we consider both zero-step and one-step capture motions.

7.3 Iterative Kino-Dynamic Optimization

In order to decide whether a capturing motion exists for the full robot model, we use the kino-dynamic optimization method described by Herzog et al. (2016). Given a sequence of collision-free contact poses, the method decomposes the problem of optimizing dynamically-consistent whole-body motions and contact forces into 1) a dynamic optimization problem based on the centroidal dynamics (Orin et al., 2013) and 2) a kinematic optimization problem for the full-body motions. The algorithm computes the solution of each problems iteratively until both parts reach consensus over the center of mass $\mathbf{r}$, linear $\mathbf{l}$ and angular momentum $\mathbf{k}$ trajectories, leading to a locally optimal solution of the original problem.

In this work, we use the algorithm proposed by Ponton et al. (2018) with fixed-time to efficiently compute a solution for the dynamic optimization problem. The centroidal dynamics expressed at the robot CoM is given by

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{l}} \\ \dot{\mathbf{k}} \end{bmatrix} = \begin{bmatrix} \frac{1}{M}\mathbf{l} \\ M\mathbf{g} + \sum \mathbf{f}_e \\ \sum (T_e(\mathbf{z}_e) - \mathbf{r}) \times \mathbf{f}_e + \tau_e \end{bmatrix} \tag{7.1}$$

M is the robot mass. $\mathbf{z}_e$ is the center of pressure (CoP) of each contact in the contact frame. $\mathbf{f}_e$ and τ_e are the contact force and torque at the CoP of each end-effector and finally, T_e is a coordinate transform in the CoM frame. In addition to Eq. (7.1), contact forces need to be inside friction cones, and CoPs inside the support regions of each contact, to prevent the contact from sliding and tilting.

To compute a dynamically robust motion we follow Ponton et al. (2018) to minimize the weighted sum of the square norm of $\mathbf{l}$, $\dot{\mathbf{l}}$, $\mathbf{k}$, $\dot{\mathbf{k}}$, $\mathbf{f}_e$, and τ_e. Lower values of $\mathbf{l}$ and $\dot{\mathbf{l}}$ help improve dynamic stability (Wieber, 2008). Reducing $\mathbf{k}$ and $\dot{\mathbf{k}}$ help the robot perform more natural motion (Herr and Popovic, 2008). The $\mathbf{f}_e$ and τ_e terms encourage a more even distribution of forces and torques over all the contacts, which increases the controllability of the robot. The dynamic optimizer is run before the kinematic optimizer. After the first iteration, torque limits are included in the dynamic optimizer by using the kinematic solution to find an approximation of the torque changes during the centroidal dynamics optimization. To simplify the problem, in this work, collision avoidance is not considered in the optimization. In future works, we would like to incorporate the collision constraints using methods in Schulman et al. (2013).

A contact transition is considered capturable if the algorithm converges to consensus to a solution that satisfies all constraints after a maximum number of iterations, where we set constraints on the linear and angular momenta at the end of the movement to zero to ensure the robot will come to a stop.

7.4 Modeling External Disturbances

We model an external disturbance as an **instant** change in linear centroidal momentum. Therefore, an external disturbance δ is a 3D vector: $\delta \in \mathbb{R}^3$. We assume there is a known probability distribution of potential disturbances in each location $\mathbf{x} \in \mathbb{R}^3$ in the environment and the distribution is fixed during planning and execution time. To facilitate capturability checking, we discretize the distribution by sampling a set of representative disturbances from the distribution, and the probability of each disturbance sample is the total probability integrated over the Voronoi cell of the disturbance sample. Let $D(\mathbf{x})$ be the set of all representative disturbances. We assume that for any short period of time T, there will only be one disturbance, so we have

$$\sum_{i=1}^{N_D(\mathbf{x})} P\left(\delta_i, T\right) = 1,\ D(\mathbf{x}) = \left\{\delta_i \,|i = 1, 2, \ldots, N_D(\mathbf{x})\right\} \tag{7.2}$$

where $P(\delta_i, T)$ is the probability that δ_i happens once within time duration T, and $N_D(\mathbf{x})$ is the number of disturbance samples in $D(\mathbf{x})$.

7.5 Evaluation of Capturability

To evaluate capturability, we adopt the approach of iterative kino-dynamic optimization described in Section 7.3. Since we model the disturbance as an instant change in linear momentum, we use the post-disturbance centroidal dynamics state $[\mathbf{r}_0, \mathbf{l}_0, \mathbf{k}_0]^T$, the centroidal dynamics state immediately after the disturbance δ, as the initial state of the iterative kino-dynamic optimization, and define it as $[\mathbf{r}_0, \mathbf{l}_0, \mathbf{k}_0]^T = [\mathbf{r}_b, \mathbf{l}_b + \delta, \mathbf{k}_b]^T$, where $[\mathbf{r}_b, \mathbf{l}_b, \mathbf{k}_b]^T$ is the centroidal dynamics state before disturbance.

In this work, we consider two kinds of capture motions: zero-step capture (capturing without making new contacts), and one-step capture (capturing by making one new contact). For zero-step capture, the initial condition of the optimization includes $[\mathbf{r}_0, \mathbf{l}_0, \mathbf{k}_0]^T$, and existing contact poses. For one-step capture, in addition to the above initial conditions, we also specify a target contact pose for one of the free end-effectors.

To determine capturability, we first optimize the initial kinematic states $[\mathbf{q}_0, \dot{\mathbf{q}}_0]$ to track $[\mathbf{r}_0, \mathbf{l}_0, \mathbf{k}_0]^T$ and the existing stance S (set of contacting end-effectors poses), and then run kino-dynamic optimization for three iterations. If a kino-dynamically feasible solution can be found such that the linear and angular momentum converge to zero at the end of the motion, then the robot is capturable under the specified initial conditions: $(\mathbf{r}_0, \mathbf{l}_0, \mathbf{k}_0, S)$. For the one-step capture case, we try three different durations for the robot to move the end-effector to make contact: 0.2, 0.4 and 0.6 seconds. If any duration is feasible, then the robot is capturable given the initial conditions. The evaluation for each contact pose takes from the order of 100 ms to 1 s depending on the difficulty of the situation and the number of iterations attempted. Although it is prohibitively long to be included in a planning loop, we can collect the result offline, and fit it with an computationally efficient model.

7.6 Learning the Result of the Kino-Dynamic Optimization of Capture Motions

For each contact transition evaluated in contact planning, the planner needs to decide if the robot can capture itself under a set of disturbances D, and for each disturbance $\delta_i \in D$, many potential contacts may be considered to capture the robot

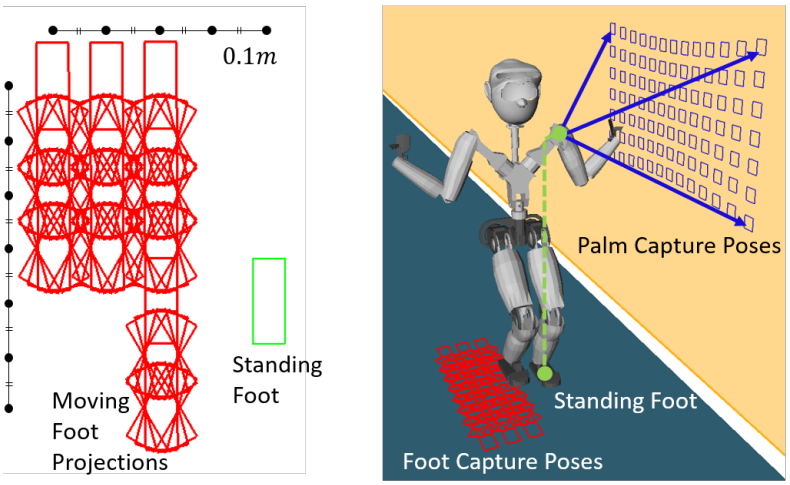


Figure 7.2: (a) Left: Foot contact transition model in searching contact sequence, (b) Right: Possible foot and palm contact projections for one-step capture motion given the standing foot pose. The projections are shown on flat surfaces as an illustrative example. When generating training data we sample contact poses with random tilt angles.

in one step. Therefore, it is computationally prohibitive to run the iterative kino-dynamic optimization in the planning loop. To reduce online computation, we train a set of neural network classifiers offline to determine capturability. Each neural network corresponds to a separate capture motion involving different contacts, as shown in Figure 7.3.

The classifiers predict whether the optimizer can find a kino-dynamically feasible solution to capture the robot given the initial conditions described in Section 7.5. Since angular momenta are generally low in walking motion (Herr and Popovic, 2008), we assume $\mathbf{k}_0 = 0$ and do not include it as the input of the network to improve data efficiency. As shown in Figure 7.3, the classifiers take the initial standing foot pose, the capture contact pose, and $[\mathbf{r}_0, \mathbf{l}_0]^T$ as inputs, and have a 1D binary output, which represents whether the optimizer can find a kino-dynamically feasible solution to capture the robot. Because most humanoid robots have symmetric kinematic structures, we utilize this symmetry and define 4 kinds of capture motion, as shown in Figure 7.3. For zero-step capture cases, the involved contact poses are only the existing contact poses; for one-step capture cases, a new contact pose of a free end-effector is considered. Each contact pose is a $\mathbb{R}^6$ vector which consists of position and orientation in Tait-Bryan angles, convention X-Y-Z, in $[-\pi, \pi)$. To capture the spatial relationship of the orientation with angles near $\pm\pi$, we duplicate those samples with $\mp 2\pi$ in the training data.

Index	Capture Motion Type	Capture Motion	Input Dim.
0	Zero-Step Capture	Maintain one foot contact	12
1	One-Step Capture	Make the other foot contact	18
2		Make the same side palm contact	18
3		Make the opposite side palm contact	18

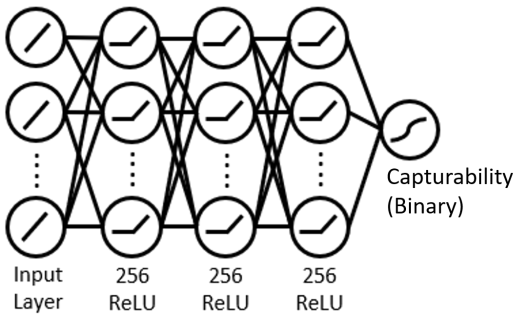


Figure 7.3: Left: Capture motions considered in this work and their feature dimension. Every capture motion initially has one foot contact, and the side of the palm contacts is relative to the standing foot side. Right: The network structure to predict capturability. The learning rate is 5×10^{-5} and there are dropout layers between fully-connected layers with 0.1 dropout rate.

To collect meaningful training data, we determine the sampling space based on the robot's reachability and the target application. If we randomly sample contact poses in a wide space, most samples are not feasible or not useful for our application. To address this issue, we first get a rough estimate of the robot's reachability with kinematic optimizers on a set of widely-sampled contact poses, and then reduce the sampling space by defining sampling intervals in each dimension of $SE(3)$ to focus more on the robot's reachable space and poses required by the application. Although the training data will need to be recollected if different robots or applications are considered, in this way, we can get a more balanced data set.

In this work, we collect data by sampling the initial standing foot contact pose with random tilt angles within $\pm 25^\circ$ from Z axis. $\mathbf{r}_0$ is randomly sampled relative to the foot pose based on the robot's reachability, and $\mathbf{l}_0$ is randomly sampled in the magnitude interval of $m[0, 1]\text{kg} \cdot \text{m/s}$, where m is the robot mass, and its orientation is randomly sampled within $\pm 45^\circ$ from the XY plane. For one-step capture cases, we sample capture contact poses using models shown in Figure 7.2. Each contact is projected with randomly selected depth and tilt angle to form a diverse set of initial conditions. Each sampled initial condition is supplied to the kino-dynamic optimizer described in Section 7.5 to decide its label. A different neural network is trained to determine capturability for each type of capture motion, but we use the same network structure for all capture motions to simplify the implementation, as shown in Figure 7.3.

7.7 Anytime Discrete-Search Contact Planner

We formulate the contact planning problem as a graph search problem. Each state s in the graph is represented by a set of: a stance $S(s)$, a CoM position $\mathbf{r}(s)$, and a linear momentum $\mathbf{l}(s)$. Each action is a foot contact transition, which means moving one foot to a new pose. Contact transitions are predefined as a discrete set of foot projections, shown in Figure 7.2(a), and we adopt the contact projection approach in Lin and Berenson (2017).

For each contact transition $\varepsilon(s, s')$ from state s to state s', the planner generates a new state with a stance which differs from the current stance by the moving contact pose. We assume there is a 0.4 second long swing phase followed by 0.6 second double support phase for each contact transition. We follow methods in VI, given $S(s)$, $S(s')$, $\mathbf{r}(s)$ and $\mathbf{l}(s)$, we use neural networks to predict dynamic feasibility of the contact transition, and determine $\mathbf{r}(s')$ and $\mathbf{l}(s')$.

We solve the contact planning problem with Anytime Non-parametric A*(ANA*) algorithm (van den Berg et al., 2011). ANA* is an anytime variation of the A* algorithm. It initially inflates the heuristic and determines which node to expand mainly by evaluating its heuristic. Once a solution is found, it then reduces the inflation of the heuristic, and improves the solution over time. In this way, a feasible solution can be generated quickly, and helps reduce the search space to find a better solution over time. The cost of each action connecting two states s and s' is defined as

$$\Delta g(s, s') = d(s, s') + w_s + w_{cap} c_{cap}(s, s') \tag{7.3}$$

where $d(s, s')$ is the euclidean XY distance between the mean foot positions of state s and s', w_s is a fixed step cost, and c_{cap} is the capturability cost and w_{cap} is its corresponding weight. We aim to generate a contact sequence which maximizes the robot's success rate to reach the goal without falling due to disturbance. Therefore, c_{cap} should be determined by the probability that the robot can capture itself during the contact transition $\varepsilon(s, s')$ from s to s' given the probability distribution of the disturbances. We denote the capture probability as $P_{\text{success}}(\varepsilon(s, s'))$.

To determine $P_{\text{success}}(\varepsilon(s, s'))$, we consider two different approaches:

- Swing Phase Discretization: Considering $n_{\mathbf{t}}$ pairs of $(\mathbf{r}, \mathbf{l})$ from discretized time steps during the swing phase of contact transition from s to s', as shown in Figure 7.4.

- Worst-case CoM Estimate: Considering only the $(\mathbf{r}, \mathbf{l})$ pair right after the robot

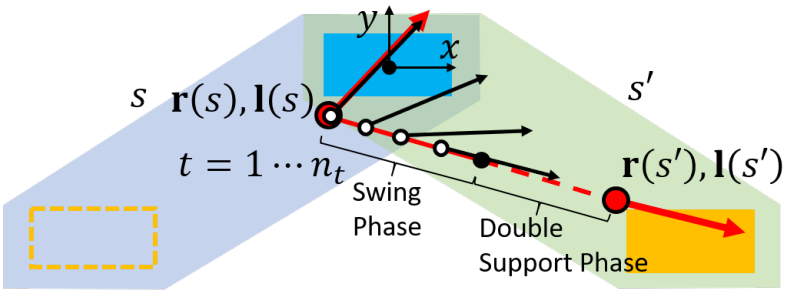


Figure 7.4: Approximated CoM position and linear momentum used to check capturability in Swing Phase Discretization. Blue and yellow boxes represent standing and swing foot, respectively. In practice, we let $n_t = 4$ to represent 4 time steps in the swing phase: $0^+, 0.1, \cdots, 0.3$ seconds from the start of the swing.

breaks a contact to start the swing phase (approximated as $(\mathbf{r}(s), \mathbf{l}(s))$).

For Swing Phase Discretization, $P_{\text{success}}(\varepsilon(s, s'))$ is defined as

$$\prod_{t=1}^{n_t} \sum_{i=1}^{N_D(\mathbf{r}_t)} P_{\text{reject}}\left(\mathbf{r}_t, \mathbf{l}_t, S_{\text{swing},\varepsilon(s,s')}, \delta_i\right) P\left(\delta_i, \frac{0.4}{n_t}\right) \tag{7.4}$$
$$\begin{cases} \mathbf{r}_t = \frac{n_t - t}{n_t - 1}\mathbf{r}(s) + \frac{t-1}{n_t-1}\mathbf{r}_{\text{swing},\varepsilon(s,s')} \\ \mathbf{l}_t = \frac{n_t - t}{n_t - 1}\mathbf{l}(s) + \frac{t-1}{n_t-1}\mathbf{l}_{\text{swing},\varepsilon(s,s')} \end{cases} t = 1, \cdots, n_t$$

where $P_{\text{reject}}\left(\mathbf{r}_t, \mathbf{l}_t, S_{\text{swing},\varepsilon(s,s')}, \delta_i\right)$ means the probability of the robot rejecting disturbance $\delta_i \in D(\mathbf{r}(s))$ with centroidal dynamics state before disturbance $[\mathbf{r}_b, \mathbf{l}_b, \mathbf{k}_b]^T = [\mathbf{r}_t, \mathbf{l}_t, 0]^T$, and the robot's stance in swing phase $S_{\text{swing},\varepsilon(s,s')}$. $\mathbf{r}_{\text{swing},\varepsilon(s,s')}$ and $\mathbf{l}_{\text{swing},\varepsilon(s,s')}$ are $\mathbf{r}$ and $\mathbf{l}$ at the end of the swing phase, and they are set empirically to be $\mathbf{r}_{\text{swing},\varepsilon(s,s')} = 0.4\mathbf{r}(s') + 0.6\mathbf{r}(s)$ and $\mathbf{l}_{\text{swing},\varepsilon(s,s')} = \mathbf{l}(s')$, respectively. Although only time steps in swing phase are considered here, empirically we find that for each step cycle, the robot has similar performance to reject disturbances by reactive stepping in double support phase or one-step capture in swing phase. Therefore, to reduce the computation load, we sample only from the swing phase in planning.

For Worst-case CoM Estimate, $P_{\text{success}}(\varepsilon(s, s'))$ is defined as

$$\sum_{i=1}^{N_D(\mathbf{r}(s))} P_{\text{reject}}\left(\mathbf{r}(s), \mathbf{l}(s), S_{\text{swing},\varepsilon(s,s')}, \delta_i\right) P(\delta_i, 0.4) \tag{7.5}$$

In this definition, $P_{\text{success}}(\varepsilon(s, s'))$ only depends on s and $S_{\text{swing},\varepsilon(s,s')}$, so for all s' with the same $S_{\text{swing},\varepsilon(s,s')}$, $P_{\text{success}}(\varepsilon(s, s'))$ is the same. Therefore, compared to Swing

Phase Discretization, Worst-case CoM Estimate reduces the computation time significantly because it only considers one centroidal dynamics state. During the contact transition, disturbances pushing toward $+y$ direction in standing foot frame are hard to capture with the swing foot because of the kinematic constraints. As seen in Figure 7.4, we observe that in dynamic walking, at the start of the swing phase, the robot has the highest $+y$ component of the linear momentum. Therefore, in Worst-case CoM Estimate, we sample the start of the swing phase of each $\varepsilon(s, s')$, and use it to determine $P_{\text{success}}(\varepsilon(s, s'))$.

7.7.1 Modelling disturbance rejection probability

Both definitions of $P_{\text{success}}(\varepsilon(s, s'))$ require the disturbance rejection probability $P_{\text{reject}}(\mathbf{r}, \mathbf{l}, S_{\text{swing},\varepsilon}, \delta)$. For each $S_{\text{swing},\varepsilon}$, we use the foot and palm projection model shown in Figure 7.2(b) to find all possible capture poses. We then query the neural networks with $\mathbf{r}$, $\mathbf{l}$, $S_{\text{swing},\varepsilon}$ and each of those capture poses, and count the number of queries that output "capturable", including the zero-step capture motion, denoted as n_c. Since the neural networks simplify the capturability check by abstracting the initial kinematics state to be a combination of a stance and a dynamics state, and assuming no initial angular momentum, we expect errors caused by these simplifications. Therefore, we would like to improve the planner robustness by favoring transitions $\varepsilon(s, s')$ which are predicted by the networks to be capturable with more capture poses (higher n_c for each disturbance). Therefore, we model $P_{\text{reject}}(\mathbf{r}, \mathbf{l}, S_{\text{swing},\varepsilon}, \delta)$ as $1 - \exp(-\gamma n_c)$, where $\gamma \in \mathbb{R}^+$ is a user defined constant. This model captures the idea that the robot is more likely to reject the disturbance if more network queries with different capture poses determine the condition to be capturable.

7.7.2 Capturability Cost

For a path T_{cp} (a sequence of K contact transitions), the probability that the robot finishes the path without falling due to external disturbance is

$$P_{\text{success}}(T_{cp}) = \prod_{k=1}^{K} P_{\text{success}}(\varepsilon_k) \tag{7.6}$$

where ε_k is the kth contact transition in T_{cp}. Our goal is to maximize $P_{\text{success}}(T_{cp})$, which can be achieved by minimizing $\sum_{k=1}^{K} -\log(P_{\text{success}}(\varepsilon_k))$. Therefore, we define c_{cap} as

$$c_{cap}(s, s') = -\log(P_{\text{success}}(\varepsilon(s, s'))) \tag{7.7}$$

With this definition of c_{cap}, we can find a path with maximum success rate by minimizing the total capturability cost of the path, which is done by the ANA* algorithm. In practice, we set $w_{cap} \gg w_s, d(s, s')$ to let ANA* focus on maximizing $P_{\text{success}}(T_{cp})$.

7.7.3 Contact Planning Heuristic

To guide the search, we follow same method in 6.4 and define the heuristic function by computing a policy for a simplified robot model moving on an SE(2) grid. The robot simplified model is a floating box. We first prune out every cell in the grid where there is no ground or there is collision between the box and the environment, and plan with Dijkstra's algorithm from the goal cell using an 8-connected grid transition model. By doing so, every cell connected to the goal cell will get a shortest distance $d_{\text{Dijkstra}}(s)$ to reach the goal and a policy which indicates the neighboring cell to go to. During contact planning, the planner queries this policy with the mean foot position on the XY plane, and the mean foot rotation about the Z axis to compute the heuristic.

$$h(s) = d_{\text{Dijkstra}}(s) + w_s \frac{d_{\text{Dijkstra}}(s)}{\Delta d_{max}} \tag{7.8}$$

where Δd_{max} is an overestimate of the maximum distance the mean foot pose can travel in one transition.

7.8 Experiments

We evaluate the performance of the proposed approaches in three test environments in simulation: a narrow, flat strip of ground, a field of rubble with an adjacent wall, and part of an oil platform, as shown in Figure 7.5. For each test, we allow 1 minute planning time, and set $w_s = 3$, $w_{cap} = 1000$, $\gamma = 0.1$ and the friction coefficient is 0.5. We compare the proposed approaches with the baseline approach which only considers moving distance and step number ($w_{cap} = 0$). For all test environments, we show the planned footstep sequences in Figure 7.5, and summarize the quantitative results in Figure 7.7.

Since small disturbances can be handled by the robot's momentum controller, and do not require the planner to explicitly find capture motion to reject them, in the below experiment, we only consider the relatively rare but dangerous case that high disturbances D_{high} act on the robot. Unless otherwise stated, we set the probability of those high disturbances happening within every time step (0.1 second) as $P(D_{\text{high}}, 0.1) = 1\%$. To make the result easier to interpret, we let $P(\delta_i, 0.1),\ \delta_i \in$

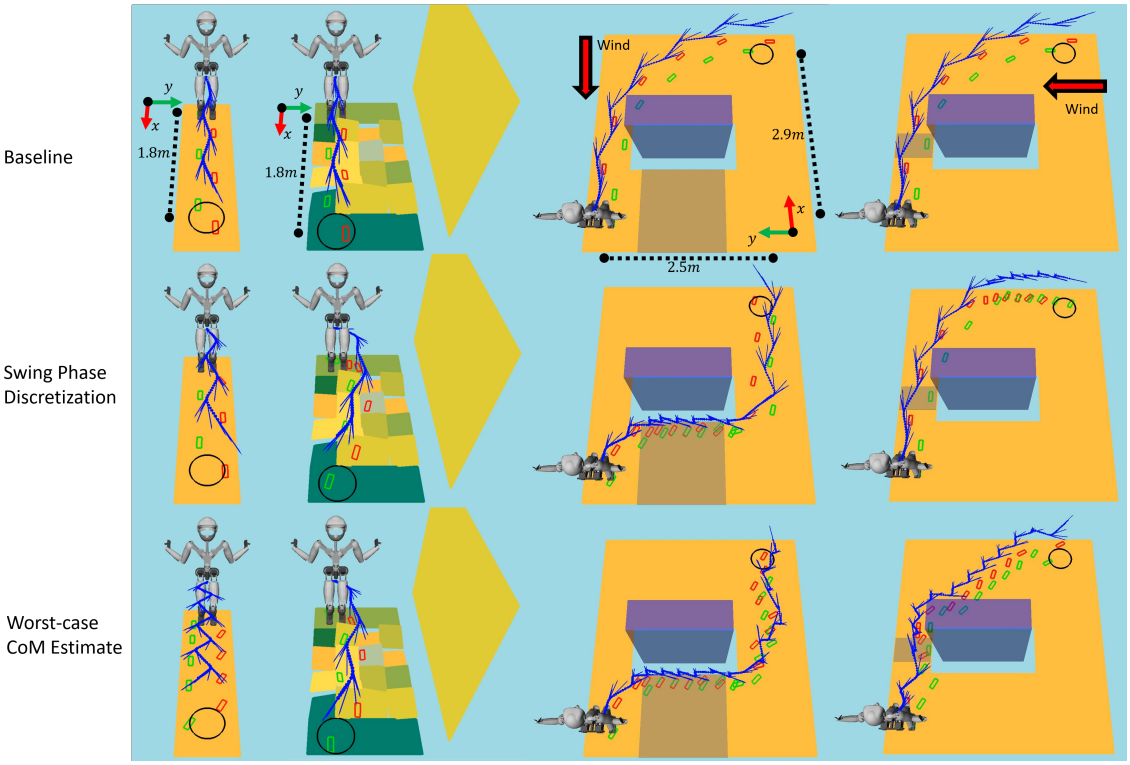


Figure 7.5: From left to right: The planned footstep sequence in the narrow flat corridor, the rubble with wall, and the oil platform (wind in $-X$ and $+Y$ direcitons). The CoM trajectories returned by the kino-dynamic optimizer given the footstep sequences are shown in blue.

Index	Precision	Recall	Accuracy
0	97.4%	98.3%	97.8%
1	98.0%	98.0%	98.0%
2	95.9%	94.3%	95.2%
3	92.2%	90.3%	91.3%

Figure 7.6: The neural networks' performance

D_{high} evenly divide $P(D_{\text{high}}, 0.1)$.

To evaluate the planned contact sequence, we first get its corresponding kinematic trajectory using the iterative kino-dynamic optimizer described in Section 7.3. Each trajectory is a discrete sequence of $\mathbf{q}, \dot{\mathbf{q}}$ with time steps of 0.1 second. For each time step t_j of the kinematic trajectory, including both swing and double support phases, we take the configuration as the initial kinematic state, and apply disturbances $\delta_i \in D_{\text{high}}(\mathbf{r}(t_j))$ one by one and check if the robot can capture itself using the approach described in Section 7.5. For each disturbance $\delta_{\mathbf{i}}$, we first check if the condition is zero-step capturable, if not, we then check if it is one-step capturable with any of the capture poses generated using contact projection shown in Figure 7.2. In double support phase, when testing one-step capturability, we allow the robot to break one existing contact, and make contact at a capture pose. With the capturability of the robot for each time step - disturbance pair, we finally compute the probability that the robot finishes the path without falling due to external disturbance $P_{\text{success}}(T_{cp})$ to evaluate the path quality.

We run the experiments on an Intel i7-8700K 3.7GHz CPU, and use an NVIDIA GeForce RTX 2080 GPU to speed up network queries for the Swing Phase Discretization approach. The neural networks are trained with Keras 2.2.4, and queried with Tensorflow 1.4 C++ API. The robot model we use is a Sarcos Humanoid robot.

7.8.1 Prediction of Zero-Step and One-Step Capturability

Figure 7.6 summarizes the performance of the neural networks in predicting capturability given an initial stance, a CoM position and a linear momentum. For each capture motion category, we train the network with 10^5 examples, and test it with another 1000 examples. Although all models perform well in predicting the capturability, the performance of predicting capture motions using palm contacts is worse than its counterpart using foot contact. This may be because capture motions using palm contacts are more likely to violate kinematic constraints and have higher variance in kinematic state, which cause them to be harder to learn.

7.8.2 Narrow Flat Ground Test Environment

In this test environment, we would like to show an intuitive result of how the robot can adjust its footstep placement to be more robust to external disturbances. We consider two lateral disturbances: $D_{\text{high}} = \left\{m[0, \pm 0.6, 0]^T\right\}$ kg $\cdot$ m/s. In this case, the most dangerous situation is when the robot shifts its CoM to one side, and the disturbance pushes in the same direction. In this situation, the robot mainly relies on zero-step capture motion to reject the disturbance. The proposed approaches make the robot increase the step width of the motion, which expands the support region in the y direction, and hence makes the robot more stable.

7.8.3 Rubble with Wall Test Environment

In this test, the robot has to traverse through a rubble with a side wall, similar to the rubble environment used in the DARPA Robotics Challenge. We test for five randomly generated rubble surfaces with different tilt angles, and set $D_{\text{high}} = \left\{m[0, 0.5, 0]^T, m[0, 0.6, 0]^T, \ \ m[0, 0.7, 0]^T, m[0, 0.8, 0]^T\right\}$ kg $\cdot$ m/s. Although the wall provides a wide space for the robot to capture itself using palm contacts, it is too far away for the robot to reach if the robot simply walks straight to the goal. The planner is able to incorporate this information, and adjust the path to be close to the wall, and achieves a much more robust footstep sequence under the disturbances.

7.8.4 Oil Platform Test Environment

This test demonstrates how the planner adapts to different sets of disturbances. We consider a part of an offshore oil platform with wind blowing. There are structures on the oil platform that can block the wind, but are not suitable for palm contacts, such as electronics and pipes. We first consider $D_{\text{high}} = \left\{m[-0.6, 0, 0]^T, m[-0.7, 0, 0]^T, m[-0.8, 0, 0]^T\right\}$ kg$\cdot$m/s, and the wind is blocked by the structure in the center, which creates a region without disturbance, shown in grey in Figure 7.7. We show that the proposed approaches leverage this region to produce low-risk contact sequences.

In another test, we consider a different wind direction with $D_{\text{high}} = \left\{m[0, 0.6, 0]^T, m[0, 0.7, 0]^T, m[0, 0.8, 0]^T\right\}$ kg$\cdot$m/s. In this test, we show that the proposed approach is able to adapt to this change and produce a different contact sequence, shown in Figure 7.5. However, this wind direction imposes great challenges to the planner because the robot will have to travel a long distance under the strong wind. This will create many high-cost edges, which drive predicted $P_{\text{success}}(T_{cp})$ low, and many paths look similarly costly in planning. Therefore, it is not easy for ANA* to reduce

search space quickly. In this case, the Worst-case CoM Estimate approach and the baseline outperform the Swing Phase Discretization approach. The first reason is that the shortest path happens to be a good path in this case. The second reason is that Swing Phase Discretization approach branches each state much slower than the other approaches due to the large amount of network queries. Therefore, it failed to find a good solution within the time limit.

7.8.5 Summary of the Planning Results

In summary, we show that the proposed approaches generate contact sequences more robust to disturbance for the scenarios considered, except for the oil platform environment with $+Y$ wind direction where Worst-case CoM Estimate approach and the baseline have similar performance. Although Worst-case CoM Estimate simplifies the capturability check of each contact transition for higher efficiency, its performance is comparable to Swing Phase Discretization approach. In general, compared to the baseline, the proposed approaches take longer to plan a contact sequence. However, if we consider scenarios with shorter horizon, such as the narrow flat ground and rubble with wall environment, Worst-case CoM Estimate approach has planning time much shorter than the execution time, and could be used in a receding horizon fashion.

7.9 Discussion

While the proposed approaches perform much better than the baseline, there still are time steps that the robot failed to reject the disturbances when following the footstep sequence generated with the proposed approaches. In addition to wrong predictions by the network, the disturbance rejection probability model could sometimes be misleading. In Section 7.7.1, we define the disturbance rejection probability to depend on the number of feasible capture poses. Since many capture poses are similar, as shown in Figure 7.2, if there is a wrong prediction, the network is likely to have multiple wrong predictions given by similar capture poses. To improve the model, one possible direction for future work is to use ensemble learning to increase the prediction's robustness.

In this work, we plan humanoid contact sequences which enable the robot to more easily capture itself under external disturbances. While the decision on where to place contacts is crucial for a successful capture, CoM position and centroidal momentum also play an important role. In our current approach, during planning, the CoM position and centroidal momentum of the robot in each state is determined

Test Environment	Approach	Number of Failed Time Step - Disturbance Pairs			Step Number	Planning Time (s) (First Solution/Best Solution within the Time Limit)	P_{success} (T_{cp})
		Total	Swing Phase	Double Support Phase			
Narrow Flat Ground	Baseline	36	14/40	22/60	5	0.52/0.52	83.49%
	Swing Phase Discretization	**15**	4/40	11/60	5	0.88/2.13	**92.75%**
	Worst-case CoM Estimate	18	6/72	12/108	9	0.60/4.29	91.37%
Rubble with Wall	Baseline	89.80±2.79	35.40±2.06/80	54.40±3.72/120	5±0	0.54±0.01/0.54±0.01	79.83±0.56%
	Swing Phase Discretization	17.80±7.30	16.60±5.46/128	1.20±2.39/192	8±0.63	1.17±0.84/13.74±4.58	95.84±1.65%
	Worst-case CoM Estimate	**11.6±1.36**	11.6±1.36/112	0±0/168	7±0	0.58±0.07/1.09±0.16	**97.14±0.33%**
Oil Platform (Wind in $-X$ direction)	Baseline	25	10/144	15/216	12	0.54/11.353	91.99%
	Swing Phase Discretization	**0**	0/288	0/432	24	1.249/22.381	**100%**
	Worst-case CoM Estimate	1	1/384	0/576	32	0.582/30.728	99.67%
Oil Platform (Wind in $+Y$ direction)	Baseline	45	18/144	27/216	12	0.54/11.353	86.02%
	Swing Phase Discretization	61	23/252	38/378	21	1.103/2.572	81.54%
	Worst-case CoM Estimate	**42**	23/312	19/468	26	0.555/35.894	**86.90%**

Figure 7.7: The performance of each approach in all test environments. Note that there are 4 and 6 time steps in swing and double support phase, respectively. $P_{\text{success}}(T_{cp})$ is only affected by failed time step - disturbance pairs, so even some contact sequences are longer, its $P_{\text{success}}(T_{cp})$ can still be higher.

by a neural network which learns from a dynamics optimizer without the information of the disturbances. The solution quality may increase if we includes CoM position and centroidal momentum as decision variables in the planner. However, this will significantly increase the branching factor, and slow down the planning.

While our approach is capable of finding footstep sequences that are more robust to potential disturbances, it is still necessary to have a controller to react to disturbances during execution and select the appropriate next contact in real-time. Several approaches have been proposed to find the next contact location that helps stabilize a robot, such as in Mason et al. (2018), but they often use a simplified model of the dynamics. It could be interesting to extend the learning part of our approach to use it in a real-time controller in order to remove the need for simplifying assumptions on the dynamics.

7.10 Conclusion

In this capter we addressed the problem of finding contact sequences that are not only dynamically feasible but are also robust to external disturbances. It is the first time, to the best of our knowledge that a contact planning algorithm explicitly considers the effect of external disturbances. In order to enable a fast evaluation of the capturability of a transition, we trained classifiers using neural networks, leading to a significant speed-up in planning time. Experimental results demonstrate that our approach can quickly find contact plans that are less susceptible to external disturbances, which leads to more robust behaviors when executed on a real robot.

CHAPTER VIII

Conclusion

This dissertation presented frameworks to address the humanoid multi-contact navigation planning problem in unstructured environment. Based on a search-based planner over the contact poses, we focused on efficiently selecting contact poses for a humanoid robot to reach the goal assuming multi-contact motion which follows kinematic and dynamic constraints. From Chapter III to V, we focused on techniques that reduces planning time in a large unstructured environment by using better heuristics learned from environment traversability features and applying trajectory optimization approaches to regions with low traversability. The results show great improvement in success rate and planning time compared to a conventional search-based contact planner. In the remaining two chapters, we proposed the use of neural networks as function approximators to efficiently consider humanoid robot dynamics in contact planning. The proposed approach can efficiently verify motion dynamic feasibility and predict capturability under external disturbances, which enables the contact planner to produce dynamically robust contact sequences.

One of the limitations in this dissertation is the assumption to decompose the humanoid motion planning problem into contact planning and trajectory planning. Although this assumption helps simplify the problem and enables the state-of-the-art personal computer to solve the problem efficiently, this user-defined decomposition can drop information and produce suboptimal solution. In this dissertation, we improve the connection between contact planning and trajectory planning with the learned centroidal dynamics approximator. However, the trajectory planning module still cannot alter the contact poses returned by the contact planner. It would be interesting to see an iterative approach where trajectory planning module provides gradient for the contact planner to improve its solution locally.

APPENDICES

BIBLIOGRAPHY

BIBLIOGRAPHY

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

B. Aceituno-Cabezas, H. Dai, J. Cappelletto, J. C. Grieco, and G. Fernndez-López, "A mixed-integer convex optimization framework for robust multilegged robot locomotion planning over challenging terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernndez-Lpez, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, July 2018.

B. Aceituno-Cabezas, J. Cappelletto, J. C. Grieco, and G. Fernández-López, "A generalized mixed-integer convex program for multilegged footstep planning on uneven terrain," *CoRR*, vol. abs/1612.02109, 2016. [Online]. Available: http://arxiv.org/abs/1612.02109

H. Audren, J. Vaillant, A. Kheddar, A. Escande, K. Kaneko, and Y. E., "Model preview control in multi-contact motion-application to a humanoid robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014. [Online]. Available: http://dx.doi.org/10.1109/IROS.2014.6943129

L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida, "Real-time replanning using 3d environment for humanoid robot," in humanoids," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2011.

D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *ICRA*, 2012.

E. Bresler and J. P. Frankel, "The forces and moments in the leg during level walking," *Journal of Applied Mechanics*, vol. 72, pp. 27–36, 1950.

T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *The Internation Journal of Robotics Research*, vol. 25, no. 4, pp. 317–342, Apr. 2006. [Online]. Available: http://dx.doi.org/10.1177/0278364906063979

O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," in *IJRR*, 2002.

S. Caron and A. Kheddar, "Multi-contact walking pattern generation based on model preview control of 3D COM accelerations," *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2016.

S. Caron, Q. Pham, and Y. Nakamura, "Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics," in *Robotics: Science and Systems (RSS)*, 2015.

S. Caron, A. Escande, L. Lanari, and B. Mallein, "Capturability-based pattern generation for walking with variable height," Mar. 2019, submitted. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01689331

J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2003.

A. Chilian and H. Hirschmuller, "Stereo camera based navigation of mobile robots on rough terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

F. Chollet *et al.*, "Keras," https://keras.io, 2015.

S. Chung and O. Khatib, "Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

D. Coleman, I. Sucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners," in *ICRA*, 2015.

C. Cunningham, W. L. Whittaker, and I. A. Nesnas, "Improving slip prediction on mars using thermal inertia measurements," in *Robotics: Science and Systems (RSS)*, 2017.

H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2016.

R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014.

A. Del Prete, S. Tonneau, and N. Mansard, "Zero step capturability for legged robots in multicontact," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1021–1034, Aug 2018.

R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, 2010.

A. Dornbush, K. Vijayakumar, S. Bardapurkar, F. Islam, and M. Likhachev, "A Single-Planner Approach to Multi-Modal Humanoid Mobility," *ArXiv e-prints*, Jan. 2018.

J. Englsberger, C. Ott, and A. Albu-Schffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, April 2015.

A. Escande, A. Kheddar, S. Miossec, and S. Garsault, "Planning support contact-points for acyclic motions and experiments on hrp-2," in *Experimental Robotics*, 2009, pp. 293–302.

P. Fernbach, S. Tonneau, A. D. Prete, and M. Tax, "A kinodynamic steering-method for legged multi-contact locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

P. Fernbach, S. Tonneau, and M. Taïx, "CROC: Convex resolution of centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

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

R. J. Griffin, G. Wiedebach, S. Bertrand, A. Leonessa, and J. Pratt, "Walking stabilization using step timing and location adjustment on the humanoid robot, atlas," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 667–673.

L. Gurobi Optimization, "Gurobi optimizer reference manual," 2018. [Online]. Available: http://www.gurobi.com

K. Hauser, T. Bretl, K. Harada, and J. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2006.

T. Hermann *et al.*, "frugally-deep," https://github.com/Dobiasd/frugally-deep, 2016.

H. Herr and M. Popovic, "Angular momentum in human walking," in *Journal of Experimental Biology*, 2008, pp. 467–481.

A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016. [Online]. Available: http://arxiv.org/abs/1605.08571

J. K. Hodgins and M. N. Raibert, "Adjusting step length for rough terrain locomotion," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 289–298, 1991.

P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics - Theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977.

A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012.

A. Ibanez, P. Bidaud, and V. Padois, "Emergence of humanoid walking behaviors from mixed-integer model predictive control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, 2014.

N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," in *Autonomous Robots*, 2013.

S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Resolved momentum control: humanoid motion planning based on the linear and angular momentum," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

O. Kanoun, E. Yoshida, and J. P. Laumond, "An optimization formulation for footsteps planning," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2009.

J. Kim, N. S. Pollard, and C. G. Atkeson, "Quadratic encoding of optimized humanoid walking," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

C. Knabe, J. Seminatore, J. Webb, M. Hopkins, T. Furukawa, A. Leonessa, and B. Lattimer, "Design of a series elastic humanoid for the darpa robotics challenge," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015.

N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

T. Koolen, M. Posa, and R. Tedrake, "Balance control using center of mass height variation: Limitations imposed by unilateral contact," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 8–15.

T. Koolen, T. de Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012. [Online]. Available: https://doi.org/10.1177/0278364912452673

J. Kuffner, "Effective sampling and distance metrics for 3d rigid body path planning," in *ICRA*, 2004.

J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Footstep planning among obstacles for biped robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

M. Lau and J. Kuffner, "Precomputed search trees: planning for interactive goal-driven animation," in *SCA*, 2006.

Y. Lin and D. Berenson, "Humanoid navigation planning in large unstructured environments using traversability-based segmentation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

——, "Using previous experience for humanoid navigation planning," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2016.

——, "Humanoid navigation in uneven terrain using learned estimates of traversability," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2017.

S. Mason, N. Rotella, S. Schaal, and L. Righetti, "An MPC Walking Framework With External Contact Forces," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade, "Vision-guided humanoid footstep planning for dynamic environments," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2005.

M. Morisawa, S. Kajita, F. Kanehiro, K. Kaneko, K. Miura, and K. Yokoi, "Balance control based on capture point error compensation for biped walking on uneven terrain," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, Nov 2012, pp. 734–740.

H. Myung, J. Kuffner, and T. Kanade, "Efficient two-phase 3d motion planning for small fixed-wing uavs," in *ICRA*, 2007.

Q. Nguyen, X. Da, J. W. Grizzle, and K. Sreenath, "Dynamic walking on stepping stones with gait library and control barrier functions," *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pp. 384–399, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-43089-4_25

D. Orin, A. Goswami, and S. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, no. 2-3, pp. 161–176, Oct. 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-013-9341-4

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

B. Ponton, A. Herzog, A. D. Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

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*, Dec 2006, pp. 200–207.

J. E. Pratt and S. V. Drakunov, "Derivation and application of a conserved orbital energy for the inverted pendulum bipedal walking model," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 4653–4660.

A. D. Prete, S. Tonneau, and N. Mansard, "Fast algorithms to test robust static equilibrium for legged robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

O. E. Ramos and K. Hauser, "Generalizations of the capture point to nonlinear center of mass paths and uneven terrain," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 851–858.

S. Schaal, "The sl simulation and real-time control software package," University of Southern California, Los Angeles, CA, Tech. Rep., 2009, clmc. [Online]. Available: http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf

J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems (RSS)*, 2013.

M. Shneier, T. Chang, T. Hong, W. Shackleford, R. Bostelman, and J. S. Albus, "Learning traversability models for autonomous mobile vehicles," *Autonomous Robots*, vol. 24, no. 1, pp. 69–86, Jan 2008.

B. Suger, B. Steder, and W. Burgard, "Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3d-lidar data," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

T. Sugihara, "Standing stabilizability and stepping maneuver in planar bipedalism based on the best com-zmp regulator," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 1966–1971.

T. Takenaka, T. Matsumoto, T. Yoshiike, and S. Shirokura, "Real time motion generation and control for biped robot -2nd report: Running gait pattern generation-," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 1092–1099.

S. Tonneau, A. D. Prete, J. Pettr, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

R. Valenzano, N. R. Sturtevant, J. Schaeffer, and F. Xie, "A comparison of knowledge-based gbfs enhancements and knowledge-free exploration," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

J. van den Berg, R. Shah, A. Huang, and K. Goldberg, "Anytime nonparametric A*," in *AAAI*, 2011.

A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006.

S. Wang and K. Hauser, "Unified multi-contact fall mitigation planning for humanoids via contact transition tree optimization," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2018.

M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

P. Wieber, "Holonomy and nonholonomy in the dynamics of articulated motion," *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*, pp. 411–425, 2006.

——, "Viability and predictive control for safe locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.