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

Yu-Chi Lin[1] and Dmitry Berenson[1]

***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. 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 guiding torso path which accounts for the difficulty of traversing the environment as predicted by learned regressors; and 2) Decomposes the guiding path into a set of segments, each of which is assigned a motion mode (i.e. a set of feet and hands to use) and a planning method. Easily-traversable segments are assigned 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 this segmentation approach greatly outperforms standard discrete planning and that using the library-based method for more difficult segments gives a benefit over using discrete planning.**

I. 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 Figure 1). Even though the robot's sensor range may be limited to only a few meters, it is still important to construct a long-term navigation plan to ensure the robot can reach its goal. Such a plan can be constructed from a pre-generated map of the environment; e.g., using a drone to map the environment before the humanoid enters.

In such environments, humanoid navigation can benefit greatly from the use of palm contacts. Palm contacts provide additional support to allow the robot to make larger steps to avoid obstacles, cross gaps, or help with balance. However, considering palm contact in discrete-search navigation planning algorithms [1]–[3] greatly increases the branching factor of the search, resulting in impractical planning times for large environments. The planning is also difficult because palm contacts may not be available in all locations and sometimes they may be unnecessary, so the robot needs to decide when and where to use its palms. In previous work we explored using library-based methods to address difficult navigation planning problems requiring palm contacts [4], but such methods are not efficient when navigating an easy-to-traverse part of the environment. To maximize efficiency, we would like to use discrete-search, which we call Planning from Scratch (PFS), to traverse easy areas and switch to the library-based method, which we call Retrieve and Adapt (RA), when traversal becomes difficult.

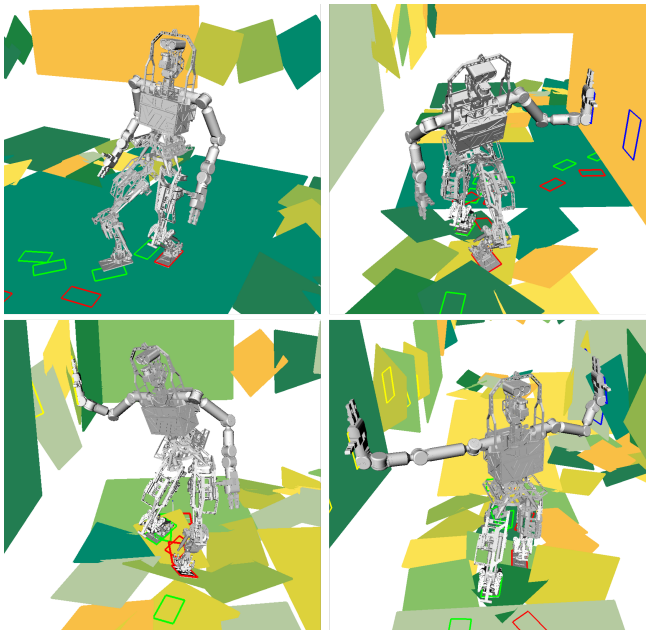

Fig. 1. Using different motion modes to traverse unstructured environments.

Thus, to plan a contact sequence in a large unstructured environment we present the framework shown in Figure 2. This framework relies heavily on the concept of humanoid *traversability*, which we introduced in previous work [5]. Traversability is defined as the time PFS will require to traverse a given area of the environment. Computing it is computationally expensive, so we have developed a way to learn a traversability estimator from data. In this paper, we extend this process to consider multiple predefined motion modes (i.e. different combinations of palms and feet).

The key novel contribution of our framework is the method to segment the guiding torso path to minimize planning time. We first segment the guiding path into motion modes based on traversability predictions for each mode. We then further

[1]Yu-Chi Lin and Dmitry Berenson are with the University of Michigan, Ann Arbor, MI, U.S.A. linyuchi@umich.edu, berenson@eecs.umich.edu. This work was supported in part by the Office of Naval Research grant N00014-17-1-2050.

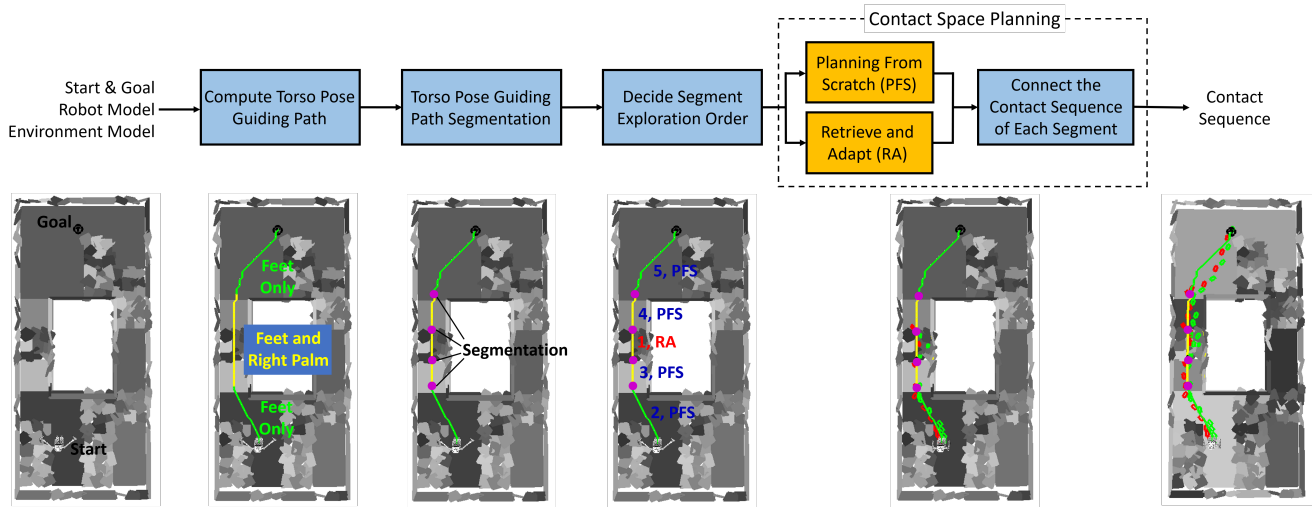

Fig. 2. The data flow of the proposed framework. Yellow denotes that the blocks operate on segments of the torso pose guiding path.

segment each segment based on the average traversability within the segment. This process results in segments that have either high or low average traversability. Based on the motion mode and the traversability of each segment we then assign a planning method to use: either PFS, when the segment is easy to traverse, or RA, when it is difficult. In addition to this contribution, we also improve on the computational overhead of our RA method and generalize it to consider motion plans of widely-varying length.

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

II. Related Work

Humanoid footstep planning has been studied extensively: [1]–[3], [6], [7] 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 [8], [9]. 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.

There is also work which focuses on humanoid navigation in unstructured environments using multiple contacts. [10] used optimization to find contacts in the neighborhood of a "rough" trajectory. However, the planning time was prohibitively long. [11] combined discrete-search-based contact space planning with a local trajectory optimizer to quickly compute a whole-body trajectory using multiple contacts. [12] 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. We share the idea of using a guiding path to reduce the search space. However, the above work defines which motion mode will be used before planning. 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 also been recent work addressing humanoid locomotion planning using different planners or action types. [13] 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. [14] 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.

III. Problem Statement

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

IV. Method Overview

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

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

V. 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 Figure 3. The ellipse shape captures the fact that the robot can travel farther with a forward or backward step than a lateral step.

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

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

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

$$P_{tp} = \{(m_1, p_{t,1}), (m_2, p_{t,2}), \ldots, (m_{N_p}, p_{t,N_p})\} \quad (2)$$

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

A. 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

$$\begin{aligned} &\Delta g_{tp}\left((m_i, p_{t,i}), (m_j, p_{t,j})\right) = \\ &l_{p_{t,j}}^{p_{t,i}} + w_s + w_{tr}\Delta g_{tr}\left(p_{t,i}, p_{t,j}, m_j\right) + M\left(m_i, m_j\right) \\ &M\left(m_i, m_j\right) = \begin{cases} 0, & m_i = m_j \\ w_m, & m_i \neq m_j \end{cases} \end{aligned} \quad (3)$$

where w_s is a fixed cost of taking a step, w_m is a fixed motion mode changing cost, $\Delta g_{tr} (0 \leq \Delta g_{tr} \leq 1)$ is the traversability cost associated with the transition from $p_{t,i}$ to $p_{t,j}$ using motion mode m_j (described in Section V-B), and w_{tr} is a weighting factor for Δg_{tr}. Possible actions are the combination of torso pose transitions shown in Figure 3 and the motion modes used. The heuristic function for planning the torso pose guiding path is

$$h_{tp}\left((m_i, p_{t,i})\right) = d_{goal}^{t}(p_{t,i}) + w_s \frac{d_{goal}^{t}(p_{t,i})}{d_{t,max}} \quad (4)$$

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

B. Traversability Estimates

Traversability describes how quickly the contact space planner can find a contact sequence to traverse through a region. If the planner knows which region has a higher traversability before planning, it can bias its search to avoid difficult regions, and generate a contact sequence more quickly. However, the true traversability will only be known after the contact space planner has found a path. Therefore, our previous work [5] quickly estimated traversability using a learning approach. We summarize this approach and our modifications below. For a full description of the method, please see [5].

For a given torso pose p_t in an environment $\mathbf{E}$, a traversability estimator is defined as $|\Gamma_+| : \{\mathbf{v}, m\} \rightarrow \mathbb{R}_+$, where $\mathbf{v}$ is a 2D torso pose translation in the XY plane, and $\mathbf{E}$ is expressed as the set of planar contact surfaces. We use a finite set of $\mathbf{v}$, as shown in Figure 3, and train an estimator for each $\{\mathbf{v}, m\}$ pair. Given a transition between two torso pose $p_{t,i}$ and $p_{t,j}$ using motion mode m, we can compute $\mathbf{v}$, and then use the estimator with the matching m and closest $\mathbf{v}$. To produce a traversability estimate, the estimator extracts contact clearance features in the neighborhood of torso pose p_t in the environment $\mathbf{E}$, and queries a learned regressor with these features. We describe this process below.

We adopt the Approximate Contact Checking (ACC) approach in [5] to extract features from the environment, and extend the traversability learning framework to include different motion modes. The ACC feature vector presented in [5] consists of 5 dimensions: The first dimension S_f represents the approximate clearance for the foot contacts to move the robot from a given torso pose p_t by a 2D torso pose transition $\mathbf{v}$, and the last four dimensions $[S_p[1], S_p[2], S_p[3], S_p[4]]$ corresponds to the approximate clearance for the palm contacts in the four quadrants of the torso pose frame. The left side of the robot is the 1st and 2nd quadrant of the torso pose frame, and the right side is the 3rd and 4th quadrant. Therefore, we define the ACC feature vector $\mathbf{S}\left(p_t, \mathbf{v}, m, \mathbf{E}\right)$ as:

$$\mathbf{S}\left(p_t, \mathbf{v}, m, \mathbf{E}\right) = \begin{cases} \left[S_f\right], & m = \text{feet only} \\ \left[S_f, S_p[1], S_p[2]\right], & m = \text{feet and left palm} \\ \left[S_f, S_p[3], S_p[4]\right], & m = \text{feet and right palm} \\ \left[S_f, S_p[1], S_p[2], S_p[3], S_p[4]\right], & m = \text{all end-effectors} \end{cases} \tag{5}$$

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

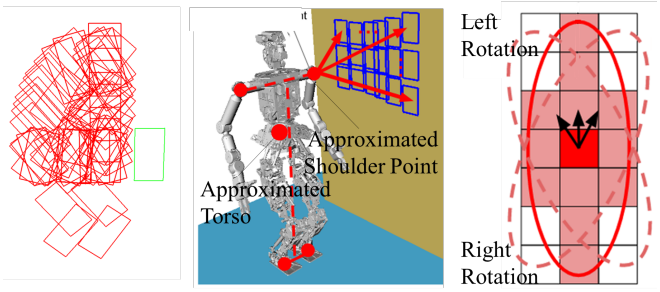

Fig. 3. Left: Foot contact transition model (57 steps) Middle: Palm contact transition model Right: Torso pose transition model in the torso pose grid.

VI. Torso Pose Guiding Path Segmentation

As mentioned in Section IV, 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. 2, we can extract the torso pose transition sequence $T_\delta\left(P_{tp}\right)$ defined as:

$$\begin{aligned} T_\delta\left(P_{tp}\right) &= \{\delta_1, \delta_2, \ldots, \delta_{N_\delta}\} \\ \delta_i &= \left(\mathbf{v}\left(p_{t,i}, p_{t,i+1}\right), \Delta\theta\left(p_{t,i}, p_{t,i+1}\right), m_{i+1}\right) \end{aligned} \tag{6}$$

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

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

$$\begin{aligned} \underset{\psi \in \Psi(\psi_{mm}[k])}{\text{argmax}} \quad & \sum_{i=1}^{|\psi|} \left| \sum_{\delta_j \in \psi[i]} \Delta g_{tr}\left(\mathbf{v}\left(\delta_j\right), m\left(\delta_j\right)\right) - |\psi[i]|\, T_{tr} \right| \\ \text{subject to} \quad & |\psi[i]| \geq N_{seg} \end{aligned} \tag{7}$$

where ψ is a segmentation of the kth torso pose transition sequence, $\psi[i]$ is the ith segment in that segmentation, and T_{tr} is a traversability cost threshold which serves as a way to decide which method (PFS or RA) to use to generate the contact sequence. This optimization will try to generate segments whose average Δg_{tr} is above or below T_{tr} as much as possible. We also add a constraint to exclude segments that are too short. Again, it is important to reduce the number of segments for the reasons described in Section V. To solve the optimization problem we could apply existing segmentation methods, however we found that the space of segmentations was relatively small and the objective function was very fast

to evaluate, thus instead we enumerate all segmentations, compute the cost of each, and choose the one that is optimal.

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

A. Decide Segment Exploration Order

After the segmentation is complete, each segment is planned for using either PFS or RA separately. To better connect motion plans in each segment, if a segment using RA directly follows a segment using PFS, we can generate the contact sequence of the latter segment first, and set the first stance in the latter segment as the goal for PFS in the previous segment. Similarly, if two neighboring segments both use PFS, we will always explore the previous one first, so that the latter segment can use the last stance of the previous segment as the initial state. By doing this, we automatically connect these two segments using PFS. The only exception is the connection between two segments both using RA. In this case, we will run another PFS starting from the last stance in the previous segment, and set the first stance in the latter segment as goal. Algorithm 1 shows the procedure used to decide the segment exploration order.

Algorithm 1: Decide Segment Exploration Order

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

VII. The Planning From Scratch (PFS) Approach

We now summarize the PFS approach to plan contact placements, which is taken from our our previous work [5]. To help clarify the framework used in this work, we outline its formulation here. Please refer to [5] for more details.

In PFS, we formulate the contact space planning problem as a graph search problem, and solve it with the ANA* algorithm [15]. Each state 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 shifted to new poses based on a predefined discrete transition model, as shown in Figure 3.

Since 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. 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 [16] 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.

We use a heuristic function to allow the planner to explore transitions in a goal-biased way. To compute the heuristic for each state, similar to the approach for planning the torso pose guiding path, we first plan on the torso pose grid. Our purpose here is to find the expected cost from each cell on the torso pose grid to the goal cell. Therefore, we adopt the edge cost definition in Eq. 3, but use Dijkstra's algorithm to find the cost of each cell to the goal cell. This algorithm outputs a torso policy (i.e. a direction to move for each cell) in the form of a tree. PFS queries the policy using the mean feet poses to get the corresponding cost of each cell g_{tp} for use in the contact space planner's heuristic function:

$$\begin{aligned} g_{tp}(p_t, m) &= \sum \Delta g_{tp}(p_i, p_{i,parent}, m) \\ &= l^t_{goal}(p_t) + w_s N_s + w_{tr} g_{tr}(m) \end{aligned} \tag{8}$$

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

A. Heuristic for Contact Space Planning

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

$$\begin{aligned} h_{p,lp}(p_t) &= l_{lp}(P_{p_t}) + w_s \frac{l_{lp}(P_{p_t})}{d_{lp,max}} \\ h_{p,rp}(p_t) &= l_{rp}(P_{p_t}) + w_s \frac{l_{rp}(P_{p_t})}{d_{rp,max}} \end{aligned} \tag{9}$$

where P_{p_t} is the path from the cell containing p_t to the goal in the torso policy, l_{lp} is the length of the portion of P_{p_t}

where it is possible to make left and palm contact with the environment, and likewise l_{rp} for right palm contact. $d_{lp,max}$ and $d_{rp,max}$ are the maximum distances each palm contact can travel in one action. For a given mode, we define the palm heuristic $h_p(p_t, m)$ 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: $h(p_t, m) = g_{tp}(p_t, m) + h_p(p_t, m)$.

VIII. The Retrieve and Adapt (RA) Approach

In [4], we showed that deforming an existing contact sequence to fit to the environment is an efficient approach to solve difficult contact space planning problems. We constructed a motion plan library, sorted the motion plans based on how well the contacts matched to the environment, and finally deformed motion plans one-by-one until a matching motion plan was found. In this work, we keep the motion plan contact sequence matching process presented in [4], but modify it to have less computational overhead in selecting a motion plan from the library. We also generalize the original approach to allow extraction of partial motion plans in order to fit a longer plan to a closer goal. In addition we allow connecting multiple plans to reach a distant goal by making multiple queries to the library for a single segment.

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

A. 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 4 shows some examples. Each motion plan π consists of a joint trajectory, the corresponding contact sequence $\mathcal{C}(\pi)$, and the motion plan torso path $P_t(\pi)$. $\mathcal{C}(\pi)$ is defined as

$$\mathcal{C}(\pi) = \{\langle \mathbf{c}_k, e_k \rangle \,|\mathbf{c_k} \in SE(3); k = 1, 2, ..., N_c\} \quad (10)$$

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

$$P_t(\pi) = \{p_k | p_k \in SE(2); k = 1, 2, ..., N_p\} \quad (11)$$

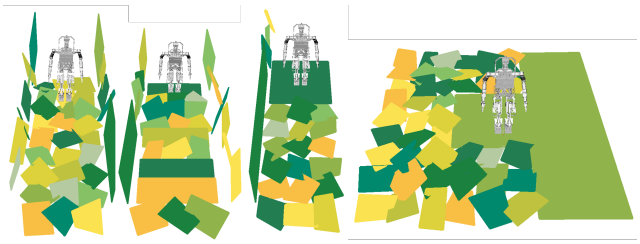

Fig. 4. Several example environments used to collect the motion plans to construct the motion plan library.

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

$$D(\pi) = \left\{ d_k | d_k = d(p_1, p_k), d_1 \leq, \ldots, \leq d_{N_p}, p_k \in P_t \right\} \quad (12)$$

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

B. Querying the Motion Plan Library

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

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

In both cases, if we cannot find a π_e to meet the distance requirement, we reject this motion plan. If π_e is found, we would like to deform its joint trajectory to move the contact

poses in $\mathcal{C}(\pi_e)$ to the surface patches in the environment so that the robot can make contact with the environment. To do this we apply the plan deformation process in [4], which aligns the plan to the environment using an iterative Jacobian to reduce the distance from the plan's contacts to the environment and then deforms the plan. We summarize the process below (see [4] for details). This process first treats the plan as a rigid object and is initialized in the following way: Given a query environment with the start $(x_s, y_s, z_s(x_s, y_s), \theta_s)$ and the goal $(x_g, y_g, z_g(x_g, y_g), \theta_g)$, the algorithm initializes the rigid plan pose $\mathbf{T}_{rp} = (x_{0,rp}, y_{0,rp}, z_{0,rp}, \theta_{0,rp})$ as:

$$\begin{aligned} &x_{0,rp} = x_s;\ y_{0,rp} = y_s;\ z_{0,rp} = z_s \\ &\theta_{0,rp} = atan2(y_g - y_s, x_g - x_s) \end{aligned} \tag{13}$$

After iteratively updating $\mathbf{T}_{rp}$ until convergence, we check if the plan's contacts are too far from their nearest surfaces, and if so we reject the plan. If not, the motion plan, now expressed as a sequence of configurations, is modified and optimized to fit the query environment with Contact-Consistent Elastic Strips (CES)[11]. Each configuration of the trajectory will moves contacts toward the nearest contact region. To speed up the process, we do not check the balance constraint in the loop of CES. Instead, we check if the resulting contact sequence follows the end-point balance constraints. If not, we reject the motion plan. Furthermore, to ensure connection between the motion plans generated in each segment of the torso pose guiding path, we force the first and last torso pose in the motion plan to be close in orientation to its corresponding pose in the torso pose guiding path segment, which means the final motion plan should obey these constraints after the deformation:

$$|\theta_s - \theta_{\pi_e,1}| \leq \theta_\Delta, |\theta_g - \theta_{\pi_e,N_e}| \leq \theta_\Delta \tag{14}$$

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

IX. Connecting the Contact Sequences

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

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

X. Experiments and Results

We evaluate the performance of the proposed framework in planning to navigate through two types of environments and we compare the proposed framework with two baselines: The first one is the standard contact space planning approach: PFS with all motion modes possible (PFS only). Since this planner is not required to use any palm contacts, it uses the feet-only motion mode heuristic to estimate the cost-to-go. The second baseline (Segmentation+PFS) uses our segmentation approach but only uses PFS to plan motion plan in each segment. Since it only uses PFS, we segment the torso pose guiding path only when motion mode changes. For the proposed framework, we also implemented two versions using different decision criteria to decide whether to plan with PFS or RA for a given segment: mean(Δg_{tr}) (Our Framework-Mean) and max(Δg_{tr}) (Our Framework-Max).

We implemented our algorithms in OpenRAVE[17], and tested on the Escher [18] robot model. All experiments were run on an Intel Core i7-4790K 4.40 GHz CPU with 16GB RAM. We use the following parameter values: $N_{mp} = 50, T_{tr} = 0.3, N_{seg} = 5, w_s = 3, w_m = 2, w_{tr} = 10, r_g = 0.2m, \theta_\Delta = 30^\circ$. The torso path grid is discretized to 0.15m resolution in x and y, and 30° in θ.

A. Two-Corridor Environment Test

In the two-corridor environment, we construct the environment as two wide rooms connected with two parallel corridors (see Figure 5). 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 $[-20^\circ, 20^\circ]$. The walls are also generated in the same manner. We set the start and the goal to be a random location in the lower and the upper room, respectively. We set a 500 second time limit. If the planner finds a contact sequence within the time limit in a trial, the trial is counted a success. We run on 50 testing environments, and compare the performance of the different approaches in terms of success rate and planning time for the successful trials (see Table I).

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

B. Two-Staircase Environment Test

A two-staircase environment is shown in Figure 5. Testing in this environment confirms that the framework can be

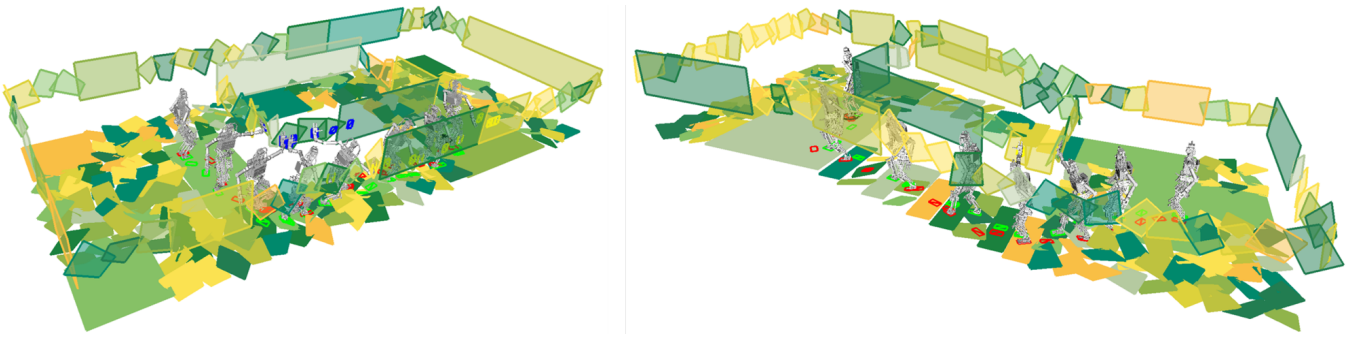

Fig. 5. Executing a planned contact sequence in Left: a two-corridor environment. Right: a two-staircase environment.

TABLE I

THE RESULT FOR TWO-CORRIDOR TEST ENVIRONMENT AND TWO-STAIR TEST ENVIRONMENT

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

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

XI. CONCLUSION

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

REFERENCES

[1] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Footstep planning among obstacles for biped robots. In *IROS*, 2001.

[2] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami. Planning biped navigation strategies in complex environments. In *Humanoids*, 2003.

[3] P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade. Vision-guided humanoid footstep planning for dynamic environments. In *Humanoids*, 2005.

[4] Y.C. Lin and D. Berenson. Using previous experience for humanoid navigation planning. In *Humanoids*, 2016.

[5] Y.C. Lin and D. Berenson. Humanoid navigation in uneven terrain using learned estimates of traversability. In *Humanoids*, 2017.

[6] L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida. Real-time replanning using 3d environment for humanoid robot," in humanoids. In *Humanoids*, 2011.

[7] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz. Anytime search-based footstep planning with suboptimality bounds. In *Humanoids*, 2012.

[8] O. Kanoun, E. Yoshida, and J. P. Laumond. An optimization formulation for footsteps planning. In *Humanoids*, 2009.

[9] R. Deits and R. Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *Humanoids*, 2014.

[10] A. Escande, A. Kheddar, S. Miossec, and S. Garsault. Planning support contact-points for acyclic motions and experiments on hrp-2. In *ISER*, 2008.

[11] S.Y. Chung and O. Khatib. Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots. In *ICRA*, 2015.

[12] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettre. A reachability-based planner for sequences of acyclic contacts in cluttered environments. In *ISRR*, 2015.

[13] M. X. Grey, A. D. Ames, and C. K. Liu. Footstep and motion planning in semi-unstructured environments using randomized possibility graphs. In *ICRA*, 2017.

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

[15] J. van den Berg, R. Shah, A. Huang, and K.Y. Goldberg. Anytime nonparametric A*. In *AAAI*, 2011.

[16] S. Caron, Q. Pham, and Y. Nakamura. Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics. In *RSS*, 2015.

[17] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, 2010.

[18] Coleman Knabe, John Seminatore, Jacob Webb, Michael Hopkins, Tomonari Furukawa, Alexander Leonessa, and Brian Lattimer. Design of a series elastic humanoid for the darpa robotics challenge. In *Humanoids*, 2015.