Autonomous Robots
https://doi.org/10.1007/s10514-021-09996-3

Long-horizon humanoid navigation planning using traversability estimates and previous experience

Yu-Chi Lin[1,2] · **Dmitry Berenson**[3]

Received: 15 February 2020 / Accepted: 9 May 2021

Abstract
Humanoids' abilities to navigate stairs and uneven terrain make them well-suited for disaster response efforts. However, humanoid navigation in such environments is currently limited by the capabilities of navigation planners. Such planners typically consider only footstep locations, but planning with palm contacts may be necessary to cross a gap, avoid an obstacle, or maintain balance. However, considering palm contacts greatly increases the branching factor of the search, leading to impractical planning times for large environments. Planning a contact transition sequence in a large environment is important because it verifies that the robot will be able to reach a given goal. In previous work we explored using library-based methods to address 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 discrete planners when an area is easy to traverse and switch to the library-based method only when traversal becomes difficult. Thus, in this paper we present a method that (1) Plans a torso guiding 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 a discrete-search planner, while other segments are assigned a library-based method that fits existing motion plans to the environment near the given segment. Our results suggest that the proposed approach greatly outperforms standard discrete planning in success rate and planning time. We also show an application of the method to a real robot in a mock disaster scenario.

Keywords Motion planning · Humanoid robots · Multi contact locomotion planning

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 paper focuses on constructing navigation plans for a humanoid in such large unstructured environments (see Fig. 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 (Scherer et al. 2012; Fang et al. 2017) before the humanoid enters. Long-horizon planning is especially important for disaster-response because it can save significant execution time by allowing the robot to avoid being caught in cul-de-sacs and needing to backtrack.

In disaster environments the terrain is often uneven and 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 conventional discrete-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 impose stricter contact reachability constraints. The contacts are also

✉ Yu-Chi Lin
linyuchi@umich.edu

Dmitry Berenson
berenson@eecs.umich.edu

[1] Robotics Program, University of Michigan, Ann Arbor 48109, MI, USA

[2] Nuro, Inc., Mountain View, CA 94043, USA

[3] Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor 48109, MI, USA

Published online: 27 June 2021

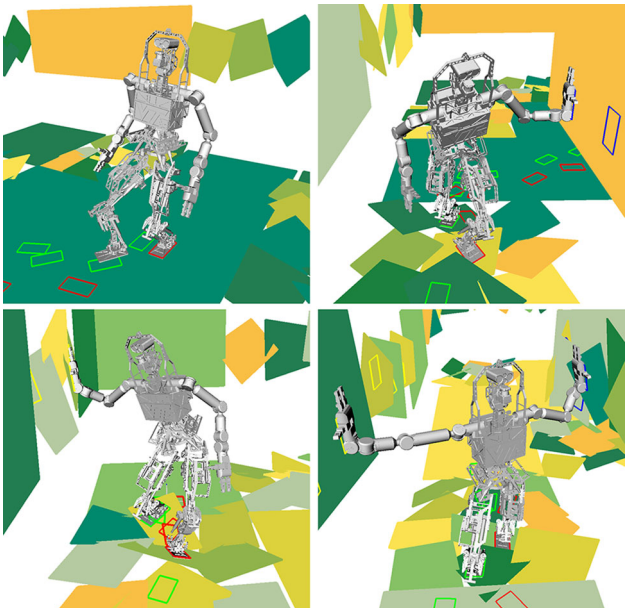


Fig. 1 Using different motion modes to traverse unstructured environments

usually non-coplanar, which prevents the use of a planar balance checking method, such as the support polygon, and thus makes it hard to evaluate the balance constraint. Conventional discrete-search-based planners explore contacts based on a heuristic which is agnostic to contact reachability and balance constraints, and could cause the planner to be trapped in a cul-de-sac. Therefore, in our framework, we propose two major contributions to aid with contact-space planning:

First, we introduce a new heuristic for search-based contact-space planners that allows the planner to avoid difficult regions in the environment. To compute this heuristic, we propose to learn a function which predicts the number of useful footstep transitions in a given region, where *useful* is defined as producing a significant motion in a given direction.Traversability indicates how difficult it is for the planner to find contacts in the environment. By including a learned traversability estimate into the heuristic function of the contact-space planner, we can bias the planner to search the areas with more contactable regions, and thus find contact transition sequences more efficiently.

Second, we propose the use of library-based methods to avoid search-based planning when possible. With a set of previously-generated robot locomotion trajectories, we can optimize those trajectories one at a time to check if they obey kinematic constraints in a given portion of the environment until a feasible one is found. This approach directly evaluates an entire trajectory, which can be much more efficient than search-based planning in difficult-to-traverse parts of the environment. However, checking trajectories can be slower than search-based planning in easy-to-traverse parts of the environment. To maximize efficiency, we would like to use discrete-search, which we call Planning from Scratch (PFS), to traverse easier areas and switch to the library-based method, which we call Retrieve and Adapt (RA), when traversal becomes difficult.

The key to our framework is determining where to use PFS or RA over a long planning horizon. An overview of our framework is shown in Fig. 2. We begin by planning a guiding path with a simplified torso pose model using traversability estimates to guide the search for contact placements. Since palm contacts may not be available in all locations and sometimes they may be unnecessary, the torso planner also specifies the motion mode (i.e. different combinations of palms and feet) to use along the path. We then segment the guiding path into motion modes based on traversability predictions for each mode. Then, We 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.

Our results on randomly-generated large unstructured environments in simulation suggest that using our framework greatly outperforms standard discrete planning in terms of success rate and planning time. We then test our method on a real mobile manipulator to demonstrate a practical application. We have also released our code open source.[1]

This paper combines and extends the work appearing in Lin and Berenson (2017) and Lin and Berenson (2018). Specifically, this paper integrates the above papers to provide more details of the proposed framework, expands on related work, and further demonstrates the proposed approach on a real robot platform.

2 Related work

Humanoid footstep planning has been studied extensively: Kuffner et al. (2001), Chestnutt et al. (2003), Michel et al. (2005), Baudouin et al. (2011), Hornung et al. (2012), Maier et al. (2013) and Griffin et al. (2019) are discrete-search-based approaches which formulate the footstep planning problem as a graph search problem. There are also optimization-based footstep planners which deform a footstep sequence to obey constraints (Kanoun et al. 2009; Deits and Tedrake 2014). These approaches consider locomotion on flat or piecewise-flat ground using only foot contacts. In our work, we deal with uneven terrains, and use not only the foot contacts, but also palm contacts to help balance the robot.

[1] https://github.com/UM-ARM-Lab/Traversability-Based-Contact-Space-Planner.

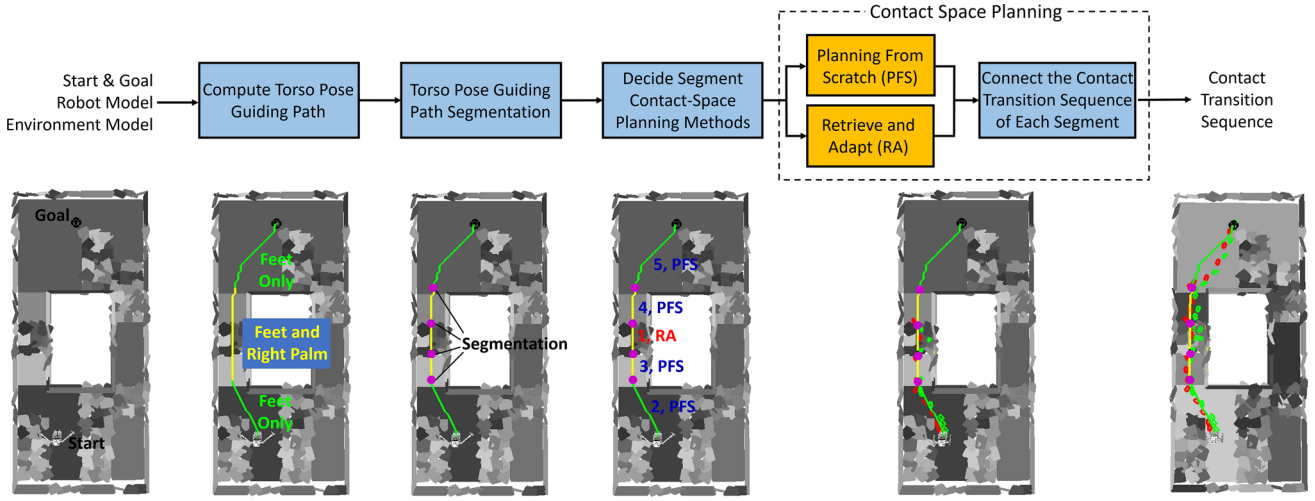


Fig. 2 Illustration of the procedure of the proposed framework. Left to Right: (1) A given environment; (2) A torso pose guiding path is planned, specifying the motion mode to use in each region; (3) The guiding path is segmented by changes in motion mode and traversability; (4) Each segment is assigned a planning method (PFS or RA) based on the traversability of the segment; (4) The contact transition sequence is generated with the specified planner for each segment; (5) The segments are connected, producing the final contact-space plan

There is also work which focuses on humanoid navigation in unstructured environments using multiple contacts. Escande et al. (2009) used optimization to find contacts in the neighborhood of a "rough" trajectory. However, the planning time was prohibitively long. Chung and Khatib (2015) combined discrete-search-based contact-space planning with a local trajectory optimizer to quickly compute a whole-body trajectory using multiple contacts. Tonneau et al. (2018) utilized the robot reachability volume to generate a guiding path to be close to possible contact locations, and then planned for contact placement along this path, which significantly sped up the planning process. Kumagai et al. (2019) extend Tonneau et al. (2018)'s work to improve its efficiency by further decomposing the contact-space planning to first find the footsteps along the guiding path, and then decide the hand contacts with approximated balance and reachability constraints. We share the idea of using a guiding path to reduce the search space. However, the above work either defines which motion mode will be used before planning or favors the use of hand contact when it is available. In our work, we focus on the decision of using different motion modes to deal with environments with varying contact options and difficulty.

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. Wellhausen et al. (2019) apply similar ideas on quadruped robots. They learn a mapping between the terrain images and the terrain reaction forces, and use this information to select paths going through preferred terrain types. In our work, the traversability does not measure the effect of the texture of the terrains on 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 traversability of the robot in the environment.

There has also been recent work addressing humanoid or quadruped 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. Grey et al. (2016) further extends this framework to include different motion modes, such as crawling and jumping. We share the same idea of using different motion modes and planners based on the geometry of the environment. However, while they assume that the robot is on flat ground with rich contacts, our work focuses on finding contact placements in a geometrically complex environment. Dornbus 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. Brandao et al. (2019) proposed a quadruped motion planner which decides not only the path, but also the applied controller to traverse through each region based on a combination of energy and feasibility costs. This is similar to our strategy of choosing different motion modes for different parts of a path, though our method for segmentation and our planning methods are different.

3 Definitions

In this section, we define terms that will be used in this paper.

- Contact pose: The pose of an end-effector in contact, specified in $SE(3)$.
- Stance: A set of contact poses of a robot.
- Contact transition: The transition from one stance to another by moving one end-effector to a new contact pose or breaking a palm contact.
- Contact transition sequence: A sequence of contact transitions.
- Torso pose: The pose of the simplified robot model derived from a stance as the mean pose of the feet in $SE(2)$.

4 Problem statement

We address the humanoid contact-space navigation planning problem. We first define the environment $\mathbf{E}$ as a set of contactable polygonal surfaces. Let $\mathbf{s}$ be a stance such that every end-effector is in contact and within the boundary of one of the polygonal surface in $\mathbf{E}$. We wish to output a *feasible* contact transition sequence from the start stance to a goal region, specified in $SE(2)$, in the workspace as quickly as possible. For a contact transition sequence to be feasible, the robot must obey quasi-static balance and collision constraints at all times when executing this contact transition sequence. We call searching for a feasible contact transition sequence "contact-space planning." We assume that the robot should always use the foot contacts, but can choose to use one 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 of each surface are given to check quasi-static balance.

5 Method overview

Our framework is depicted in Fig. 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 transition 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 the two contact-space planning approaches used in this work: PFS and RA. We then introduce how we compute the torso pose guiding path, and how traversability for different motion modes is estimated to guide the search in contact-space. Finally, we introduce the segmentation algorithm and describe how segments are ordered in search and how contact transition sequences of each segment are connected.

6 The planning from scratch (PFS) approach

In PFS, we formulate the contact-space planning problem as a graph search problem. We solve the planning problem using the Anytime Nonparametrtic A* (ANA*) algorithm (van den Berg et al. 2011) because anytime or weighted A* has been applied in many footstep planners and deployed on real robots to find footstep sequences efficiently (Hornung et al. 2012; Maier et al. 2013; Griffin et al. 2019). Each state in PFS is a stance represented as a set of contacting end-effector poses, and an action is either shifting one end-effector to a new contact pose, or breaking one palm contact. The contacts are

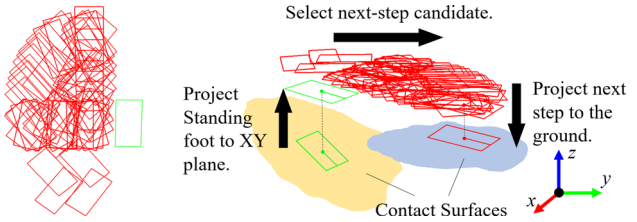


Fig. 3 Left: Foot contact transition model, $\mathbf{FC_1}$. (57 steps) Right: The projections to get the next step pose. The candidate foot poses are projected to surfaces by first finding z on the projected surface based on the (x, y) position of the candidate footstep, and then setting the projected footstep's pointing direction by taking the direction of $(\mathbf{n} \times \mathbf{f}) \times \mathbf{n}$, where $\mathbf{n}$ is the normal of the projected surface and $\mathbf{f}$ is the candidate footstep's pointing direction before projection

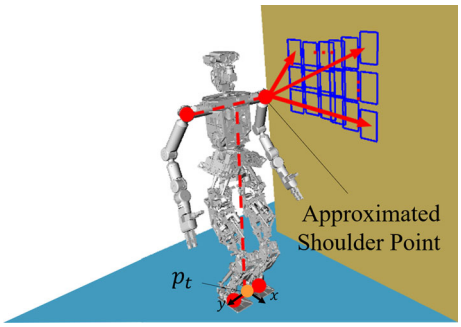


Fig. 4 Palm contact transition model, $\mathbf{PC_1}$, represented as a set of projection vector from the approximated shoulder point. Given p_t, we first identify approximated shoulder point with a fixed transform from p_t, and then project palm contact from the shoulder points using projection vectors in $\mathbf{PC_1}$

shifted to new poses based on a predefined discrete transition model. For feet contact, we project the moving foot onto the environment relative to the standing foot pose, as shown in Fig. 3. For palm contact, we first define an approximate torso pose p_t as

$$p_t = \left[\frac{x_{lf}+x_{rf}}{2} \;\; \frac{y_{lf}+y_{rf}}{2} \;\; \frac{z_{lf}+z_{rf}}{2} \;\; 0 \;\; 0 \;\; \frac{\theta_{lf}+\theta_{rf}}{2}\right]^T \tag{1}$$

$[x_{lf}, y_{lf}, z_{lf}]$ and $[x_{rf}, y_{rf}, z_{rf}]$ are the left and right foot positions, respectively, and θ_{lf} and θ_{rf} are the rotations of each foot about the z axis. We can then derive approximated shoulder points based on the approximated torso pose. The palm contacts are projected from each approximated shoulder point to the environment with a predefined set of projection vector, as shown in Fig. 4.

Since our PFS is a search-based planner, a cost is required for each action. For the foot action, we define the cost function as $\Delta g_f = d_t + w_s$, where d_t is the distance the approximated torso travels in this action and w_s is a fixed cost of taking a step. For the palm action, the cost is $\Delta g_p = d_p + w_s$, where d_p is the distance the palm travels in this action. Each state is feasible if there exists a collision-free and statically-balanced inverse kinematics solution for the specified end-effector poses. We use the method described in Caron et al. (2015) to verify quasi-static balance at each configuration, 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. We call this the end-point balance constraint.

For the heuristic of the contact-space planner, we could use Euclidean distance from the contact poses to the goal. However, this approach can easily lead the planner into a cul-de-sac. Instead, we compute a heuristic function g_{tp} based on a policy computed for a simplification of the planning problem (i.e. planning only for the torso). This torso policy is pre-computed before the contact-space planner is run and consists of a grid of cells in $SE(2)$, each with a path to the goal for the torso. We use this heuristic because it allows us to consider obstacles and gaps in the environment. We explain the computation of g_{tp} in detail in Sect. 9 where we introduce torso pose guiding path planning.

Since the torso policy used to compute g_{tp} 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-space planner's heuristic as:

$$\begin{aligned} h_{p,lp}(p_t) &= l_{lp}(P_{tp,g}(p_t)) + w_s \frac{l_{lp}(P_{tp,g}(p_t))}{d_{lp,max}} \\ h_{p,rp}(p_t) &= l_{rp}(P_{tp,g}(p_t)) + w_s \frac{l_{rp}(P_{tp,g}(p_t))}{d_{rp,max}} \end{aligned} \tag{2}$$

where $P_{tp,g}(p_t)$ is the path from the cell containing the torso pose p_t (as generated from the contact state (stance) by Eq. 1) to the goal in the torso policy, l_{lp} is the length of the portion of $P_{tp,g}(p_t)$ where it is possible to make left 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 motion mode m, we define the palm heuristic $h_p(p_t, m)$ 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: $h(p_t, m) = g_{tp}(p_t, m) + h_p(p_t, m)$.

7 The retrieve and adapt (RA) approach

While PFS works well in environments with abundant contacts, its performance depends heavily on the heuristic and

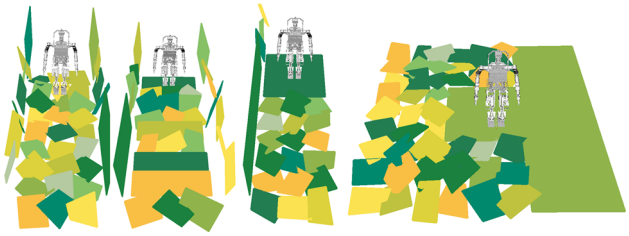


Fig. 5 Several example environments used to collect the motion plans to construct the motion plan library

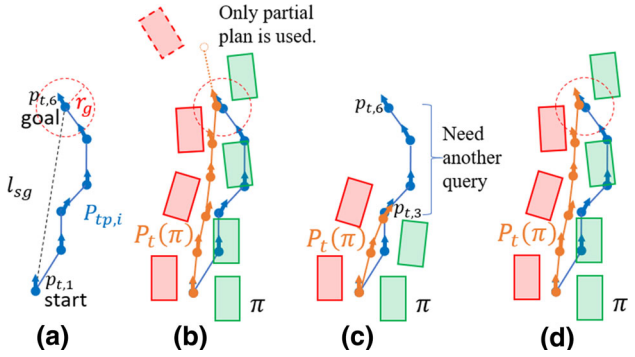


Fig. 6 Different cases of motion plan library query. Red and green rectangles represent the contact poses of motion plans. **a** A segment of torso pose guiding path. **b** The motion plan is too long compared to the torso pose guiding path. Partial motion plan is extracted to match the torso pose guiding path. **c** The motion plan is so short that it can cover only part of the torso pose guiding path. It will need another query from torso pose $p_{t,3}$ to $p_{t,6}$. **d** The motion plan matches the torso pose guiding path with $l_{sg} - r_g \leq l_{mp} \leq l_{sg} + r_g$. We use full motion plan in this case. It is a special case of **b**.

the search can get stuck in a cul-de-sac when the states suggested by the heuristic are infeasible, such as going through a region with limited contacts. PFS also requires a finely discretized transition model to navigate through such region. Therefore, in Lin and Berenson (2016), we proposed a planning method based on trajectory optimization, which locally deforms an existing humanoid joint-space trajectory (i.e. a motion plan) so that the contacts of this trajectory lie on surfaces in a new environment. This method is particularly well-suited for spaces with few accessible contacts if the motion plan we start with is relatively close to the new environment's surfaces.

To select a good initial trajectory, Lin and Berenson (2016) showed how to construct a motion plan library, sort the motion plans based on how well the contacts matched to the new environment, and finally deform motion plans one-by-one until a matching motion plan was found. In Lin and Berenson (2018) and this work, we keep the motion plan contact pose sequence matching process presented in Lin and Berenson (2016), 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 (the inability to do this was a significant limitation of Lin and Berenson (2016)). In addition we allow connecting multiple plans to reach a distant goal by making multiple queries to the library for a single segment.

7.1 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 shows some examples. Each motion plan π consists of a joint trajectory, the corresponding contact pose sequence $\mathcal{C}(\pi)$, and the motion plan torso path $P_t(\pi)$. $\mathcal{C}(\pi)$ is defined as

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

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 of each stance, and then project each approximated torso pose on the torso pose grid to form a torso path:

$$P_t(\pi) = \{p_k | p_k \in SE(2); k = 1, 2, ..., N_p\} \quad (4)$$

In practice, we first derive the stances corresponding to the motion plan π, and then compute $\mathcal{C}(\pi)$ and $P_t(\pi)$ accordingly. When matching a motion plan to a start and a goal specified as torso pose in $SE(2)$, P_t provides a mapping between the contact pose sequence and its location on the torso pose grid. Therefore, $P_t(\pi)$ can help extract partial contact pose 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_1 and all torso poses $p_k \in P_t(\pi)$.

$$D(\pi) = \{d_k | d_k = d(p_1, p_k), d_2 \leq, ..., \leq d_{N_p}, p_k \in P_t\} \quad (5)$$

We force d_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.

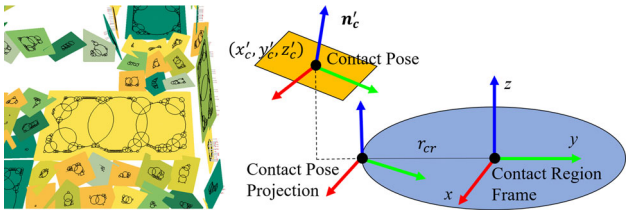


Fig. 7 Left: Extracted contact region sampling on the surfaces. Right: Contact pose versus contact region distance

7.2 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 $P_{tp,i}$, as shown in Fig. 6a. We denote the Euclidean distance in position 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 any $d_j \in D(\pi)$ of a motion plan is greater than $l_{sg} - r_g$ and less than $l_{sg} + r_g$, the motion plan has the potential to move the robot from the start to the goal region. Therefore, we 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 first 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, as shown in Fig. 6b. 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, as shown in Fig. 6c. 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 achieve this, we first adopt the method in Chung and Khatib (2015) and Lin and Berenson (2016) to extract contact regions from the environment, as shown in Fig. 7. The contact regions are a set of overlapping circular regions which mark where the robot can place its end-effectors. We denote the set of contact regions as CR, and define the distance between a contact pose $\mathbf{c}$ and a contact region $cr \in CR$ as:

$$\begin{aligned} \gamma(\mathbf{c}, cr) &= \sqrt{d_{xy}^2 + d_z^2} + w_r d_{ori} \\ d_{xy} &= max\left(0, \left|\left(x'_c, y'_c\right)\right| - r_{cr}\right), \; d_z = \left|z'_c\right| \\ d_{ori} &= 1 - \mathbf{n}'_c \cdot [0, 0, 1]^T \end{aligned} \tag{6}$$

where $\left(x'_c, y'_c, z'_c\right)$ 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 the contact region radius. 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 Fig. 7.

Before deformation of the contact poses, we treat the contact pose sequence $\mathcal{C}(\pi_e)$ 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 $\left(x_{rp}, y_{rp}, z_{rp}, \theta_{rp}\right)$, we wish 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 approach. This approach "snaps" the rigid plan to the nearest set of contact surfaces. 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:

$$\begin{aligned} &x_{0,rp} = x_s; \; y_{0,rp} = y_s; \; z_{0,rp} = z_s \\ &\theta_{0,rp} = atan2(y_g - y_s, x_g - x_s) \end{aligned} \tag{7}$$

After initialization, we iteratively update $\mathbf{T}_{rp}$ using its Jacobian to move the rigid plan's $\mathcal{C}(\pi_e)$ closer to their nearest contact regions. At each iteration, we find $cr_{min,i}$, the closest contact region to $\langle \mathbf{c}_i, e_i \rangle \in \mathcal{C}(\pi_e)$. 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 $\mathbf{c}_i$. By combining the Jacobians for all $\mathbf{c}_i$, we have J such that:

$$\begin{bmatrix} \dot{\mathbf{c}}_1 \\ \dot{\mathbf{c}}_2 \\ \vdots \\ \dot{\mathbf{c}}_{N_e} \end{bmatrix} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_{N_e} \end{bmatrix} \dot{\mathbf{T}}_{rp} = J \dot{\mathbf{T}}_{rp} \tag{8}$$

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

$$\dot{\mathbf{T}}_{rp} = J^+ \left[\dot{\mathbf{c}}_1{}^T, \dot{\mathbf{c}}_2{}^T, ..., \dot{\mathbf{c}}_{N_e}{}^T\right]^T \tag{9}$$

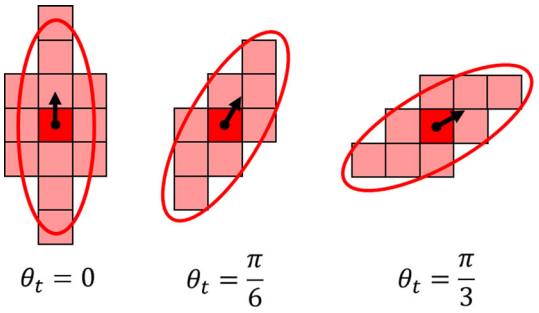


Fig. 8 Torso pose transition model. The position and orientation resolution used by this work is 0.135 m and $\frac{\pi}{6}$ radian, respectively. The pink cells show where the torso can move from the red cell based on its orientation in one step. In torso pose guiding path planning, for each step, the torso is allowed to change its orientation by $\pm\frac{\pi}{6}$ and move to a position cell specified by the current orientation. Torso pose transition model for other orientations can be derived from the above transition models and rotate the grid by $n\frac{\pi}{2}$ radian, $n \in \mathbb{N}$

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

$$\Gamma(\mathcal{C}(\pi_e), CR) = \frac{1}{N_e}\sum_{i=1}^{N_e}\gamma\left(\mathbf{c}'_i, cr_{min,i}\right) \tag{10}$$

where $\mathbf{c}'_i$ is the ith contact pose in $\mathcal{C}(\pi_e)$ transformed by $\mathbf{T}'_{rp}$.

For each converged rigid plan $\mathbf{T}'_{rp}$, 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 Contact-Consistent Elastic Strips (CES) (Chung and Khatib 2015). Each configuration of the trajectory will move 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 transition sequence follows the end-point balance constraints. If the check passes, RA will output this final motion plan as the result.

8 Estimating traversability

In the previous two sections, we first introduced PFS, a discrete search-based planner. We then described RA, which chooses and locally deforms joint trajectories to fit a motion plan to a given environment. PFS is more efficient when there are many accessible contacts, but RA can handle more difficult environments where contacts are limited. To get the best of both worlds, we would like to determine which method (PFS or RA) to use to efficiently plan a contact sequence through different parts of a large environment. Our approach to this is to estimate how difficult it will be to find useful contact transitions in a given region of the environment and use that estimate to select a planning method for that region. We describe how to compute this estimate below:

Traversability is defined as the number of useful footstep transitions in a given region, where *useful* is defined as producing a significant motion in a given direction. Intuitively, if there are many useful footstep transitions in a given region, the PFS planner is much more likely to find a path through that region. Thus if the planner knows which region has a higher traversability before planning, it can bias its search to avoid difficult regions via torso pose guiding path planning, and generate a contact transition sequence more quickly. Therefore, for any given torso pose p_t in an environment $\mathbf{E}$, we define traversability as $|\Gamma_+| : \{\mathbf{v}, m\} \rightarrow \mathbb{R}_+$, where $\mathbf{v}$ is a 2D torso translation in the XY plane, m is the motion mode, and $\mathbf{E}$ is expressed as the set of planar contact surfaces. For each p_t, We use a finite set of $\mathbf{v}$, as shown in Fig. 8.

8.1 Traversability measure

To calculate the traversability, we start by finding a set of feasible footstep combinations Γ at p_t. This process of finding Γ corresponds to line 2 in Algorithm 1 To compute Γ, we first use the foot contact transition model $\mathbf{FC_1}$ shown in Fig. 3 to find possible footstep combinations centered around $p_t = (x_t, y_t, \theta_t)$. For each footstep in the foot contact transition model, the left foot pose is specified relative to the right foot pose in $SE(2)$: $\left(x^{rf}_{lf}, y^{rf}_{lf}, \theta^{rf}_{lf}\right)$. We define each possible footstep combination pose centered around p_t as:

$$\begin{aligned}
\theta_{lf} &= \theta_t + \frac{1}{2}\theta^{rf}_{lf}\ ;\ \theta_{rf} = \theta_t - \frac{1}{2}\theta^{rf}_{lf} \\
\begin{bmatrix} x_{lf} \\ y_{lf} \end{bmatrix} &= \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{bmatrix}\begin{bmatrix} x^{rf}_{lf} \\ y^{rf}_{lf} \end{bmatrix} \\
\begin{bmatrix} x_{rf} \\ y_{rf} \end{bmatrix} &= \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \frac{1}{2}\begin{bmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{bmatrix}\begin{bmatrix} x^{rf}_{lf} \\ y^{rf}_{lf} \end{bmatrix}
\end{aligned} \tag{11}$$

where $(x_{lf}, y_{lf}, \theta_{lf})$ and $(x_{rf}, y_{rf}, \theta_{rf})$ are left and right foot poses, respectively. Note that the torso pose p_t is the mean of the left and right feet poses in each footstep combination with this definition. These $SE(2)$ feet poses will then be projected to the environment to obtain the full $SE(3)$ pose, as shown in Fig. 3. A footstep combination is feasible if there exists a valid projection on the environment for both feet.

Given an environment $\mathbf{E}$, a motion mode m, a torso translation $\mathbf{v}$ and a starting footstep combination $\gamma \in \Gamma$, if the PFS planner can generate a single footstep transition from γ with palm contacts specified by m and moves the mean feet position to the cell to which $[p_t[x], p_t[y]]^T + \mathbf{v}$ belongs in the torso pose grid, we call such γ an *useful* footstep combination. For each γ, if m is a motion mode which requires

Algorithm 1: Compute Ground Truth Label for Traversability

Input : p_t (Torso pose), $\mathbf{v}$ (Torso translation), m (Motion mode), $\mathbf{E}$ (Environment, specified as a set of surfaces), $\mathbf{FC_1}$ (Foot contact transition model) , $\mathbf{PC_1}$ (Palm contact transition model);
$\Gamma \leftarrow$ GetFeasibleFootstepCombinations (p_t, $\mathbf{E}$, $\mathbf{FC_1}$);
$\Gamma_+ \leftarrow \{\}$;
for γ **in** Γ **do**
  if *ContactSequenceExists* (γ, $\boldsymbol{v}$, m, $\mathbf{E}$, $\mathbf{FC_1}$, $\mathbf{PC_1}$) **then**
    $\Gamma_+ \leftarrow \{\gamma\} \cup \Gamma_+$;
  end
end
return $|\Gamma_+|$;

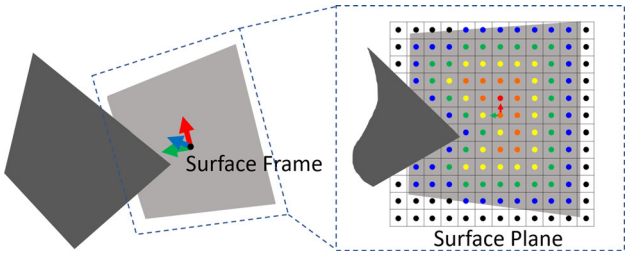


Fig. 9 The grid of contact points on a surface plane. The distance of each contact point to the closest obstacle or surface boundary is marked in color spectrum order. Note that the light gray surface is covered by the dark gray surface, which causes part of its contact points to be infeasible

palm contacts, the planner will start from a set of initial states, whose foot locations are γ, with all combinations of possible palm contacts given by the palm transition model $\mathbf{PC_1}$, as shown in Fig. 4. The above process is line 5 in Algorithm 1.

8.2 Computing contact clearance feature to estimate traversability

In this work, we quantify the traversability as the number of useful footstep combinations, denoted $|\Gamma_+|$, which serves as an indicator for how difficult it is for the planner to find a contact transition sequence in environment $\mathbf{E}$ moving from torso pose p_t with $\mathbf{v}$ torso translation using motion mode m. The process computing $|\Gamma_+|$ is summarized in Algorithm 1. However, to compute the traversability in an environment $\mathbf{E}$ would require running a PFS planner for every combination of $(p_t, \mathbf{v}, m)$, which is clearly too time-consuming. Therefore, we would like to learn a traversability estimator for each $(\mathbf{v}, m)$ pair which predicts $|\Gamma_+|$ based on *contact clearance features* extracted in the neighborhood of torso pose p_t in an environment $\mathbf{E}$. Contact clearance features represent how much surface area there is to make contact around the queried p_t making a torso pose transition for the given $\mathbf{v}$ and m. While it is possible to also consider other features, such as inclination angle, we focus on the contact clearance feature in this work because it strongly influences which planning method (PFS or RA) is likely to succeed. When the surface area is small, it is difficult for PFS to find a feasible contact within the discrete contact transition set ($\mathbf{FC_1}$ and $\mathbf{PC_1}$). However RA, which is a trajectory optimization approach, can deform the robot configurations and the contact poses continuously to place contacts in small areas.

To compute the contact clearance feature, we first discretize each surface polygon into a set of contact points C_i which form a grid. We denote the set of all contact points, which is also the union of all C_is from all surface, as C. For each contact point $c \in C$, we cast a ray along the normal of each surface for a short distance (accounting for the swing foot height of the robot) to check if the contact point is collision-free. The distance of each contact point to the closest obstacle or surface boundary, denoted as $\delta(c)$, is approximated as the closest distance to any contact point in collision or out of boundary. Figure 9 shows an example contact point grid of a surface. We can define the following scoring function to represent the clearance of each contact point:

$$S(c) = \begin{cases} 0 & \delta(c) < r_{\text{ins}} \\ \frac{\delta(c) - r_{\text{ins}}}{r_{\text{cir}} - r_{\text{ins}}} & r_{\text{ins}} \leq \delta(c) < r_{\text{cir}} \\ 1 & \delta(c) \geq r_{\text{cir}} \end{cases} \tag{12}$$

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 $\delta(c)$ is larger than r_{cir}, there must exist enough free space for any contact pose at the contact point c. However, if $\delta(c)$ is lower than r_{ins}, it is impossible to make contact at c regardless of the orientation of the contact.

S describes how likely a contact pose is feasible given its corresponding contact point c. 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 Fig. 3, 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-space planning search tree based on the transition model shown in Fig. 3 for only one step. For each expansion, we can define a footstep translation tuple α as $\alpha = \left\{\mathbf{t}^t_{lf}, \mathbf{t}^t_{rf}, \mathbf{t}^t_{ex}\right\}$. $\mathbf{t}^t_{lf}$ and $\mathbf{t}^t_{rf}$ are the 2D translation of the left foot and right foot in the torso frame in the XY plane, and $\mathbf{t}^t_{ex}$ 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. 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 $A(\mathbf{v})$.

Algorithm 2: Generate the Contact Clearance Feature Vector

Input : p_t (Torso pose), $\mathbf{v}$ (Torso translation), $\mathbf{E}$ (Environment, specified as a set of surfaces), A ($\mathbf{v}$) (Set of footstep translation tuple) , $\mathbf{PC_1}$ (Palm contact transition model), C (Set of contact points);

1 $T_{p_t} \leftarrow$ GetTransformaionMatrix (p_t);
2 $S_f \leftarrow 0$, $\mathbf{S}_p \leftarrow [0, 0, 0, 0]$;
3 **for** α **in** A ($\boldsymbol{v}$) **do**
4 $S_\alpha \leftarrow 1$;
5 **for** t **in** α **do**
6 $c_{\text{Nearest}} \leftarrow$ GetNearestContactPoint $\left(T_{p_t}\mathbf{t}, \text{C}\right)$;
7 $S_\alpha \leftarrow S_\alpha S(c_{\text{Nearest}})$;
8 **end**
9 $S_f \leftarrow S_f + S_\alpha$;
10 **end**
11 **for** pc **in** $\mathbf{PC_1}$ **do**
12 $p_{palm} \leftarrow$ GetPalmPose (p_t, pc);
13 $c_{\text{Neareset}} \leftarrow$ GetNearestContactPoint $\left(p_{palm}, \text{C}\right)$;
14 $i_q \leftarrow$ GetPalmQuadrant $\left(p_t, p_{palm}\right)$;
15 $\mathbf{S}_p[i_q] \leftarrow \mathbf{S}_p[i_q] + S(c_{\text{Neareset}})$;
16 **end**
17 **return** $\left[S_f, \mathbf{S}_p\right]$;

Lines 4 to 10 in Algorithm 2 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. 12. The multiplication shown in Line 8 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.

The palm contact direction also affects the robot's ability to move in a certain direction. For example, if there exist only palm contacts in the direction opposite to the direction of motion, the planner may not be able to find a valid contact transition sequence. Therefore, we add an additional feature: the contact clearance feature of palm contacts around p_t. We first find all palm contact projection onto the environment with a palm contact transition model $\mathbf{PC_1}$, as shown in Fig. 4. Each projection then returns a nearest contact point c on one of the surfaces. To represent the direction of the palm contacts relative to p_t, 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 10 to 14 in Algorithm 2. With $\left[S_f, S_p[1], S_p[2], S_p[3], S_p[4]\right]$, we define the contact clearance feature vector $\mathbf{S}\left(p_t, \mathbf{v}, m, \mathbf{E}\right)$ as:

$$\mathbf{S}\left(p_t, \mathbf{v}, m, \mathbf{E}\right) = \begin{cases} \left[S_f\right], & m = \text{feet only} \\ \left[S_f, S_p[1], S_p[2]\right], & m = \text{feet and left palm} \\ \left[S_f, S_p[3], S_p[4]\right], & m = \text{feet and right palm} \\ \left[S_f, S_p[1], S_p[2], S_p[3], S_p[4]\right], & m = \text{all end-effectors} \end{cases} \tag{13}$$

To train the estimator for each motion mode, we generate multiple environment with randomly tilt 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 $\Delta g_{tr}\left(p_{t,i}, p_{t,j}, m_j\right)$ as:

$$\Delta g_{tr}\left(p_{t,i}, p_{t,j}, m_j\right) = e^{-|\Gamma_+|(p_{t,i}, \mathbf{v}, m_j)} \tag{14}$$

where $\mathbf{v}$ is the torso translation closest to $[p_{t,j}[x] - p_{t,i}[x], p_{t,j}[y] - p_{t,i}[y]]^T$ and $|\Gamma_+|$ is the appropriate traversability estimate for the transition from $p_{t,i}$ to $p_{t,j}$. With this definition, higher traversability implies a lower traversability cost, and vice versa.

9 Torso pose guiding path

The purpose of computing a torso pose guiding path with a simplified model is to guide the higher-dimensional contact-space 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 paper, 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 Fig. 8. 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:

$$P_{tp} = \{p_{t,1}, p_{t,2}....., p_{t,N_p} \,|\, p_{t,1}, ..., p_{t,N_p} \in SE(2)\} \tag{15}$$

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:

$$P_{tp} = \{\left(m_1, p_{t,1}\right), \left(m_2, p_{t,2}\right), ..., \left(m_{N_p}, p_{t,N_p}\right)\} \tag{16}$$

where m_i is the motion mode used to reach torso pose $p_{t,i}$, (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.

9.1 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

$$\begin{aligned} &\Delta g_{tp}\left(\left(m_i, p_{t,i}\right), \left(m_j, p_{t,j}\right)\right) = \\ &l_{p_t,j}^{p_{t,i}} + w_s + w_{tr}\Delta g_{tr}\left(p_{t,i}, p_{t,j}, m_j\right) + M\left(m_i, m_j\right) \\ &M\left(m_i, m_j\right) = \begin{cases} 0, & m_i = m_j \\ w_m, & m_i \neq m_j \end{cases} \end{aligned} \tag{17}$$

where $l_{p_t,j}^{p_{t,i}}$ is the distance between torso pose $p_{t,i}$ and $p_{t,j}$, 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 Sect. 8), and w_{tr} is a weighting factor for Δg_{tr}. Possible actions are the combination of torso pose transitions shown in Fig. 8 and the motion modes used. The heuristic function for planning the torso pose guiding path is

$$h_{tp}\left(\left(m_i, p_{t,i}\right)\right) = d_{goal}^{t}(p_{t,i}) + w_s \frac{d_{goal}^{t}(p_{t,i})}{d_{t,max}} \tag{18}$$

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.

9.2 Torso pose policy planning for heuristic in PFS

In Sect. 6, we briefly introduced the heuristic function used in PFS in this paper. Now that we have defined the torso planning problem in the previous section, we are ready to specify the heuristic precisely: To create a useful heuristic for PFS that is aware of obstacles and gaps, we first compute a policy for the torso pose. To compute the torso pose policy, similar to the approach for planning the torso pose guiding path, we plan on the torso pose grid, and adopt the edge cost definition in Eq. 17, but use Dijkstra's algorithm to compute the length of the path going from *each cell* to the goal cell. PFS queries this policy with the torso pose p_t of a contact state it is considering to get the length of the path $l_{goal}^{t}(p_t)$ of going from the cell containing p_t to the goal cell. g_{tp} is then used to compute the heuristic of PFS for that contact state:

$$g_{tp}\left(p_t, m\right) = l_{goal}^{t}(p_t) + w_s N_s + w_{tr} g_{tr}\left(m\right) \tag{19}$$

where 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.

Since the edge cost of contact-space planning does not have cost corresponding to g_{tr} in Eq. 19, $h\left(p_t, m\right)$ is inadmissible, and the returned contact transition sequence can be suboptimal. However, in a large unstructured environment, the main challenge is to efficiently find a *feasible* solution. Therefore, our goal is to increase the success rate in finding a feasible solution. g_{tr} helps the planner identify the part of the environment richer in contact transition to traverse, and thus increase the success rate.

10 Torso pose guiding path segmentation

As mentioned in Sect. 5, 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. 16, we can extract the torso pose transition sequence $T_\delta\left(P_{tp}\right)$ defined as:

$$\begin{aligned} &T_\delta\left(P_{tp}\right) = \{\delta_1, \delta_2, ..., \delta_{N_\delta}\} \\ &\delta_i = \left(\mathbf{v}\left(p_{t,i}, p_{t,i+1}\right), \Delta\theta\left(p_{t,i}, p_{t,i+1}\right), m_{i+1}\right) \end{aligned} \tag{20}$$

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}.

Algorithm 3: Torso Pose Guiding Path Segmentation

Input : $T_\delta(P_{tp})$ (Torso pose transition sequence);
$\psi_{mm} \leftarrow$ GetMotionModeSegmentation $\left(T_\delta(P_{tp})\right)$;
$\psi^* \leftarrow \{\}$;
for $\psi_{mm}[k]$ **in** ψ_{mm} **do**
  if $|\psi_{mm}[k]| \geq N_{seg}$ **then**
    $\psi_k^* \leftarrow$ GetOptimalSegmentation $(\psi_{mm}[k])$, (Eq. 21);
  end
  else
    $\psi_k^* \leftarrow \psi_{mm}[k]$;
  end
  $\psi^* \leftarrow \psi^* \cup \psi_k^*$;
end
return ψ^*;

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}:

$$\begin{aligned} &\underset{\psi \in \Psi(\psi_{mm}[k])}{\text{argmax}} && \sum_{i=1}^{|\psi|} \left| \sum_{\delta_j \in \psi[i]} \Delta g_{tr}\left(\mathbf{v}\left(\delta_j\right), m\left(\delta_j\right)\right) - |\psi[i]|\, T_{tr} \right| \\ &\text{subject to} && |\psi[i]| \geq N_{seg} \end{aligned} \tag{21}$$

where ψ is a segmentation of the kth torso pose transition sequence, and $\psi[i]$ is the ith segment in that segmentation. Since we decide which contact transition sequence generation method is used in each segment based on its traversability, we define the traversability cost threshold, $T_{tr} \in \mathbb{R}^+$, to serves as the decision boundary. This optimization will try to generate segments whose average Δg_{tr} is above or below T_{tr} as much as possible , so that our segments consist of torso pose transitions which all have similar traversability. We also add a constraint to exclude segments that are too short (containing fewer than N_{seg} transitions). Again, it is important to reduce the number of segments for the reasons described in Sect. 9. 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. The segmentation process is summarized in Algorithm 3.

After the segmentation, the contact transition 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 transition 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 Sect. 12.

Algorithm 4: Decide Segment Exploration Order

Input : ψ^* (Optimal segmentation);
$\psi_{\text{explore}} \leftarrow \{\}$;
$\psi_{\text{PFS}} \leftarrow \{\}$;
for $\psi^*[k]$ **in** ψ^* **do**
  if $\mu(\psi^*[k]) = PFS$ **then**
    $\psi_{\text{PFS}} \leftarrow \psi_{\text{PFS}} \cup \psi^*[k]$;
  end
  else
    if $\mu(\psi^*[k]) = RA$ **then**
      $\psi_{\text{explore}} \leftarrow \psi_{\text{explore}} \cup \psi^*[k] \cup \psi_{\text{PFS}}$;
      $\psi_{\text{PFS}} \leftarrow \{\}$;
    end
  end
end
$\psi_{\text{explore}} \leftarrow \psi_{\text{explore}} \cup \psi_{\text{PFS}}$;
return ψ_{explore};

10.1 Deciding 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 RA directly follows a segment using PFS, we can generate the contact transition 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 4 shows the procedure used to decide the segment exploration order.

11 Connecting the contact sequences

As discussed in Sect. 10.1, 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 transition 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.

12 Experiments

In our experiments, we first show the performance of the traversability prediction model. We then demonstrate the performance of the proposed framework in challenging environments by comparing with a set of baselines. These baselines include simplifications of our methods (i.e. removing some components) as well as alternative cost functions for torso pose guiding path planning, inspired by Brandao et al. (2019). We evaluate performance by showing the success rate and planning time for finding a feasible contact transition sequence using each approach. We also check if the quasi-static plan produced by solving inverse kinematics for a planned contact sequence is feasible. Finally, we extend the application of our framework to a mobile manipulator, and show the advantage of the proposed framework using traversability estimates in a real robot experiment.

The experiments were run on an Intel Core i7-4790K 4.40 GHz CPU with 16GB RAM, and we implemented our algorithms in OpenRAVE (Diankov 2010). We tested our algorithms on the Escher humanoid robot model (Knabe et al. 2015). Escher has 6 and 7 degrees of freedom (DOF) for each leg and arm, respectively, and 1 waist yaw DOF.

12.1 Traversability prediction models' performance

In this section, we show the performance of the traversability prediction models of different motion modes. Although there are four motion modes, we take advantage of the symmetric geometry of humanoid robots, and train three kinds of traversability prediciton models: Feet Only, Feet and One Palm and All End-Effectors with 10,000 transitions for each model. We tested the models in randomly-tilted surface environments shown in Fig. 5, and each model is tested with 100 torso pose transitions. The traversability prediciton models predict the number of useful footstep combinations $|\Gamma_+|$ given the surrounding environment. Since the traversability prediction models are regressors, we show their mean absolute error compared to the range of the ground truth in test data, as shown in Table 1. Clearly the error is quite small relative to the range of data.

12.2 Results for the full framework and comparisons

We evaluate the performance of the proposed framework in planning to navigate through two types of environments and we compare the proposed framework with three simplifications: The first baseline is the most basic contact-space PFS planner. It does not use traversability and has no specified motion mode (PFS Only). The second baseline is also PFS, but now we allow it to use traversability in torso pose guiding path planning, however we do not specify the motion mode (PFS using Traversability). Since these planners are not required to use any palm contacts, they use the feet-only motion mode heuristic to estimate the cost-to-go. The third baseline uses our segmentation approach but only uses PFS to plan motion in each segment. Since it only uses PFS, we segment the torso pose guiding path only when motion mode changes (Segmentation+PFS). For the 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 also show comparisons with variants of the proposed framework inspired by Brandao et al. (2019). Brandao et al. (2019) presented a framework which generates guiding paths for a quadruped robot to traverse different regions using different controllers. While the idea is similar to our approach, which can use different motion modes for different regions, the cost definition in the guiding path planning is fundamentally different and thus we cannot present a side-by-side comparison. However, to avoid cost weight tuning, Brandao et al. (2019) also emphasized using a cost aggregation method which normalizes each term in edge cost by its range. Each cost term is thus between 0 and 1. Thus, inspired by Brandao et al. (2019), we create variants of the proposed approach (Segmentation+PFS, Our Framework-Max and Our Framework-Mean) by normalizing each cost term in Eq. 18 to be from 0 to 1 (NC Segmentation+PFS, Our Framework-NC-Max and Our Framework-NC-Mean). Comparing to this cost normalization approach helps show the degree of sensitivity of our framework to the tuning of weights in the cost function.

We use the following parameter values: $N_{mp} = 50$, $T_{tr} = 0.3$, $N_{seg} = 5$, $w_s = 3$, $w_m = 2$, $w_{tr} = 10$, $r_g = 0.2m$. The torso path grid is discretized to 0.15m resolution in x and y, and 30° in θ.

In Sect. 6, we described that the PFS planner uses endpoint balance constraints to check balance. To verify if this simplification is reasonable, we did the following: For all

Table 1 Traversability prediction models performance (units are the number of useful footstep combinations)

	Feet only	Feet and one palm	All end-effectors
Mean absolute error	1.09	1.53	1.31
Ground truth range	[0, 32]	[0, 26]	[0,28]

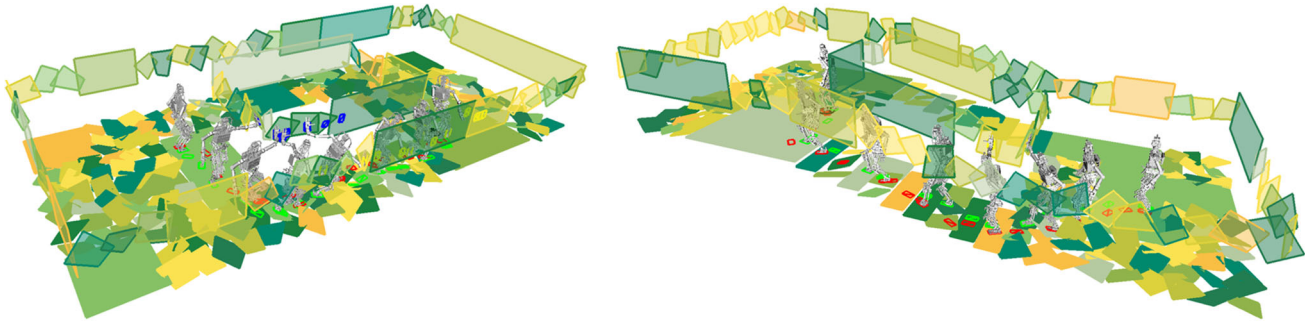


Fig. 10 Executing a planned contact transition sequence in Left: a two-corridor environment. Right: a two-staircase environment

the contact transition sequences returned by any method we evaluated, we test if a quasi-statically balanced configuration-space path can be found for each contact transition. We ran this test by enumerating many different parabolic paths for the swing end-effector of each contact transition, solving inverse kinematics for each parabola, and checking if there exists a quasi-statically balanced path. The results show that we can find quasi-statically balanced configuration paths in 496 out of 498 contact transition sequences returned by all baselines and the proposed framework in two test environments, as shown in Table 2. This high success-rate suggests that we are indeed making reasonable assumptions about the feasibility of our transitions in difficult environments. The two failed cases are caused by the robot attempting a long foot contact transition with a very tilted standing foot contact and a palm contact far from the new foot contact. Although palm contacts help balance the robot, it also creates more constraints on the robot's kinematic solutions, which cause the inverse kinematics to fail in some rare case. We believe that the problem can be addressed with a more conservative contact transition model.

12.2.1 Two-corridor environment test

In the two-corridor environment, we construct the environment as two wide rooms connected with two parallel corridors (see Fig. 10). 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 transition 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 2).

The results show that using traversability in PFS helps improve success rate, although it is not a large improvement. Incorporating traversability to get the Segmentation+PFS approach provides a larger boost in performance, improving success rate further 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.

Compared to our framework using tuned cost weights, the normalized cost approach (NC Segmentation+PFS) performs slightly worse for Segmentation+PFS. However, when RA is used (i.e. the full framework), the performance of our cost weight and the normalized cost weight is very similar. The reason for this result is that the normalized cost approach has relatively lower weight on traversability cost compared to the tuned cost, and more segments of the guiding path go through regions which harder to traverse. Therefore, the normalized cost approach (Our Framework-NC-Mean and Our Framework-NC-Max) using the full framework has a higher ratio of the segments using RA as compared to our framework using tuned cost weights. Yet using RA for those

Table 2 The result for two-corridor test environment and two-stair test environment

Environment	Approach	Success rate	Quasi-static execution success rate	Average number of segments (PFS/RA/Total)	Planning time (s)				
					Torso path planning	Torso path segmentation	Contact space planning	Segment contact sequence connection	Total time
Two-corridor environment	PFS only	13/50	13/13	1/0/1	6.51	0	200.01	0	206.52
	PFS using traversability	17/50	17/17	1/0/1	24.05	0	114.05	0	138.10
	Segmentation+PFS	27/50	27/27	4.11/0/4.11	24.81	0.01	138.95	0	163.77
	Our framework-Mean	38/50	38/38	4.08/1.63/5.71	25.63	0.03	96.91	1.76	**124.33**
	Our framework-Max	**42/50**	41/42	2.78/2.93/5.71	26.16	0.03	109.03	17.73	152.95
	NC segmentation+PFS	23/50	23/23	3.70/0/3.70	25.74	0.01	140.51	0	166.25
	Our framework-NC-Mean	39/50	39/39	2.95/2.77/5.72	23.98	0.27	100.14	2.22	126.60
	Our framework-NC-Max	41/50	41/41	1.83/4.61/6.44	23.26	0.25	110.59	5.44	139.54
Two-staircase environment	PFS only	17/50	17/17	1/0/1	6.47	0	181.66	0	188.13
	PFS using traversability	23/50	23/23	1/0/1	24.74	0	146.05	0	170.79
	Segmentation+PFS	35/50	34/35	5.72/0/5.72	23.91	0.01	115.62	0	139.54
	Our framework-Mean	**39/50**	39/39	4.31/3.18/7.49	24.74	0.70	117.66	1.34	144.44
	Our framework-Max	38/50	38/38	2.74/4.75/7.49	24.67	0.71	108.11	5.19	138.68
	NC segmentation+PFS	32/50	32/32	4.69/0/4.69	25.37	0.01	129.93	0	155.30
	Our framework-NC-Mean	37/50	37/37	3.03/4.11/7.14	23.69	0.40	108.84	3.10	**136.03**
	Our framework-NC-Max	37/50	37/37	1.86/5.70/7.57	23.91	0.40	110.31	8.65	143.27

The bold face marks the best result among all different approaches compared in the experiments. In particular, it marks the approaches with highest success rate and approaches with shortest total time

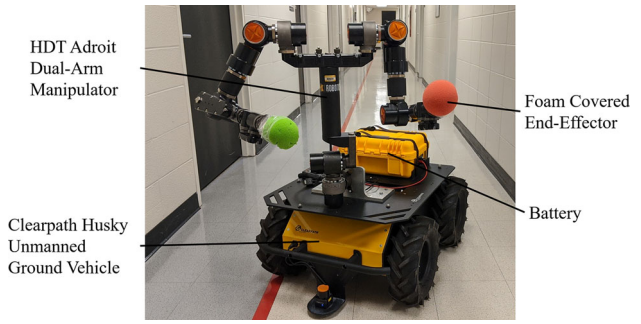


Fig. 11 The mobile manipulator used in the experiment. The end-effectors are padded with foam covers to reduce damage on the surface of the end-effector when making contacts

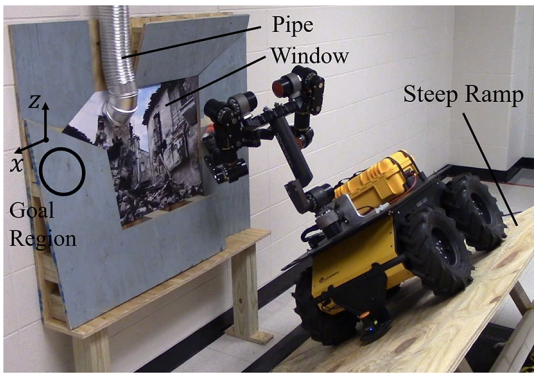


Fig. 12 The experiment setting: The mobile manipulator moves along a steep ramp while using palm contacts to stabilize itself. The robot has to find a contact transition sequence to move across the window, and can only make contact to the four cracked wall surfaces showing in grey

difficult segments is effective, so the success rate is similar to our framework using tuned cost weights. This is actually a very encouraging result, because it shows that even if we don't use human intuition to tune the weights in our cost function (and thus move through more difficult regions in the environment than necessary), our overall framework is still able to plan effective contact sequences.This result shows that the proposed approach does not require intensive manual weight tuning since automatic weighting achieves similar performance.

12.2.2 Two-staircase environment test

A two-staircase environment is shown in Fig. 10. 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 1 and 1.5 m, 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. As we observed in the two-corridor environment, the variants of the proposed framework using normalized cost (Our Framework-NC-Mean and Our Framework-NC-Max) achieve similar performance as our framework.

12.3 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 transition sequence generated by the proposed framework. We use a mobile manipulator to demonstrate a real robot motion based on a planned contact transition sequence in a disaster-response scenario. Figure 11 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 (Fig. 12). The robot has to plan contact transition 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.

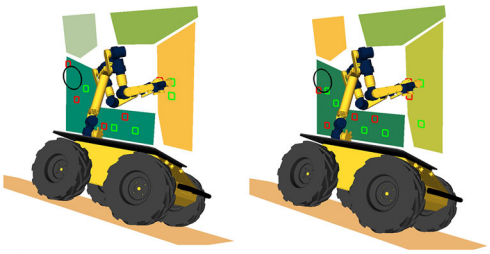
We also evaluated the end-effector position error for this robot to verify the feasibility of reliably achieving the contacts we planned. To compute the end-effector error, we used a Vicon motion-capture system to estimate the ground-truth position of the base and the end-effectors. We then generated end-effector trajectories in the the relevant part of the robot's workspace (with the robot facing to the side and leaning forward) and computed the difference between the end-effector position generated via forward kinematics on the planned joint angles and the end-effector position from the motion capture system. We found that the average end-effector error was 18.8 mm with a standard deviation of 5.08 mm for the left arm and 24.9 mm with a standard deviation of

2.75 mm for the right arm. Thus it was quite feasible to execute the end-effector trajectories computed by our method on the robot in the environment shown in Fig. 12 because we do not require a high contact placement accuracy to traverse this environment. In future work, it would be interesting to explore methods to plan for more difficult environments where precise contact is required to complete the task by accounting for the end-effector position error in planning.

To help the robot deal with actuation error in the arms, we padded the end-effectors with soft foam covers. This is similar to compliant ankles adopted in many legged robot to help the robot better comply to uneven terrains. Since the joints are position-controlled with high stiffness, this added compliance can also help prevent hard collisions. The foam also provides higher friction which helps prevent the end-effectors from sliding when the base moves.

Since the experiment requires traveling only a short distance with the base, the robot's position is tracked by the base's odometer. In larger environments, a localization method such as a Kalman or particle filter would be necessary. We also coordinated the base and upper body to maintain the end-effector positions while the base moves. To do this, we ran an inverse kinematics solver at 100Hz to compute upper-body joint goals with base poses supplied by the odometer. A PD controller is then used to track the joint goals. We exploit the friction provided by the foam cover and move the robot slowly so that the controller can prevent the end-effector from sliding. This is important because sliding can cause the robot to fall on the wall and potentially break the arms. This experiment also shows that it is quite difficult for a robot to maintain multiple contacts, especially if they do not align with the direction of gravity, and still move quickly, which is also part of the reason why executing multi-contact motion for humanoid robots is still an open problem.

We compare the proposed framework with the PFS Only planner, which does not consider traversability. The contact-space planning for the mobile manipulator is analogous to the formulation used in humanoid contact-space planning shown in Sect. 6. We transform the ground described in Sect. 6 to be the wall in the mobile manipulator experiment, and the mobile manipulator is viewed as "walking" on the wall, as shown in Fig. 12. To check quasi-static balance for each contact transition in contact-space 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 Sect. 6. The contact-space 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



	PFS Only	The Proposed Framework
Planning Time (Sec.)	3.52	**0.11**
Number of Steps	**8**	10

Fig. 13 Left: The planned contact transition sequence using PFS Only. Right: The planned contact transition sequence using the proposed framework. Left and right palm contact are shown in red and green, respectively

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 Fig. 12. Since the contact orientation is always 0 degree, the torso orientation also remains 0 degree in the XZ plane. We use the following parameter values for the proposed framework: $w_s = 3$, $w_{tr} = 10$.

Although this experiment uses a different platform which is not a humanoid robot, we can still use the same approach described in Sect. 8 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. We also collected a motion plan library for the mobile manipulator. However, there are abundant contacts in the test environment. Therefore, in this test, the proposed framework outputs a torso pose guiding path with only one segment, and uses PFS (which accounts for traversability) for that segment.

The result in Fig. 13 shows that the proposed framework has a much shorter planning time, but the resulting path takes more steps. Since the PFS Only planner does not consider traversability, the planner will explore the slightly-shorter path above the window first. However, the gap created by the pipe makes the path above the window require more steps than the path under the window and the PFS Only 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 PFS Only planner's, however we note that a difference of two steps in this context is not very large.

13 Discussion

On the other hand, the proposed planner uses the traversability estimates to identify that the gap caused by the pipe will reduce the number of contact transitions available in that region, and bias the PFS planner to take the path under the

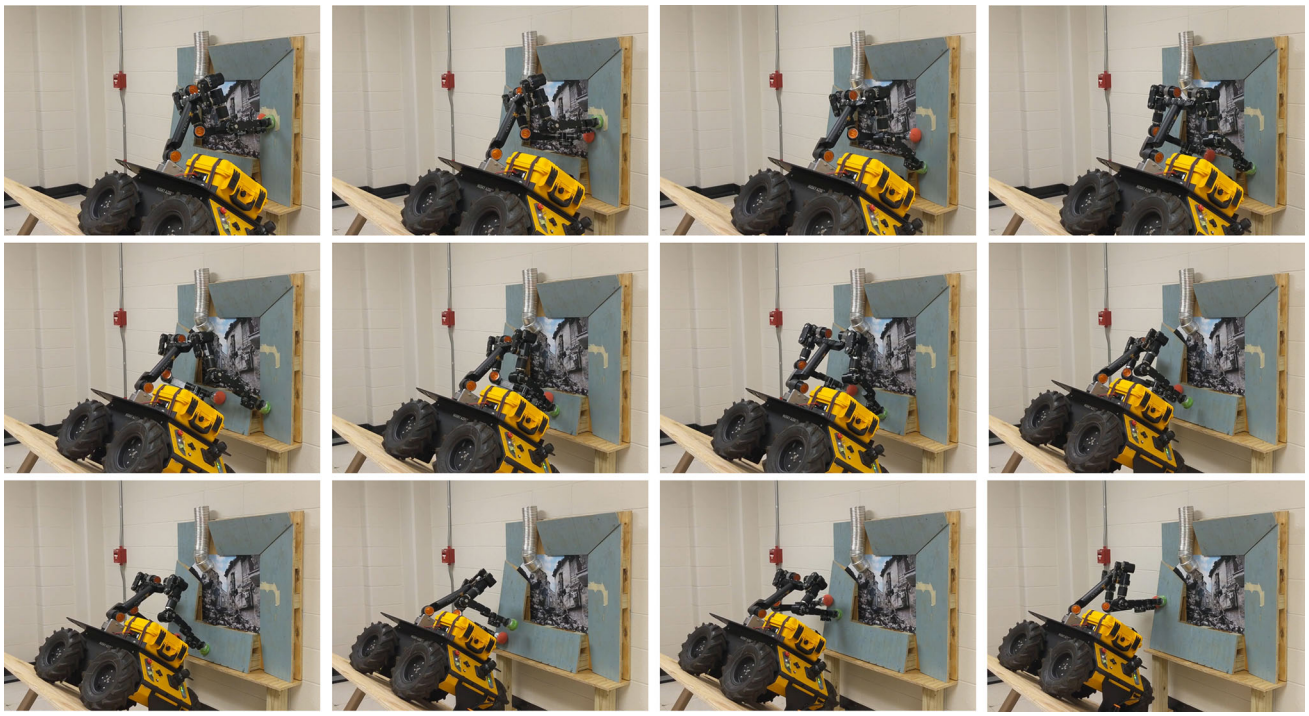


Fig. 14 Execution of the plan shown in Fig. 13 on the dual-arm mobile manipulator (Upper left: first frame, Lower right: last frame)

window. Therefore, the proposed planner is much faster than the standard planner. However, the proposed heuristic, which includes the traversability cost, is not admissible. Therefore, the proposed planner produces contact transition sequences with more steps.

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-space planning. With the real robot experiment, we also show that the planned contact transition sequence is executable by a real robot in Fig. 14 and the accompanying video. The video demonstrates that the planner can plan quasi-statically balanced motion for the mobile manipulator to slowly but steadily execute the path on the slope.

As demonstrated in the results, the main advantage of the proposed approach is to find contact transition sequences with a high success rate. Therefore, neither PFS nor RA in our approach optimizes the solution quality, such as the number of contacts. However, based on the proposed framework, we can further improve the solution quality if needed. PFS can utilize the anytime nature of ANA* to improve the solution over time. For RA, although the number of contacts is fixed for each motion plan during the process, the solution can be improved with post processing. One possible approach to reduce the number of contacts for RA is to iteratively sample contacts in the returned contact transition sequence and remove that contact if the contact transition sequence is still feasible without it.

In this work, only quasi-static balance is considered in order to reduce the computational difficulty of the problem. Compared to dynamic feasibility, quasi-static balance is faster to compute, but is more conservative, which makes it harder for the contact-space planner to find a solution. Recent work has addressed the computation time problem of evaluating dynamic feasibility in contact-space planning. Lin et al. (2019) use neural networks to predict motion dynamic feasibility for a given robot to speed up the process, but it suffers from accumulated error for longer planning horizons. Fernbach et al. (2018, 2020) efficiently evaluate the dynamic feasibility of contact transitions, but they are conservative in the selection of center of mass trajectories. Efficient evaluation of motion dynamic feasibility is still an active field of study, and we would like to test our approach with these methods in future work.

The torso pose guiding path provides informed guidance for the contact-space planner to find a feasible contact transition sequence efficiently. From Table 2, we see that the torso pose guiding path using traversability provides better guidance than not considering traversability. However, it is still possible that such guidance is misleading due to prediction error in the traversability estimates. To recover from an erroneous guiding path the region at which the contact-space planner is stuck can be marked as low-traversability in the

torso pose grid. The planner can then rerun the torso pose guiding path planner to restart the planning process.

The proposed approach assigns either PFS or RA for each segment of the torso pose guiding path based on the traversability of each segment. In the unlikely event that RA runs out of all N_{mp} motion plans in the library before time out or before finding a feasible plan ($N_{mp} = 50$ in the experiment), the planner will fall back to using PFS to try to find a path for the segment. While PFS may not find a path in the given time limit if the problem is difficult, this fallback can be effective if the segment was erroneously assigned to RA due to an error in the traversability estimator.

Despite our effort to improve the planner to find feasible contact sequence in difficult environments, there are cases when the proposed planner cannot find a solution. One direction to improve system reliability is to use a less conservative and higher fidelity model, such as a configuration space optimizer, specifically for regions where a solution cannot be found with simplified models. Another common approach to improve system reliability is to include a human operator to help the robot when the planner cannot find a solution. From industrial collaborative robots to autonomous driving, it is common for human operators to take over when robot autonomy fails in complicated situations. In our context, if the planner is unable to find a feasible solution to proceed, a human operator can suggest possible contact placements, which can then be verified by the robot. A human operator may also tele-operate the robot to clear out obstacles with its arms before calling the planner again.

14 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.

Considering traversability allows our framework to outperform an uninformed planner, and it allows us to segment a guiding path into parts where PFS or RA may be most appropriate. Our simulation experiments confirm that our framework outperforms simplifications that don't use all of its components in large environments. They also show that the framework is not very sensitive to cost function tuning parameters. The real robot experiment demonstrates the applicability of the proposed framework to a real robot in a disaster scenario.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10514-021-09996-3.

Funding Funding was provided by Office of Naval Research (US) (N00014-17-1-2050).

References

Baudouin, L., Perrin, N., Moulard, T., Lamiraux, F., Stasse, O., & Yoshida, E. (2011). Real-time replanning using 3d environment for humanoid robot. In *IEEE-RAS international conference on humanoid robots (humanoids)*.

Brandao, M., Fallon, M., & Havoutis, I. (2019). Multi-controller multi-objective locomotion planning for legged robots. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*

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

Chestnutt, J., Kuffner, J., Nishiwaki, K., & Kagami, S. (2003). Planning biped navigation strategies in complex environments. In *IEEE-RAS international conference on humanoid robots (Humanoids)*.

Chilian, A., & Hirschmuller, H. (2009). Stereo camera based navigation of mobile robots on rough terrain. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Chung, S., & Khatib, O. (2015). Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots. In *IEEE international conference on robotics and automation (ICRA)*.

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

Deits, R., Tedrake, R. (2014). Footstep planning on uneven terrain with mixed-integer convex optimization. In *IEEE-RAS international conference on humanoid robots (humanoids)*.

Diankov, R. (2010). Automated construction of robotic manipulation programs. PhD thesis, Carnegie Mellon University.

Dornbush, A., Vijayakumar, K., Bardapurkar, S., Islam, F., Ito, M., & Likhachev, M. (2018). A single-planner approach to multi-modal humanoid mobility. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 4334–4341).

Escande, A., Kheddar, A., Miossec, S., & Garsault, S. (2009). Planning support contact-points for acyclic motions and experiments on HRP-2. *Experimental Robotics, 2,* 293–302.

Fang, Z., Yang, S., Jain, S., Dubey, G., Roth, S., Maeta, S., et al. (2017). Robust autonomous flight in constrained and visually degraded shipboard environments. *Journal of Field Robotics, 34*(1), 25–52. https://doi.org/10.1002/rob.21670.

Fernbach, P., Tonneau, S., & Taïx, M. (2018) 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)*.

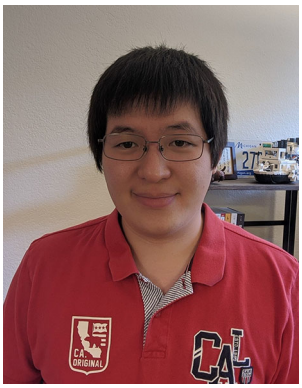
Fernbach, P., Tonneau, S., Stasse, O., Carpentier, J., & Taïx, M. (2020). C-CROC: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multicontact scenarios. *IEEE Transactions on Robotics, 36*(3), 676–691. https://doi.org/10.1109/TRO.2020.2964787.

Grey, M. X., Liu, C.K., & Ames, A. D. (2016). Traversing environments using possibility graphs with multiple action types. arXiv e-prints

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

Griffin, R. J., Wiedebach, G., McCrory, S., Bertrand, S., Lee, I., & Pratt, J. (2019). Footstep planning for autonomous walking over rough terrain. In: *2019 IEEE-RAS 19th international conference*

on humanoid robots (humanoids) (pp. 9–16). https://doi.org/10.1109/Humanoids43949.2019.9035046.
Hornung, A., Dornbush, A., Likhachev, M., & Bennewitz, M. (2012). Anytime search-based footstep planning with suboptimality bounds. In: *IEEE-RAS international conference on humanoid robots (humanoids)*.
Kanoun, O., Yoshida, E., & Laumond, J. P. (2009). An optimization formulation for footsteps planning. In *IEEE-RAS international conference on humanoid robots (humanoids)*.
Knabe, C., Seminatore, J., Webb, J., Hopkins, M., Furukawa, T., Leonessa, A., & Lattimer, B. (2015). Design of a series elastic humanoid for the DARPA robotics challenge. In *IEEE-RAS international conference on humanoid robots (humanoids)*.
Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., & Inoue, H. (2001). Footstep planning among obstacles for biped robots. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
Kumagai, I., Morisawa, M., Benallegue, M., & Kanehiro, F. (2019). Bipedal locomotion planning for a humanoid robot supported by arm contacts based on geometrical feasibility. In *IEEE-RAS international conference on humanoid robots (humanoids)*.
Lin, Y., & Berenson, D. (2016). Using previous experience for humanoid navigation planning. In *IEEE-RAS international conference on humanoid robots (humanoids)*.
Lin, Y., & Berenson, D. (2017). Humanoid navigation in uneven terrain using learned estimates of traversability. In: *IEEE-RAS international conference on humanoid robots (humanoids)*.
Lin, Y., & Berenson, D. (2018). Humanoid navigation planning in large unstructured environments using traversability-based segmentation. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
Lin, Y., Ponton, B., Righetti, L., & Berenson, D. (2019). Efficient humanoid contact planning using learned centroidal dynamics prediction. In: *International conference on robotics and automation (ICRA)*.
Maier, D., Lutz, C., & Bennewitz, M. (2013) Integrated perception, mapping, and footstep planning for humanoid navigation among 3d obstacles. In: *2013 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2658–2664). https://doi.org/10.1109/IROS.2013.6696731.
Michel, P., Chestnutt, J., Kuffner, J., & Kanade, T. (2005). Vision-guided humanoid footstep planning for dynamic environments. In *IEEE-RAS international conference on humanoid robots (humanoids)*.
Scherer, S., Rehder, J., Achar, S., Cover, H., Chambers, A., Nuske, S., & Singh, S. (2012). River mapping from a flying robot: State estimation, river detection, and obstacle mapping. *Autonomous Robots, 33*(1–2), 189–214. https://doi.org/10.1007/s10514-012-9293-0.
Shneier, M., Chang, T., Hong, T., Shackleford, W., Bostelman, R., & Albus, J. S. (2008). Learning traversability models for autonomous mobile vehicles. *Autonomous Robots, 24*(1), 69–86.
Suger, B., Steder, B., & Burgard, W. (2015). 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)*.
Tonneau, S., Prete, A. D., Pettré, J., Park, C., Manocha, D., & Mansard, N. (2018). An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics, 34*(3), 586–601.
van den Berg, J., Shah, R., Huang, A., & Goldberg, K. (2011). Anytime nonparametric A*. In *AAAI*.
Wellhausen, L., Dosovitskiy, A., Ranftl, R., Walas, K., Cadena, C., & Hutter, M. (2019). Where should I walk? Predicting terrain properties from images via self-supervised learning. *IEEE Robotics and Automation Letters, 4*(2), 1509–1516. https://doi.org/10.1109/LRA.2019.2895390.
Wermelinger, M., Fankhauser, P., Diethelm, R., Krüsi, P., Siegwart, R., & Hutter, M. (2016). Navigation planning for legged robots in challenging terrain. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.



Yu-Chi Lin received a B.S. and a M.S. in Electrical Engineering from National Taiwan University in 2012 and 2014, respectively, and received Ph.D. degree from Robotics Program, University of Michigan in 2020. He is currently a software engineer at Nuro, Inc. His research focuses on robot motion planning, humanoid robotics, and improving motion planning with data driven approaches.

Dmitry Berenson received a B.S. in Electrical Engineering from Cornell University in 2005 and received his Ph.D. degree from the Robotics Institute at Carnegie Mellon University in 2011. He completed a post-doc at UC Berkeley in 2011 and was an Assistant Professor in Robotics Engineering and Computer Science at WPI 2012-2016. He is currently an Assistant Professor in the EECS Department and Robotics Institute at the University of Michigan. He received the IEEE RAS Early Career Award and the NSF CAREER award. His current research focuses on motion planning and manipulation.