A Single-Planner Approach to Multi-Modal Humanoid Mobility

Andrew Dornbush[1] Karthik Vijayakumar[1] Sameer Bardapurkar[1] Fahad Islam[1] Masayuki Ito[2] Maxim Likhachev[1]

***Abstract*— In this work, we present an approach to planning for humanoid mobility. Humanoid mobility is a challenging problem, as the configuration space for a humanoid robot is intractably large, especially if the robot is capable of performing many types of locomotion. For example, a humanoid robot may be able to perform such tasks as bipedal walking, crawling, and climbing. Our approach is to plan for all these tasks within a single search process. This allows the search to reason about all the capabilities of the robot at any point, and to derive the complete solution such that the plan is guaranteed to be feasible. A key observation is that we often can roughly decompose a mobility task into a sequence of smaller tasks, and focus planning efforts to reason over much smaller search spaces. To this end, we leverage the results of a recently developed framework for planning with adaptive dimensionality, and incorporate the capabilities of available controllers directly into the planning process. The resulting planner can also be run in an interleaved fashion alongside execution so that time spent idle is much reduced.**

I. Introduction

Recent years have shown much interest in developing robust humanoid robots that can operate in environments that are often unstructured, cluttered, and unpredictable compared to controlled industrial settings. Furthermore, the structure that does exist is intended primarily for people, and not designed with robots in mind. Structures such as staircases, ladders, railings, complement people's mobility. This leads us to design humanoids so that they possess capabilities similar to people such as the ability to walk, climb, and use surfaces such as handrails for support.

The need for all these capabilities provides a number of challenging problems for motion planning. The most pronounced problem is the inherent high dimensionality of the robot's configuration space. To guarantee that a plan safely and efficiently accomplishes a given task may require reasoning about all of the joints of the robot and the relationship between the robot and the various objects in its environment. These constraints are expensive to evaluate.

Luckily, a complicated mobility task can often be broken down into a sequence of smaller tasks. For example, a task for a robot to move from one end of a facility to the other might include traversing large areas by walking, climbing staircases or ladders and, in situations where the environments is hazardous, crawling under fallen structures or over debris. Examples of such environments are shown in Fig. 1.

[1]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA
[2]Mitsubishi Heavy Industries, Ltd., Nagoya, Japan
This work was sponsored by Mitsubishi Heavy Industries, Ltd.

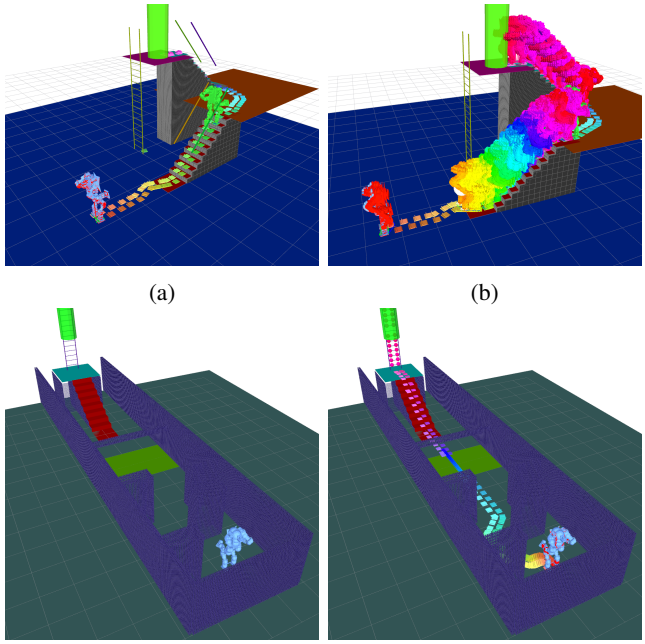


Fig. 1: Example environments and plans - hazardous "crawl-only" zones are depicted as green regions in the bottom two figures

Typically, current approaches solve this problem hierarchically: a top-level planner decomposes the complete task into smaller tasks and then runs a different planner, specialized for the each task, in isolation. Once plans are computed for each task, the top-level planner figures out how the robot will transition from one task to the next. This can be done with yet another specialized planner, or prescripted behaviors. The results of the DARPA Robotics Challenge, as demonstrated in [1], [2], [3], show the ubiquity of specialized task planners and behaviors. This common approach has shown to be brittle, as each task planner is constrained to satisfy the requirements of the original task decomposition, and must satisfy strict endpoint constraints to ensure that the transitions between tasks are feasible. In the worst case, where the top-level planner has chosen an incorrect decomposition, one of the planners may be unable to generate a solution at all or a transition between tasks is infeasible.

The approach presented here builds upon the notion of adaptive dimensionality. Rather than always search through a high-dimensional state space, adaptive dimensionality automatically figures out what dimensions are relevant in each

region of the state space. This is tremendously beneficial to planning for humanoid mobility as there are various modes of locomotion, each requiring planning in its own dimensionality. This paper presents how adaptive dimensionality can be applied to humanoid mobility, describes an implementation that yields real-time execution by interleaving planning and execution, and presents experimental results showing the practicality of the approach.

II. Related Work

Much work in humanoid mobility planning is focused on solving specific sub-problems of mobility. Examples of navigation planning using footsteps are shown in [4], [5]. These techniques plan in a low-dimensional space representing feasible footstep actions. They may rely on a controller to produce feasible joint trajectories or the planner may generate these trajectories online to ensure footstep validity. Example whole-body planning techniques have been explored in [6], [7], [8]. These approaches are generally intended for object interaction tasks, and don't consider incorporating locomotion. Some example techniques specifically for climbing ladders are presented in [9] and [10].

Relatively less work has been done for humanoid robots on adaptively reasoning about the relevant dimensionality of the problem during the search process. Some examples of adaptive reasoning include [11] and [12]. Both works decompose the robot into appropriate subsystems based on kinematics. The first RRT-based approach adaptively adds subsystems as the search gets closer to the goal. The second, optimization-based approach, plans iteratively, incorporating more descendant subsystems until a valid path is found. These approaches both iteratively increase the dimensionality of the *entire* search space, whereas our approach only increases dimensionality of the search space in regions where high-dimensional planning is required.

III. Planning Framework

Our application to multi-modal humanoid navigation targets the humanoid robot, whose design is discussed in [13]. The robot has 4 symmetric limbs, each with 7 degrees of freedom, and additional joints for reorienting the attached sensors. Each limb has a dual-purpose end effector, designed with a flat surface for walking and support, and a hook for latching onto cylindrical objects in the environment, such as ladder rungs and handrails. In this work, we consider all of the joint variables of the limbs. The pose of the robot provides an additional 6 degrees of freedom. Together these degrees of freedom define a 34-dimensional search space.

The robot is equipped with specialized controllers for bipedal walking, crawling, and climbing ladders.[1,2]. Additionally, we are able to explicitly control each of the joints to execute raw full-body paths. While it is possible to plan paths consisting of only full-body motion, we leverage the existing controllers, both to improve planning efficiency and to generate plans that can be executed more robustly on the actual robot.

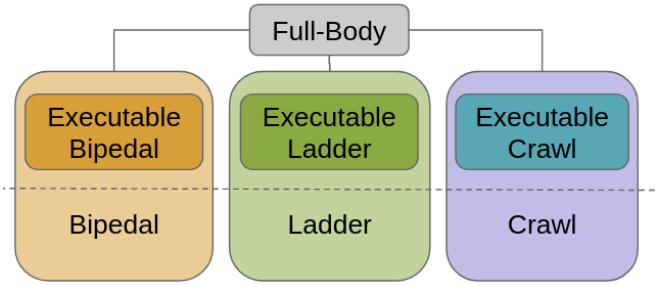


Fig. 2: Relationship between representations

We represent the planning problem as a search over a finite, discrete search space. The search space consists of a discrete state space S, and a set of transitions $T = \{(s_i, s_j) | s_i, s_j \in S\}$. Each pair $(s_i, s_j) \in T$ represents a feasible transition between two states. Each transition is associated with a scalar cost, $c(s_i, s_j) > 0$. We use the notation $\pi(s_i, s_j)$ to denote a path from state s_i to state s_j, and $\pi^*(s_i, s_j)$ to denote an optimal, least-cost path. This search space defines a graph G, with vertex set S and edge set T. The goal of the planner is to find a path in G from a given start state s_s to a goal state $s_g \in S_G$, where $S_G \subset S$.

To improve the efficiency of the search through this high-dimensional space, and to integrate the available controllers, our algorithm takes much of its inspiration from the Planning with Adaptive Dimensionality framework presented in [14] and [15]. The framework for planning with adaptive dimensionality makes the observation that, in many areas of the search space, it is often not necessary to reason about the high dimensionality of the state space, as many of the resulting paths have a low-dimensional structure. Rather than always planning in the 34-dimensional configuration space of the robot, our method searches as much as possible in the low-dimensional spaces corresponding to each controller. These spaces consider only the poses of the robot's feet for bipedal mobility, the pose of the robot's center-of-mass for crawling, or the poses of the end effectors for ladder climbing. Fig. 2 shows the relationship between the full-body representation and the low-dimensional representations.

Section IV will begin with a brief overview of the framework for planning with adaptive dimensionality. Sections V and VI will describe extensions to planning with adaptive dimensionality to enable planning with multiple low-dimensional planning representations simultaneously. Section VII will discuss the details of the search algorithm and how to incorporate information from multiple heuristics into a multi-representation search. Section VIII will describe how the search process is interleaved with execution of the plan. Section IX will present a brief overview of the control architecture on the robotic platform, and how plans are delivered to the appropriate controllers during execution.

[1]The robot and controllers were designed and developed by Waseda University and Mitsubishi Heavy Industries, Ltd.

[2]The simulation models of the robot and the experimental test field were provided by ImPACT TRC Program of Council for Science, Technology and Innovation (Cabinet Office, Government of Japan)

Section X will list the results of sample runs of the planner on targeted test environments.

IV. Planning with Adaptive Dimensionality

This section provides a brief overview of the framework for planning with adaptive dimensionality. For detailed analysis of the adaptive dimensionality framework and additional applications, see [14].

For a complete planning solution, search often needs to reason over a high-dimensional state space. However, large portions of a complete plan often exhibit a lower-dimensional structure. For example, part of a humanoid mobility task might include large segments of bipedal walking. In these scenarios, it suffices to plan only for the footstep locations, and reserve planning in the high-dimensional space for verifying that each footstep is feasible. The portions of the plan requiring high-dimensional reasoning are infrequent compared to portions that can be solved in this manner.

The planning with adaptive dimensionality framework leverages this low-dimensional structure by iteratively constructing a hybrid search space, composed primarily of low-dimensional states and transitions, and introducing high-dimensional states and transitions where necessary to ensure feasibility of the resulting path.

A. Graph Structure

The adaptive dimensionality framework considers two state spaces: the original high degree-of-freedom state space that represents valid configurations of the robot, and a projection of the original state space to a low-dimensional representation, respectively labeled S^{hd} and S^{ld}. A many-to-one mapping defined by

$$\lambda : S^{hd} \rightarrow S^{ld}$$

represents the projection from the high-dimensional space to the low-dimensional space. The inverse, one-to-many, mapping defined by

$$\lambda^{-1}(s_{ld}) = \{s \in S^{hd} | \lambda(s) = s_{ld}\}$$

represents the projection from a state in the low-dimensional space back to a subset of states in the high-dimensional space.

Both the high-dimensional and low-dimensional space can have its own set of transitions, T^{hd} and T^{ld} respectively. However, to guarantee completeness and bounded suboptimality, the following constraint is required:

$$c(\pi^*(s_i, s_j)) \geq c(\pi^*(\lambda(s_i), \lambda(s_j))), \forall s_i, s_j \in S^{hd}$$

That is, the cost of the optimal path between any two states in the high-dimensional space must be at least the cost of the optimal path between their projections in the low-dimensional space.

The notation G^{hd} and G^{ld} represent the corresponding high-dimensional and low-dimensional graphs defined as (S^{hd}, T^{hd}) and (S^{ld}, T^{ld}), respectively.

B. Search Algorithm

Rather than search for a path in the original high-dimensional search space, G^{hd}, the adaptive dimensionality search algorithm prefers to search as much as possible in the low-dimensional search space, G^{ld}. To accomplish this, the search iteratively constructs a new hybrid search space G^{ad}, composed of an adaptive state space S^{ad}, and transition set T^{ad}. This new search space is composed primarily of states and transitions from G^{ld} and is expanded to include regions of states and transitions from G^{hd} as necessary.

Initially, the adaptive search space G^{ad} includes all of G^{ld}. When a region of high-dimensional states is introduced, G^{ad} is updated so that low-dimensional states s that fall within the high-dimensional region are replaced by their high-dimensional equivalents in $\lambda^{-1}(s)$.

To be able to search this hybrid space, we must define a transition set that includes transitions between states from S^{ld} and S^{hd}. The state transition function is defined in Alg. 1. This transition set includes transitions between low- and high-dimensional states, and only includes transitions to states in the adaptive state space S^{ad}. Notice that expanding or adding a new high-dimensional region produces a new instance of G^{ad}.

Algorithm 1 Adaptive Dimensionality Transition Function

```
1:  procedure GetSuccessors(s)
2:      succs = ∅
3:      if s ∈ S^hd then
4:          for (s, s') ∈ T^hd do
5:              if s' ∈ S^ad then
6:                  Add s' to succs
7:              else
8:                  Add λ(s') to succs
9:      else                                              ▷ s ∈ S^ld
10:         for (s, s') ∈ T^ld do
11:             if s' ∈ S^ad then
12:                 Add s' to succs
13:         for s_hd ∈ λ^-1(s) do
14:             for (s_hd, s'_hd) ∈ T^hd do
15:                 if s'_hd ∈ S^ad then
16:                     Add s'_hd to succs
17:     return succs
```

The adaptive search algorithm begins by finding a path, π_{ad}, from the start to the goal in the current instance of G^{ad} (set to G^{ld} initially). This path is allowed to contain states of differing dimensionalities, and so may not be executable. If no path is found during this phase, then no path exists from the start to the goal and the search terminates. To construct an executable path from π_{ad}, another search is conducted within a tunnel surrounding π_{ad}. The tunnel τ of radius w is defined as a subgraph of G^{hd}. A high-dimensional state s is said to be inside τ if there exists a state $s_{ad} \in \pi_{ad}$ such that the distance from $\lambda(s)$ to s_{ad} (or $\lambda(s_{ad})$ if $s_{ad} \in S^{hd}$) is no larger than w for some pre-defined distance metric on S^{ld}. All transitions $(s, s') \in T^{hd}$ are included such that $s, s' \in \tau$.

If the search fails to find a path from the start to the goal within τ, one or more high-dimensional regions are introduced where the search became stuck, and the adaptive search begins a new iteration on a new instance of G^{ad}. See [14] for more details on how to identify locations to place

high-dimensional regions.

V. Adaptive Dimensionality with Multiple Low-Dimensional Representations

In the domain of humanoid mobility planning, several useful low-dimensional representations are available. In our case for example, each of the available controllers receives waypoints from a low-dimensional representation, such as footsteps, poses for the robot, or poses for its end effectors. To be able to plan solutions that incorporate all of these representations, we need the ability to combine them into a single search space.

Our approach maintains the separation between the high-dimensional and low-dimensional search spaces and their ability to define their own transition sets. Given n low-dimensional representations, we define low-dimensional discrete state spaces $S^1, S^2, \ldots, S^n$, and their corresponding transition sets, $T^1, T^2, \ldots, T^n$.

The mappings from the high-dimensional space to each low-dimensional space remain largely unchanged as well. For the i'th low-dimensional representation, a mapping defined by

$$\lambda_i : S^{hd} \rightarrow S^i$$

represents the mapping from states in S^{hd} to states in S^i. Correspondingly, the inverse functions

$$\lambda_i^{-1}(s_i) = \{s \in S^{hd} | \lambda(s) = s_i\}, \forall i \in 1..n$$

represent the mapping from states $s_i \in S^i$ to subsets of S^{hd}.

Additionally, we define functions

$$\lambda_{i,j}(s_i) = \{s \in S_j | \exists s_{hd} \in \lambda_i^{-1}(s_i) \text{ and } \lambda_j(s_{hd}) = s\}$$

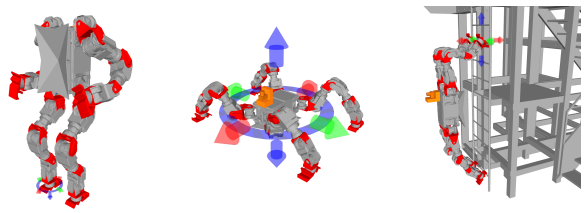
to represent the mappings between states of low-dimensional representations. These mappings may be one-to-one, one-to-many, or other variations depending on the dimensionality of the target representation.

The construction of S^{ad}, and its graph representation, G^{ad}, follows from its construction in the adaptive dimensionality framework. The initial instance of G^{ad} is the union of the search spaces $G^i = (S^i, T^i)$ for each of the low-dimensional representations. The transition set, T^{ad} is extended to include projections from the high-dimensional representation to each low-dimensional representation. Additionally, T^{ad} also contains transitions that allow the search to effectively switch between low-dimensional representations. The complete transition set is defined in Alg. 2.

Thus far, we have only described how to incorporate multiple low-dimensional representations into the adaptive dimensionality framework. This indeed speeds up the search for finding a high-dimensional path, but we also desire to explicitly reason about the controller capabilities, to avoid high-dimensional planning wherever possible. Recall that during the second phase of each search iteration, the algorithm searches for a completely high-dimensional path, within the tunnel τ. To relieve the search of needing to perform high-dimensional planning, we extend the transition set of the high-dimensional representation to include all of the transitions that correspond to actions from the low-dimensional representations that are directly executable by an available controller.

Algorithm 2 Multi-Representation Adaptive Dimensionality Transition Function

```
1:  procedure GetSuccessors(s)
2:      succs = ∅
3:      if s ∈ S^hd then
4:          for (s, s') ∈ T^hd do
5:              if s' ∈ S^ad then
6:                  Add s' to succs
7:              else
8:                  Add λ_i(s') to succs ∀ i ≤ n
9:      else                                              ▷ s ∈ S^i
10:         for (s, s') ∈ T^i do
11:             if s' ∈ S^ad then
12:                 Add s' to succs
13:         for S^j ∈ {S^k | k ∈ 1..n, k ≠ i} ∪ S^hd do
14:             for s_j ∈ λ_{i,j}(s) do
15:                 for (s_j, s'_j) ∈ T^j do
16:                     if s'_j ∈ S^ad then
17:                         Add s'_j to succs
18:     return succs
```



(a) (b) (c)

Fig. 3: Humanoid developed by Waseda University/Mitsubishi Heavy Industries, Ltd. and its low-dimensional representations

VI. Low-Dimensional Representations for Humanoid Mobility

In this domain, the high-dimensional state space represents all the controllable degrees of freedom of the robot. To incorporate the capabilities of the available controllers, we chose low-dimensional representations that match the input spaces of each controller. For crawling, the controller requires a 4-dimensional pose, (x, y, z, θ), of the robot. For bipedal walking, the controller requires paths that specify the 4-dimensional pose, (x, y, z, θ), of each foot. Finally, for ladder climbing, the controller requires only the 6-dimensional pose, $(x, y, z, \alpha, \beta, \gamma)$, for each of the four end-effectors. These low-dimensional representations are depicted in Fig. 3.

A state vector for the full-body state space contains a single discrete variable for each actuated joint, plus 6 variables for the pose of the robot. Combined, a state vector for the full-body state space is represented as

$$(pose_x, pose_y, pose_z, pose_\phi, pose_\theta, pose_\psi, j_1, j_2, \ldots, j_{28})$$

Each discrete variable corresponds to a range of continuous values, obtained by simple discretization functions. The discretization resolution was chosen as $1\,\mathrm{cm}$ for all translational variables, and as 5° for all rotational variables. The action space is built from several types of motion primitives. The

first type directly moves each joint of the robot individually by some small delta. We chose simple motion primitives that varied each joint by the state space discretization of $\pm 5^\circ$. The second type of motion primitive uses an inverse kinematics solver to move the position of one of the end effectors by a small positional delta. We chose simple primitives to move an individual end effector by $\pm 2.5\,\mathrm{cm}$ in x, y, or z. Finally, we allow full-body IK motions for the root of the robot. The root is allowed to move in x, y, or z by $\pm 5\,\mathrm{cm}$ or yaw by 12.25°, 22.5°, or 45°. Since there were no controllable degrees of freedom between the root of the robot and the base of each limb, our full-body IK solver simply runs an isolated IK solver for each of the supporting limbs. The last type of motion primitive is an adaptive motion primitive that computes the motion, on-the-fly, that achieves a selected target for one of the limbs. These targets are selected according to nearby support surfaces, such as the ground or handrails.

The representation for bipedal walking contains state vectors describing the 4-dimensional (x, y, z, θ) poses of each of the foot, plus one extra variable for restricting the gait of the robot to a left-right alternating scheme. The combined state vector is represented as

$$(x_l, y_l, z_l, \theta_l, x_r, y_r, z_r, \theta_r, pivot)$$

where $pivot$ is $left$ or $right$ to denote the pivot foot. We restrict the actions allowed in the bipedal state representation to a fixed set of target poses, offset from the current pivot foot. Application of one of these actions places the opposite, active foot with respect to the pivot foot. We included 15 total primitives: 2 of these allow reorienting of the feet to produce turning motions, and the remaining 13 move the feet forward at varying distances from $2\,\mathrm{cm}$ to $24\,\mathrm{cm}$ to trade off between state space coverage and allowing the search to quickly explore long distances via forward walking.

The representation for crawling contains state vectors describing a four-dimensional (x, y, z, θ) pose for the center-of-mass of the robot. We include a simple set of actions that allow the robot to move directly forward or backward by $10cm$, and to turn-in-place by 45°, to mimic the capabilities of the controller.

The representation for ladder climbing contains state vectors describing the three-dimensional (x, y, z) positions for each of the robot's end effectors. Each action moves all four end effectors from their current positions to positions on the ladder rung directly above or below the currently held rung, as indicated by the current end effector positions. The actions were selected to be consistent with the actual capabilities of the ladder climbing controller.

The projection functions, $\lambda_1, \ldots, \lambda_n$ from the full-body representation to each of these low-dimensional representations require solving forward kinematics for the given full-body state. The inverse projections from each low-dimensional representation to the high-dimensional representation require expensive inverse kinematics queries to determine valid configurations for the robot. To accelerate this process, we precompute a small set of nominal joint configurations for the full-body state, and perform small searches for a nearby valid configuration. Conveniently, these nominal joint configurations also correspond to the joint configurations we can expect the robot to achieve after execution of one of the controllers.

By design, the actions in the crawling and ladder representations are always directly executable by an existing controller. However, the bipedal representation contains actions that are not directly executable. For example, the bipedal representation is allowed to make footsteps that traverse up and down staircases, while the available controller is only allowed to operate on even terrain. These non-executable actions are resolved during the second phase of the search by planning in the high-dimensional space and generating full-body motion to move along these footsteps.

VII. MR-MHA*

In this section we describe a planning algorithm which is better suited to planning in multiple representation state-spaces. This algorithm, called the MultiRep-MultiHeuristic A* (MR-MHA*), is a generalization of the MHA* algorithm [16] that can reason over several different state-space representations, each of which may have its own heuristics defined.

Multi-Heuristic A* is a search framework that uses multiple inadmissible heuristics to simultaneously explore the search space, while preserving guarantees of completeness and suboptimality bounds by using a single admissible "anchor" heuristic. The algorithm has shown success in complex high-dimensional planning problems such as mobile manipulation planning for the 12D PR2 robot, where a naive weighted-A* approach is sensitive to large local minima. Two variants of MHA* are described in [16]. In this work we simply refer to the shared variant SMHA* as MHA*.

As mentioned earlier, humanoid mobility presents several low-dimensional representations like bipedal walking, ladder climbing, etc. These representations are substantially complex and fundamentally different enough that we require different heuristics to explore them effectively. Furthermore, we can leverage multiple heuristics to allow the search to explore different regions simultaneously, such as exploring whether to climb a ladder or a staircase to reach the goal. Note that MR-MHA* simplifies to vanilla MHA* algorithm when planning for a single representation. In our case, the second (full-body) phase of the search is one such example.

A. Algorithmic Details

1) Heuristic Lists: Following MHA*, we have a single admissible heuristic defined over all representations to guarantee suboptimality bounds on the solution obtained from the search. However, each one of the n additional heuristics available to the search may be defined for only one of the available representations. This assignment of heuristics to representations allows the search to explore simultaneously across representations. The $InitializeHeuristicLists()$ procedure in lines 1-5 of Alg. 3 defines the mapping from each representation to the heuristics that are defined for it.

Algorithm 3 MR-MHA*

```
1: procedure INITIALIZEHEURISTICLIST
2:     for d = 1 to max_dim do
3:         for i = 1 to n do
4:             if h_i is defined for dim d then
5:                 Add i to heuristic_list[d]
6: procedure KEY(s, i)
7:     return g(s) + w_1 × h_i(s)
8: procedure EXPAND(s)
9:     Remove s from all OPEN lists
10:    for s' ∈ SUCC(s, dim(s)) do
11:        if s' was never visited then
12:            g(s') = ∞; bp(s') = null
13:        if g(s') > g(s) + c(s, s') then
14:            g(s) = g(s') + c(s, s'); bp(s') = s
15:            if s' has not been expanded in the anchor search then
16:                insert/update s' in OPEN_0 with key(s', 0)
17:                if s' has not been expanded in any inadmissible search then
18:                    for i ∈ heuristic_list[dim(s')] do
19:                        if key(s', i) ≤ w_2 × key(s', 0) then
20:                            insert/update s' in OPEN_i with key(s', i)
21: procedure MR-MHA*
22:    g(s_goal) = ∞; bp(s_start) = bp(s_goal) = null
23:    g(s_start) = 0
24:    INITIALIZEHEURISTICLIST()
25:    for i = 0 to n do
26:        OPEN_i = ∅
27:        if i = 0 or i ∈ heuristic_list[dim(s_start)] then
28:            insert s_start into OPEN_i with key(s_start, i) as priority
29:    while OPEN_0 not empty do
30:        for i = 1 to n do
31:            if OPEN_i.MinKey() ≤ w_2 × OPEN_0.MinKey() then
32:                if g(s_goal) ≤ OPEN_i.MinKey() then
33:                    terminate and return path pointed to by bp(s_goal)
34:                s = OPEN_i.Top()
35:                expand(s)
36:            else
37:                if g(s_goal) ≤ OPEN_0.MinKey() then
38:                    terminate and return path pointed to by bp(s_goal)
39:                s = OPEN_0.Top()
40:                expand(s)
```

It is important to note that the searches used in the low-dimensional and high-dimensional phases of the adaptive dimensionality search are independent. Thus, we are allowed to use different anchor and additional heuristics when planning for only the full-body representation vs. planning for the low-dimensional representations.

2) Successor Generation: In MHA*, when a state is expanded, its successors are inserted into all additional heuristic queues that are available to the search, provided it has not been expanded from either the anchor or any of the additional searches. This enables MHA* to effectively share paths between different heuristics that can help the search in different parts of the state-space.

However, when a state is expanded in MR-MHA*, successors are generated only from that state's representation and are only inserted into heuristic queues for heuristics which are defined for the successor state's representation. These distinctions are shown in lines 10 and 18 in the algorithm. This allows MR-MHA* to effectively share paths between representations without unnecessarily expanding states from irrelevant representations. This way, each search in MR-MHA* searches its own representation. For example, one search may expand states that correspond to bipedal states while another search may expand states that correspond to crawling states. These searches proceed simultaneously and share partial solutions found on the way to generate an overall path that involves both bipedal and crawling motions to get to the goal.

B. Implementation Details

Here we summarize the heuristics used for each state-space representation in our framework.

A common approach to designing a heuristic function for a given state space is to first project it to a low-dimensional representation, and use the result of a search in the low-dimensional space as the heuristic value for a corresponding high-dimensional state. We designed several 3D grid searches with cost functions tuned to produce meaningful heuristic values, and computed those values online using a Dijkstra search from the goal to the state whose heuristic we are computing. Each grid search becomes an additional heuristic used by the search.

Bipedal Representation

- Sum of grid search distances from both feet to the goal
- Sum of grid search distances from both feet to the goal, with penalties for stepping close to the edges of staircase steps, to encourage alignment with the staircase direction
- Sum of grid search distances from both feet to the goal, with low cost for using the ladder, to encourage ladder usage

Ladder Representation

- Zero heuristics

Crawl Representation

- Grid search distance from the COM of the robot to the goal

Full-Body Representation (2nd phase of the search)

- Grid search distance from the COM of the robot to the goal
- Remaining number of steps on the low-dimensional path that need to be tracked

In addition to the 3D grid search heuristics, which provide global estimates of the path to the goal, we designed several other heuristics to guide the full-body search to accomplish local tasks such as climbing staircases and mounting ladders. These heuristics are combined additively with the 3D grid search heuristic for the full-body.

The following heuristics were designed for any state generated by the 2nd phase of the search while planning in staircase climbing regions:

- Difference in heading between the root of the robot and the feet of the robot (Fig. 4)
- Euclidean distance between COM of the current state and a desired COM above the pivoting foot
- Euclidean distance between stepping foot of the current state and the target footprint (Fig. 4)
- Curve to provide guidance for stepping feet movement during the search (Fig. 4)

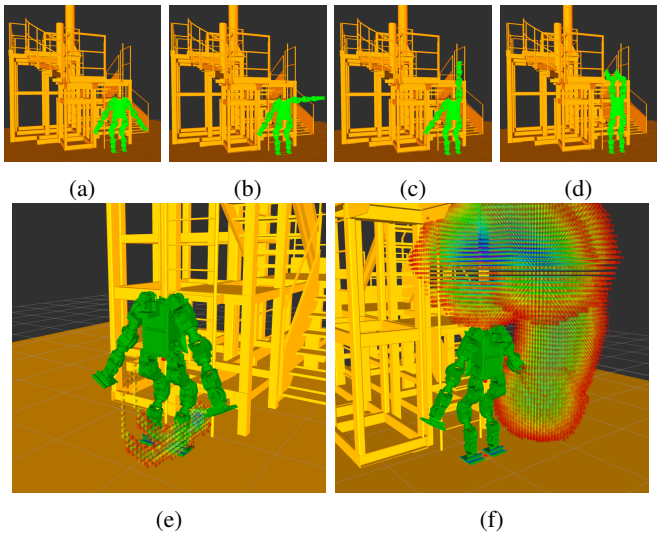


Fig. 5: Fig. 5e and Fig. 5f illustrate the heuristic values for the demonstration in Fig. 5a through Fig. 5d. Cells in blue have lower heuristic values.

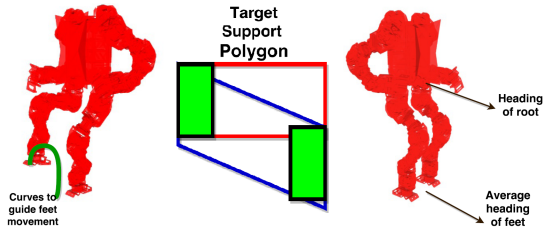


Fig. 4: Full body representation heuristics.

For the full-body planning phase, while searching for a motion that corresponds to mounting a ladder, we applied the Experience Graph (E-Graph) heuristic, discussed in [17]. The E-Graph heuristic incorporates prior experience, in our case from user demonstrations, to guide the search towards reusing paths that are likely to lead to the goal. We produced a demonstration, shown in Fig. 5, of the humanoid mounting a ladder to incorporate into the E-Graph heuristic. During the search, an E-Graph heuristic for each end effector is computed on demand, incorporating the user demonstration and fresh obstacle data. The final heuristic for a full-body state is the sum of the contributions from each of the individual end effector heuristics.

VIII. Interleaved Planning and Execution

Owing to the complexity of the planning tasks that we are addressing, typical planning times to plan a path all the way from the start to the goal are significantly high. Instead of waiting for the planner to generate the entire executable path, we interleave path planning and execution. The planner returns partial plans as it runs while the controller executes these plans on the robot in parallel. This idea has been widely studied by the real-time family of heuristic search algorithms. For the humanoid domain this approach significantly reduces the overall planning and execution time because the path execution is generally slow.

		success %		mean time (s)	
scenario	goal	plan	track	plan	track
test field	top	89.6	57.5	58.79	42.53
test field	mid	89.6	65.7	57.67	33.61
gauntlet	ladder	95.6	100.0	54.91	52.31

TABLE I

Within the underlying framework of planning with adaptive dimensionality, we only employ the interleaving scheme in the tracking phase, because the tracking phase returns a path that is executable by the robot.

Algorithm 4 Interleaved Planning and Execution

1: **Inputs:**
    lookahead
2: **while** tracking time $\neq$ lookahead **do**
3: Run Tracking
4: Reconstruct partial path
5: Send partial path to the controller for execution
6: Reset start state with the tail of the partial path
7: Loop

IX. Controller Framework

The planner outputs paths that contains transitions which correspond to executing an available controller, e.g. walking, crawling, climbing, or executing full-body joint motion. Each controller is responsible for converting a sequence of input waypoints to a full-body joint trajectory, possibly by using a local planner, gait controller, prescripted behavior, or some other method, that can be robustly executed on the robot. For example, the bipedal walking controller must compute and execute the full-body motion to achieve footsteps while maintaining stability.

As described in Section V, the paths output by the planner are hybrid paths that may contain both full-body joint motion as well as low-dimensional, executable transitions. To execute these hybrid paths, we developed a meta-controller, which sequentially dispatches segments of the hybrid path to the appropriate controllers until the entire plan has been successfully executed. Additionally, since the planner interleaves planning with execution, the meta-controller is responsible for updating the plan as new segments are received.

X. Experimental Analysis

To demonstrate the effectiveness of the multi-heuristic adaptive planning approach, we tested its ability to plan paths in the sample environments shown in Fig. 1.

The 'test field' environment shown in Fig. 1a consists of two platforms, a ladder connecting the ground level to the top platform, and two staircases connecting the ground to the middle platform and the middle to the top platform. In this environment, we tested the planner's ability to find high-dimensional paths that start on the ground level and reach the middle or the top platform. We tested the planner from 231 start poses, evenly distributed across (x, y) positions in the environment and from different starting headings, to goals positions on the mid- and top-level platforms.

The 'gauntlet' environment shown in Fig. 1c represents a hallway with large obstacles to walk around, a large overhanging bar that must be crawled under (represented by the rectangular green zone), a staircase, and a final ladder. We tested the planner's ability to surpass all these obstacles by planning from 23 different start poses on one end of the hallway to a goal position on the ladder at the other end. The resulting plan had to include transitions that switch between all of our planning representations.

In all cases, the planner was given $80\,\mathrm{s}$ to find a low-dimensional path, (generated by the 1st phase) and $180\,\mathrm{s}$ to find a high-dimensional path (generated by the 2nd phase) that tracks the low-dimensional path. Table I lists the success rates and planning times across all scenarios. The success rates are listed independently for both the low-dimensional planning phase (1st phase) and the high-dimensional tracking phase (2nd phase). However, the results for the tracking phase are with respect to the scenarios in which the planning phase was able to find a path, thus the overall success rates for the planner are 51.52% and 58.87%, respectively for the test field goals, and 95.6% for the gauntlet goal.

Note that, for experimentation, interleaved planning and execution is disabled. In reality, the robot is only idle for the duration of the planning phase, plus a small lookahead for the high-dimensional search.

The high success rate for the gauntlet scenario demonstrates the ability to robustly plan across several representations. Because the multi-representation, multi-heuristic search uses independent heuristics for each representation, and thus independent search queues, it is able to explore different representations simultaneously with little overhead.

The test field scenario exhibits high success rate in the planning phase, but a lower success rate for the tracking phase. One primary cause of this is that the low-dimensional planning phase may generate paths that are optimistic and not easily tracked in the high-dimensional planning phase. For example, a footstep action to climb the staircase sideways might represent a difficult or impossible to achieve sequence of actions in the full-body space. While the adaptive search handles this problem in successive iterations by introducing high-dimensional regions, in practice it might be beneficial to plan a low-dimensional path during the planning phase that has a high likelihood of corresponding to an executable high-dimensional path. It is future work to consider this option.

XI. Conclusion

In this work, we have presented an approach to planning for multi-modal humanoid mobility using a single search algorithm. This approach is able to simultaneously reason about all the capabilities of the robot, incorporate the capabilities of available controllers, and automatically discover the transitions for switching between modes of locomotion. The resulting planner brings together planning with an adaptively-dimensional search space, using multiple heuristics, and incorporating user demonstrations to concentrate search efforts where they're most needed. In future work we hope to incorporate planning for more complex interaction with the environment and address the robustness of scenarios requiring high-dimensional planning.

References

[1] S. J. Yi, S. McGill, L. Vadakedathu, Q. He, I. Ha, J. Han, H. Song, M. Rouleau, D. Hong, and D. D. Lee, "Thor-op humanoid robot for darpa robotics challenge trials 2013," in *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Nov 2014, pp. 359–363.

[2] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padir, F. Polido, G. G. Tighe, and X. Xinjilefu, "No falls, no resets: Reliable humanoid behavior in the darpa robotics challenge," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 623–630.

[3] Y. Zhang, J. Luo, K. Hauser, H. A. Park, M. Paldhe, C. S. G. Lee, R. Ellenberg, B. Killen, P. Oh, J. H. Oh, J. Lee, and I. Kim, "Motion planning and control of ladder climbing on drc-hubo for darpa robotics challenge," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2086–2086.

[4] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," *Robotics Research*, pp. 365–374, 2005.

[5] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*. IEEE, 2012, pp. 674–679.

[6] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 1656–1662.

[7] A. Athar, A. M. Zafar, R. Asif, A. A. Khan, F. Islam, O. Hasan, *et al.*, "Whole-body motion planning for humanoid robots with heuristic search," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4720–4727.

[8] S. Dalibard, A. Nakhaei, F. Lamiraux, and J. P. Laumond, "Whole-body task planning for a humanoid robot: a way to integrate collision avoidance," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, Dec 2009, pp. 355–360.

[9] Y. Zhang, J. Luo, K. Hauser, R. Ellenberg, P. Oh, H. A. Park, M. Paldhe, and C. S. G. Lee, "Motion planning of ladder climbing for humanoid robots," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, April 2013, pp. 1–6.

[10] M. Kanazawa, S. Nozawa, Y. Kakiuchi, Y. Kanemoto, M. Kuroda, K. Okada, M. Inaba, and T. Yoshiike, "Robust vertical ladder climbing and transitioning between ladder and catwalk for humanoid robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 2202–2209.

[11] N. Vahrenkamp, C. Scheurer, T. Asfour, J. Kuffner, and R. Dillmann, "Adaptive motion planning for humanoid robots," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 2127–2132.

[12] C. Park, J. Pan, and D. Manocha, "High-dof robots in dynamic environments using incremental trajectory optimization," *International Journal of Humanoid Robotics*, vol. 11, no. 02, p. 1441001, 2014.

[13] K. Hashimoto, S. Kimura, N. Sakai, S. Hamamoto, A. Koizumi, X. Sun, T. Matsuzawa, T. Teramachi, Y. Yoshida, A. Imai, K. Kumagai, T. Matsubara, K. Yamaguchi, G. Ma, and A. Takanishi, "Warec-1–a four-limbed robot having high locomotion ability with versatility in locomotion styles," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, Oct 2017, pp. 172–178.

[14] K. Gochev, B. Cohen, J. Butzke, A. Safonova, and M. Likhachev, "Path planning with adaptive dimensionality," in *Fourth annual symposium on combinatorial search*, 2011.

[15] K. Gochev, A. Safonova, and M. Likhachev, "Planning with adaptive dimensionality for mobile manipulation," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2944–2951.

[16] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a*," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[17] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-graphs: Bootstrapping planning with experience graphs." in *Robotics: Science and Systems*, vol. 5, no. 1, 2012.