A Multi-Heuristic framework for Humanoid Planning

Karthik Vijayakumar

CMU-RI-TR-17-57

August 2017

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Maxim Likhachev (Chair)
Manuela M. Veloso
Venkatraman Narayanan

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Abstract

Humanoid robots have been the subject of active research for several years, with the aim of developing systems that can potentially replace humans in performing dangerous tasks with similar agility and versatility. Motion planning for humanoid robots is a particularly challenging problem because of the high-dimensionality of the planning space, kinematic constraints and stability. The general approach to planning for humanoid mobility in complex environments, such as industries, with potentially multiple state-space abstractions, for e.g. bipedal, ladder, crawling etc., is to plan separately for each of these representations and combine the solution. A recently developed technique of planning with adaptive dimensionality can be used to develop a single planner to handle such multiple state-space abstractions. To this end, we develop a Multi-heuristic framework, as a generalization of MHA*, that can simultaneously plan across multiple state-space representations to produce a single solution. We test our planner in challenging environments containing several abstractions such as staircases, ladders, etc.

A heuristic based planner for high-dimensional state-space planning has a potential drawback of the user having to define good heuristic functions that guide the search. This can become a very tedious task for a system as complex as the humanoid. In this thesis, we address the issue of automatically deriving heuristic functions by learning macro-actions from a database of previous plans. By extracting spatio-temporal bases of repeatedly seen motions in prior plans, we generate new macro-actions as a planning pre-computation, subject to various constraints. We also show how these macro-actions can be used as heuristics and provide preliminary results on full-body planning for humanoids.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humanoid robots has been the subject of active research over the past several years [1],[3],[2]. The goal of humanoid research is to develop a robot with agility and precision similar to a human. This enables the usage of these robots in environments that require solving of complex tasks suited to a human, but are also dangerous for humans to operate in. Industrial environments, disaster prone settings, nuclear power plants, search and rescue, maintenance etc. are ideal examples where humanoids can replace humans for better functioning. Developing systems as complex and as versatile as humans comes with a lot of challenges. Motion planning for humanoids is especially a challenging problem, firstly because of the high dimensional state-space that a humanoid plans in, and secondly because of the various constraints that come with planning for human-like systems.

Figure 1.1: Humanoids in different application

1.1 Related Work

Several techniques have been proposed literature to tackle the motion planning problem for humanoids. [17] presents an overview of earlier motion planning techniques used independently to solve the problem of footstep planning, manipulation, full body motion etc using sampling based approaches. There have also been extensions to make motion plans which are dynamically feasible [18]. There has also been significant effort in using search-based techniques for optimal path planning for humanoid footsteps[9] [14] [13]. There has also been recent work on whole body navigation for humanoids with significant focus on combining task constrained planning and full body locomotion [8].

1.2 Preface

Though a lot of literature has focused on motion planning for humanoids to perform stair climbing, crawling on the ground, climbing ladders as individual problems, there has not been any work to develop a single framework for planning for humanoids to the best of our knowledge. In collaboration with others [1], the goal of this research project is to develop a single planner that would be able to plan among several different modes such as planning for stairs, ladders, crawling on clutter etc. This kind of framework allows the planner to simultaneously search all representations to get the best path to goal, instead of having to hierarchically breaking down the planning problem and run different planners.

To this end, in this thesis we develop a Multi-Heuristic search-based planner that can work with multiple abstractions in the environment to produce a valid plan Chapter. 2. Since heuristic based planners for high DOF systems like the humanoid are hugely heuristic driven and are often to prone to local-minima, we develop an approach to learn useful humanoid macro-actions that can be used to guide the search Chapter. 3. We provide some concluding arguments and directions for future work in Chapter. 4.

[1] Andrew Dornbush, Sameer Bardapurkar, Kalin Gochev, Fahad Islam, Masayuki

Chapter 2

Planning in Multiple Representations

2.1 Introduction

In this chapter we present a Multi-Heuristic framework to planning for a variety of motions of the humanoid. In a typical industry setting, we can expect a humanoid to be able to perform a wide range of tasks such as bipedal motion on the ground, handling stairs of different dimensions, climb ladders, crawl on cluttered objects etc. As stated earlier, the number of DOFs of a humanoid robot is very high and planning in such a state-space is hugely time consuming. While planning in a high dimensional state-space is often necessary to ensure the feasibility of the resulting path, large portions of the path have a lower-dimensional structure [11].

We take advantage of this observation to create a multi-representation state-space that sits on top of a recently developed framework called Adaptive dimensionality [11] to plan for various motions of the Humanoid. Each representation would typically correspond to a lower dimensional state-space of a unique motion that a Humanoid would execute.

To effectively make use of this Adaptive multi-representative state-space, we introduce a generalization of the Multi-heuristic A* planner to plan in such a multi-representative state-space.

2.2 Notations

In this thesis we use the exact same notation as in [11]. We represent the planning environment as a discretized finite state-space S, of dimensionality d consisting of state vectors $X = (x_1, x_2, ...x_d)$ and a set of transitions $T = \{(X_i, X_j) \,|X_i, X_j \epsilon S\}$. Each transition corresponds to a feasible transition between the corresponding state vector values and is associated with a cost $c\,(X_i, X_j)$ which is bounded from below by some positive , that is, $c\,(X_i, X_j) > \delta > 0$. Thus, we have an edge-weighted graph G with a vertex set S and edge set T. The goal of the planner is to find a least-cost path in G from the start state X_S to the goal state X_G. We will use the notation $\pi\,(X_i, X_j)$ to

denote a path in graph G from state X_i to state X_j . We will use $\pi^* (X_i, X_j)$ to denote a least-cost path. The cost of any path $\pi (X_i, X_j)$ is the cumulative costs of the transitions along it and will be notated by $c (\pi (X_i, X_j))$.

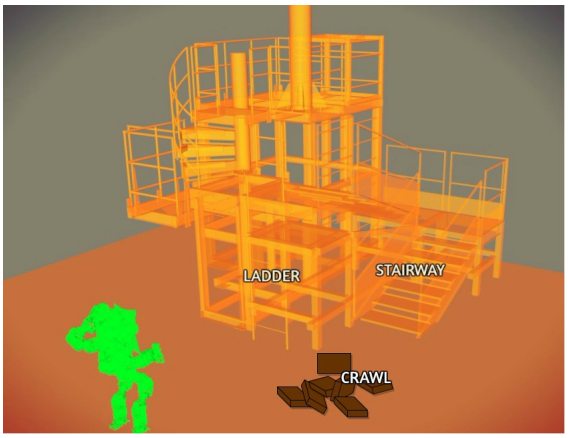


Figure 2.1: An example environment of operation for the Humanoid Robot

2.3 Adaptive Dimensionality

Here we present an overview of the Adaptive Dimensionality framework with a two state-space illustration for easier understanding [11].

Consider two state-spaces, a higher dimensional state-space S^{hd} and a lower dimensional state-space S^{ld}, where S^{hd} can be projected into S^{ld}.

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

where λ represents a many-to-one mapping. There also exists an inverse projection λ^{-1} which maps from the lower dimension to a higher dimension. Typically, this is one-to-many mapping and projects every lower dimensional state to a set of higher dimensional state.

Each state-space have its own set of transitions T^{ld} and T^{hd} and their corresponding graphs G^{ld} and G^{hd}.

The algorithm iteratively constructs and search a hybrid graph G^{ad} consisting of both low- and high-dimensional states and transitions. Initially G^{ad} is identical to G^{ld}. The iterative nature of the algorithm stems from the fact that each iteration identifies new areas of G^{ad} where high-dimensional regions need to be introduced until a valid solution is found. Upon addition of new high-dimensional regions into G^{ad}, another search iteration is performed on the new instance of G^{ad} taking into account the new high dimensional regions. The process is repeated until a search

iteration is able to successfully compute a solution that is feasible in the high-dimensional state-space and satisfies the specified cost sub-optimality bound.

To satisfy bounds on sub-optimality the algorithm requires that the cost of a least-cost path between any two states in the high dimensional state-space to be at least the cost of a least-cost path between their images in the low-dimensional state-space. if π represents the path then we have

$$c(\pi(X_i, X_j)) \geq c(\pi(\lambda(X_i), (X_j)))$$

The pseudo-code for planning with adaptive dimensionality as in [11] is shown in Fig. 2.2 below

$G^{ad} = G^{ld}$
AddFullDimRegion($G^{ad}, \lambda(X_S)$)
AddFullDimRegion($G^{ad}, \lambda(X_G)$)
loop
  search G^{ad} for least-cost path $\pi^*_{ad}(X_S, X_G)$
  if $\pi^*_{ad}(X_S, X_G)$ is not found **then**
    return no path from X_S to X_G exists
  construct a tunnel τ around $\pi^*_{ad}(X_S, X_G)$
  search τ for least-cost path $\pi^*_{\tau}(X_S, X_G)$
  if $\pi^*_{\tau}(X_S, X_G)$ is not found **then**
    let $\pi(X_S, X_{end})$ be the returned path
    if X_{end} is already within FullDimRegion in G^{ad} **then**
      GrowFullDimRegion($G^{ad}, \lambda(X_{end})$)
    else
      AddFullDimRegion($G^{ad}, \lambda(X_{end})$)
  else if $c(\pi^*_{\tau}(X_S, X_G)) > \epsilon_{\text{track}} \cdot c(\pi^*_{ad}(X_S, X_G))$ **then**
    identify a state X_r where a new FullDimRegion needs to be introduced
    if X_r is already within FullDimRegion in G^{ad} **then**
      GrowFullDimRegion(G^{ad}, X_r)
    else
      AddFullDimRegion(G^{ad}, X_r)
  else
    return $\pi^*_{\tau}(X_S, X_G)$

Figure 2.2: Pseudo code of Planning with Adaptive dimensionality

2.4 Multi-Representation Adaptive Dimensionality

In this section, we summarize a direct extension of the framework of Adaptive Dimensionality to multiple representations as described in [10]. As mentioned earlier, a humanoid robot in an industrial setting may be needed to navigate around challenging features of the environment such as ladder, stairs etc. An example figure in shown in Fig. 2.1.

A straight-forward extension of the Adaptive dimensionality framework is to consider several lower dimensional representations for a complex system such as a humanoid, and track solutions

of these lower dimensional plans in the full DOF of the robot. Hence, one can think of a lower dimensional representation each for bipedal motion, crawling motion, ladder climbing etc. A lower dimensional plan would consist of several different lower dimensional transitions and projections between these representations. We then plan in the full-body only in regions of the state-space where the lower dimensional plan does not satisfy controller constraints. Some available lower dimensional representations for the Waseda humanoid robot is mentioned in Fig. 2.3.

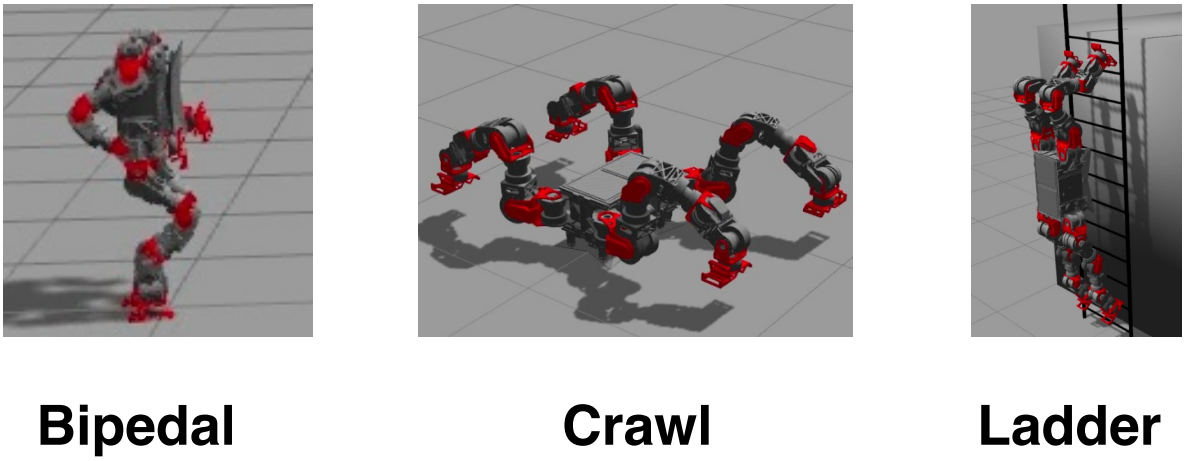


Figure 2.3: Lower dimensional representations

When developing complex robotic systems, such as legged robots, researchers often develop higher-level controllers that allow the system to execute simple tasks or behaviors based on simple inputs. For instance, controllers that maintain balance while minimizing the distance to a desired robot configuration, or even controllers that achieve basic locomotion based on a desired direction of movement and speed [29]. These controllers are usually carefully tuned operate with precision on the specific robotic system. The motion planning framework can leverage such built-in system capabilities in order to improve its performance. Rather than having to always produce full-body trajectories, the planner can produce the simplified inputs required by a given high-level controller to achieve a desired action or task, provided that the high-level controller is safe to use in the particular part of the state-space. For parts of the state-space that do not allow for the safe utilization of high-level controllers, the planner can resort to full-body planning.

Consider, for example, the problem of navigating a bipedal robot through a complex environment. Let us assume that the robot has the built-in capability to robustly follow a sequence of footstep locations which conform to a set of pre-defined constraints Q (e.g. consecutive footsteps are not too far from each other, the change in heading between consecutive footsteps does not exceed a threshold, the sequence maintains a minimum safe distance from obstacles, etc.). Thus, if the planner is made aware of this built-in capability, by specifying the state-space on which the high-level controller operates (expected input to the controller), a set of transitions available in this state-space, the capability constraints Q, and the parts of the environment that the capability is available,

then the planner can make use of this simplified state-space and perform footstep planning for large areas of the environment, thus limiting the use of full-body planning to challenging areas of the environment. The fact that such high-level controllers are available for many robotics systems can be exploited in higher dimensional tracking phase instead of full dimensional planning in the tracking phase of Adaptive Dimensionality.

Hence, a high level controller will be able to generate a high dimensional path π^{hd} from a lower dimensional path π^{ld} provided the lower dimensional path satisfies the controller constraints. For every high-level controller, we can construct a state-abstraction $A = (\lambda, \lambda^{-1}, G = (S, T), c)$ that operates in the sub-space S, by providing a set of transitions T , which satisfy controller constraints Q_t, and a cost function $c : T \rightarrow R^+$. The projection function λ and λ^{-1} are implicitly defined by the choice of S.

In the framework of adaptive dimensionality it was assumed that the lower dimensional path will not be directly executable by the robot in the higher dimension, and hence the feasibility is checked through an explicit tracking phase. However, path segments through state-abstractions constructed from high-level controllers can be executed by the robot provided that these path segments satisfy the corresponding high-level controller requirements. Thus, we can simplify and expedite the search performed during the tracking phase of the algorithm by not requiring full-dimensional tracking. This can be extremely beneficial for very high dimensional planning problems, such as motion planning for humanoid robots, as even the tunnel-constrained full-dimensional search during the tracking phase can be prohibitively expensive. A more detailed presentation of the approach of the same can be found in [10].

2.5 MultiRep-MHA*

In this section we discuss a planning algorithm which would be better suited to planning in adaptive multiple representation state-spaces. This planner called the MultiRep-MultiHeuristic A*(MR-MHA*) is a generalization of the MHA* algorithm [4] that can take into account several different state-space representations simultaneously in the planning space to successfully determine a plan.

Multi-Heuristic A* is a search framework that uses multiple inadmissible heuristics to simultaneously explore the search space, while preserving guarantees of completeness and sub-optimality bounds using a consistent heuristic. The algorithm showed success in higher dimensional complex planning problems such as Mobile manipulation planning for the 12D PR2 robot where a naive wA* approach is prone to huge depression regions. [4] describes two variants of the MHA* approach, IMHA* where individual searches run independently and SMHA*, where the searches share the current path obtained to a state. A improved variant of MHA* has also been presented in [20]. In this thesis unless otherwise stated, MHA* refers to the Shared variant of the algorithm(SMHA*).

MHA* having proved its efficacy for higher dimensional planning problems is a natural choice for planning in complex systems such as humanoids. When the planning space is composed sev-

eral different state-space representations, it might be effective if the planner can explore different representations simultaneously to reach the goal.

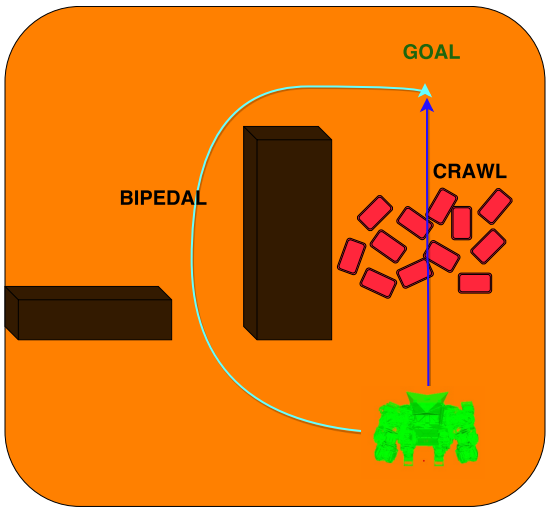


Figure 2.4: Motivating example for MR-MHA*

Consider for example, an industrial environment consisting of a staircase, a ladder and a humanoid robot has to reach from the ground to a top platform which can be reached either through the stairway or the ladder. In such a scenario, it would be preferable for the search to explore along both the ladder and stairway at the same time and whichever representation gets it faster to the goal is the shorter path. Also, since we need the statespace for bipedal and ladder representations are significantly different, we might need different heuristics to explore along each representation. This is the motivation for generalizing the MHA* to a multi-representation setting. The algorithm is presented as follows in Fig.2.5.

2.5.1 Algorithmic Details

Heuristic Lists

Following MHA*, we have a single consistent heuristic across all lower dimensional representations to satisfy sub-optimality bounds on the solution obtained from the lower dimensional planning space. Each lower dimensional representation now holds its own set of inadmissible heuristics which are independent of each other. The higher dimensional representation has its own anchor and a set of inadmissible heuristics. These are defined in lines $1 - 4$ in the algorithm.

It is important to note the lower dimensional planning phase and higher dimensional tracking phase are independent searches and hence we have a single anchor across lower dimensional representations which is different from the higher dimensional anchor heuristic. Consider a planning space

```
1  procedure InitializeHeuristicList()
2  for i = 1 to max_dim
3     heuristic_list[i].inadm = h_1^i, h_2^i, ...
4  procedure key(s, i)
5  return g(s) + w_1 * h_i(s);
6  procedure Expand(s)
7  Remove s from OPEN_i ∀ i in heuristic_list[s → dim];
8  for each s' in Succ(s)
9     if s' was never visited
10       g(s') = ∞; bp(s') = null;
11    if g(s') > g(s) + c(s, s')
12       g(s') = g(s) + c(s, s'); bp(s') = s;
13       if s' has not been expanded in the anchor search
14          insert/update (s') in OPEN_0 with key(s', 0);
15          if s' has not been expanded in any inadmissible search
16             for i in heuristic_list[s' → dim].inadm
17                if key(s', i) ≤ w_2 * key(s', 0)
18                   insert/update (s') in OPEN_i with key(s', i);
19 procedure MR-MHA*()
20 g(s_goal) = ∞; bp(s_start) = bp(s_goal) = null;
21 g(s_start) = 0;
22 InitializeHeuristicList();
23 for i = 0 to n
24    OPEN_i = ∅;
25    if i in heuristic_list[s_start → dim]
26       insert s_start into OPEN_i with key(s_start, i) as priority;
27 while OPEN_0 not empty
28    for i = 1 to n
29       if OPEN_i.Minkey() ≤ w_2 * OPEN_0.Minkey()
30          if g(s_goal) ≤ OPEN_i.Minkey()
31             terminate and return path pointed by bp(s_goal);
32          s = OPEN_i.Top();
33          Expand(s);
34       else
35          if g(s_goal) ≤ OPEN_0.Minkey()
36             terminate and return path pointed by bp(s_goal);
37          s = OPEN_0.Top();
38          Expand(s);
```

Figure 2.5: Multi-Rep Multi-Heuristic A* (MR-MHA*)

with n lower dimensional representations specified as $S^{ld1}, S^{ld2}, ...S^{ldn}$ and the higher dimensional representation specified as S^{hd}. Now for each of these representations we can define heuristics as

$$S^{ld1} : [h_0^{ld}, h_1^{ld1}, ...h_p^{ld1}]$$

$$S^{ld2} : [h_0^{ld}, h_1^{ld2}, ...h_q^{ld2}]$$

.

.

$$S^{ldn} : [h_0^{ld}, h_1^{ldn}, ...h_r^{ldn}]$$

$$S^{hd} : [h_0^{hd}, h_1^{hd}, ...h_t^{hd}]$$

where h_0^{ld} represents the anchor heuristic for the lower dimensional representation, h_0^{hd} represents the anchor heuristic for the higher dimensional representation and h_k^X represents the inadmissible heuristics for $X = [S^{ld}, S^{ld2}, ...S^{ldn}, S^{hd}]$ and $\forall k > 0$.

Successor Generation

In MHA*, whenever a state is expanded, it successors are inserted in all inadmissible heuristic queues that are available to the search provided it has not been expanded either in the anchor or any of the inadmissible searches. This enables MHA* to effectively share paths through different heuristics that can help the search in different parts of the state-space.

When a state is expanded in MR-MHA*, the representation dimension of each successor is extracted, and accordingly are only inserted in heuristic queues which are available to that particular representation as defined in the heuristic lists. This allows MR-MHA* to effectively share paths within each representation without unnecessarily expanding states out of irrelevant heuristic queues. This is shown in line 17 in the algorithm.

2.5.2 Sharing on Demand

MR-MHA* presented in the previous section is a generalization of the SMHA* algorithm. SMHA* though is very powerful in handling nested local minima [4], can be counter productive at times when path sharing drags the search into a local minima that could have been avoided if the searches were kept independent. [4] shows an example where IMHA* terminates earlier than SMHA* which is provided in the Fig. 2.6 below

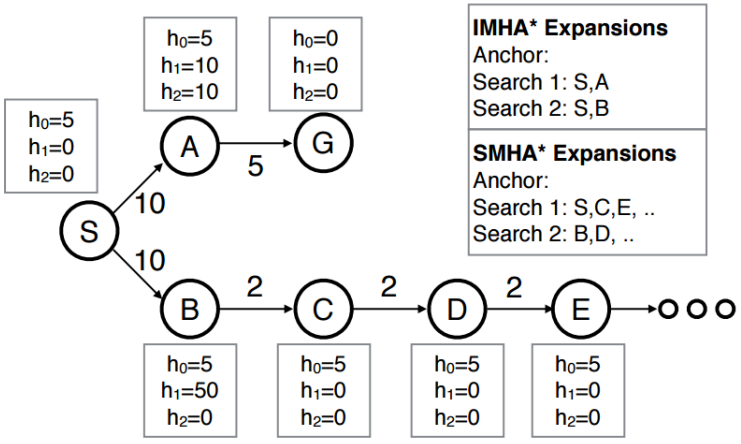


Figure 2.6: An example scenario depicting how path sharing in SMHA* can at times degrade performance

This local minima becomes more apparent in the humanoid setting where several inadmissible heuristics interact with one another and often times leads the search into a local minima similar to what we see in the figure above. To mitigate this issue, once can think of sharing between heuristic queues only when necessary and run independently otherwise. In this section, we discuss an extension to MHA* which exhibits this feature of Sharing on Demand.

Dynamic Thompson Sampling

Dynamic Thompson Sampling [12] is an algorithm which reacts to changing bandits in the Multi-Armed bandit setting. DTS maintains a β distribution for each bandit over the likelihood of the internal bernoulli parameter. [23] shows the use of the DTS for the problem of selecting the next queue to expand from in the Shared Multi-Heuristic A* framework. A picture of the β distribution is shown in Fig. 2.7 below

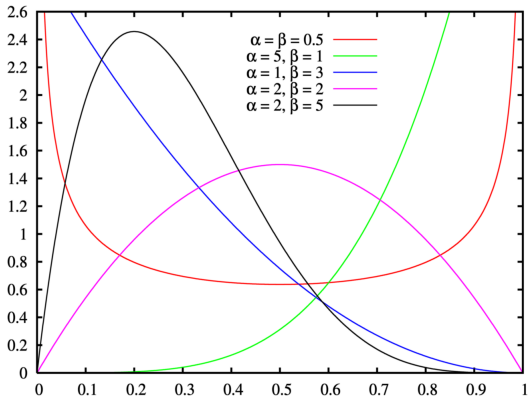


Figure 2.7: Beta distribution

We use DTS in a very similar manner as to how it is presented in [23]. The only difference is that we do not use it to choose the search to expand from. We only use it as a measure of how well a particular heuristic queue is performing. If the queue is performing well, as determined by the mean of the β distribution corresponding to the particular heuristic, then expansions from other queues cannot share successors with this queue. If the queue is not performing as well, then the queue allows sharing from other queues. The $Expand$ function of Sharing-on-Demand with MR-MHA* is mentioned in Fig.2.8 below.

```
1  procedure Expand(s)
2  Remove s from OPEN_i ∀ i in heuristic_list[s → dim];
3  for each s' in Succ(s)
4     if s' was never visited
5        g(s') = ∞; bp(s') = null;
6     if g(s') > g(s) + c(s, s')
7        g(s') = g(s) + c(s, s'); bp(s') = s;
8        if s' has not been expanded in the anchor search
9           insert/update (s') in OPEN_0 with key(s', 0);
10          if s' has not been expanded in any inadmissible search
11             for i in heuristic_list[s' → dim].inadm
12                if μ(β(i)) > constant
13                   continue
14                if key(s', i) ≤ w_2 * key(s', 0)
15                   insert/update (s') in OPEN_i with key(s', i);
```

Figure 2.8: Sharing-on-Demand

2.5.3 Experiments

In this section we draw statistics of the MR-MHA* planner by running it on $35DOF$ humanoid robot in a simulated environment. The robot has 4 limbs each $7DOF$, $6DOF$ for the Center of Mass(COM) of the robot and $1DOF$ for the Torso. We consider an environment where the robot can plan to a goal either through the stairways or the ladder. The lower dimensional representations include bipedal, ladder and crawling representations. The higher dimensional representation is the full-body humanoid representation. In all these experiments only the Bipedal and ladder low dimensional representations are active, which are tracked in the higher dimension. Also it is useful to note, as mentioned before, that the higher dimensional controllers for bipedaling and ladder climbing execute the tracking phase unless the path is non-executable by the controller (e.g. on stairways). Before we summarize the motion primitives and heuristic used in each representation, we quickly describe how a grid search in a projected state-space is used as a heuristic.

A common approach to designing heuristics for high dimensional state-spaces is to project it to a lower-dimensional grid and perform a search(eg. BFS or Dijkstra) in this grid, taking the obstacles into account. The lower dimensional search cost to get to goal is used as a heuristic for the high dimensional state.

Bipedal Representation

State-space : $6DOF(x, y, \theta)$ for each feet

Motion Primitives :

$$\Delta x_{target}, \Delta y_{target}, \Delta \theta_{target}, target = \{left\ leg, right\ leg\}$$

Heuristics :

1. Sum of 3D grid search(Dijkstra) from Goal to feet.
2. Inflate the value of the 3D grid search values only on staircases, with an inflation factor of 1.0 in the middle of the stairs and increasing outwards to the end of stairs.
3. Inflate the value of the 3D grid search values only on ladder climb platforms, with an inflation factor of 1.0 in the middle of the platform and increasing outwards so as to align the feet in front of the ladder.

Ladder Representation

State-space : $4DOF(x, y, z, \theta)$ for each limb

Motion Primitives :

$$\Delta x_{target}, \Delta y_{target}, \Delta z_{target}, \Delta \theta_{target}$$
$$target = \{left\ leg, right\ leg, left\ arm, right\ arm\}$$

Heuristics :

1. Sum of 3D grid search(Dijkstra) from Goal to all four end-effectors.

Crawl Representation

State-space : $3DOF(x, y, \theta)$ for the COM

Motion Primitives :

$$\Delta x_{COM}, \Delta y_{COM}, \Delta \theta_{COM}$$

Heuristics :

1. 3D grid search(Dijkstra) from Goal to the COM of the humanoid robot.

Humanoid Representation (Stepping)

State-space : $35DOF(\theta_1, \theta_2, ..., \theta_7)$ for each limb, $(x, y, z, roll, pitch, yaw)$ for COM, $(pitch)$ for torso

Motion Primitives :

$$\Delta\theta_1, \Delta\theta_2, \Delta\theta_3, \Delta\theta_4, \Delta\theta_5, \Delta\theta_6, \Delta\theta_7, for\ all\ four\ limbs$$

$$\Delta x, \Delta y, \Delta z, \Delta yaw, for\ all\ four\ end-effectors$$

$$\Delta x, \Delta y, \Delta z, \Delta roll, \Delta pitch, \Delta yaw, for\ COM$$

Heuristics :

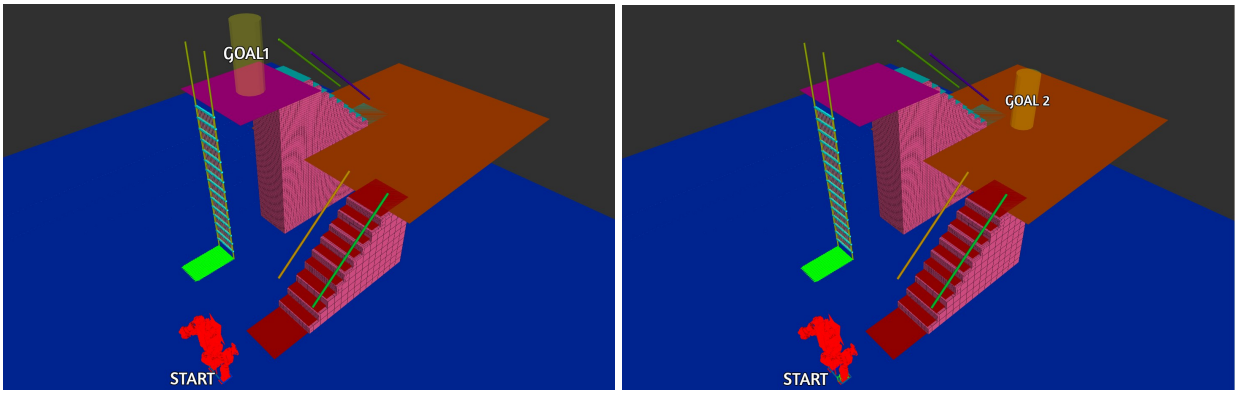
1. 3D grid search(Dijkstra) from Goal to the COM of the humanoid robot.
2. Difference in heading between the root of the robot and the feet of the robot.
3. Euclidean distance between COM of the current state and target state.
4. Euclidean distance between target feet of the current state and target state.
5. Curve to provide guidance for stepping feet movement during the search.
6. Remaining number of steps on the lower dimensional path that need to be tracked.

Planners	Goal 1		Goal 2	
	Total Time(s)	Success rate	Total Time(s)	Success rate
MR-MHA*	94.55	63.4	74.61	51.1
wA*	10.91	4.1	-	-

Table 2.1: Comparison of Total time and success rate for MR-MHA* and wA*

We make comparisons of MR-MHA* and wA* for getting to either of the goal shown in Fig. 2.9. A total of 220 random starts were sampled in the environment such that the humanoid starts in valid ground state and several statistics recorded. All experiments were ran with $w_1 = 10$ and $w_2 = 100$ for MR-MHA* and $w = 100$ for wA*. All statistic shown are averaged over all instances for each planner. It is important to note that Goal 1 can be reached both from the ladder and stairways, while Goal 2 can only be reached through the stairs. The tracking phase on the ladder is executable for the controller and hence is much faster. On the other hand, tracking on stairways is not executable and hence a full-body planning is executed in the tracking phase.

In Table. 2.1 we compare the total time and success rate between the planners for both goals from various starts. The total time includes the entirety of planning and tracking time. We can see that MR-MHA* is able to solve a lot more scenarios than wA*. Also we see that wA* has solved successful instances much faster. This is because all the instances solved successfully for wA* is a path through the ladder and none through the stairs. This becomes more apparent for Goal 2 where wA* has not solved even one instance.



(a) Goal 1 used for all our experiments

(b) Goal 2 used for all our experiments

Figure 2.9: Environment used for experiments. Random starts were sampled on the ground and planned one of the two goals shown in the figures.

Table. 2.2 and Table. 2.3 shows the planning phase and tracking phase time taken and expansions for both goals. Fig. 2.10 shows planning examples in the test environment to both goals through the stairs and the ladder.

We also conduct experiments to showcase the performance of the Sharing-on-Demand along with

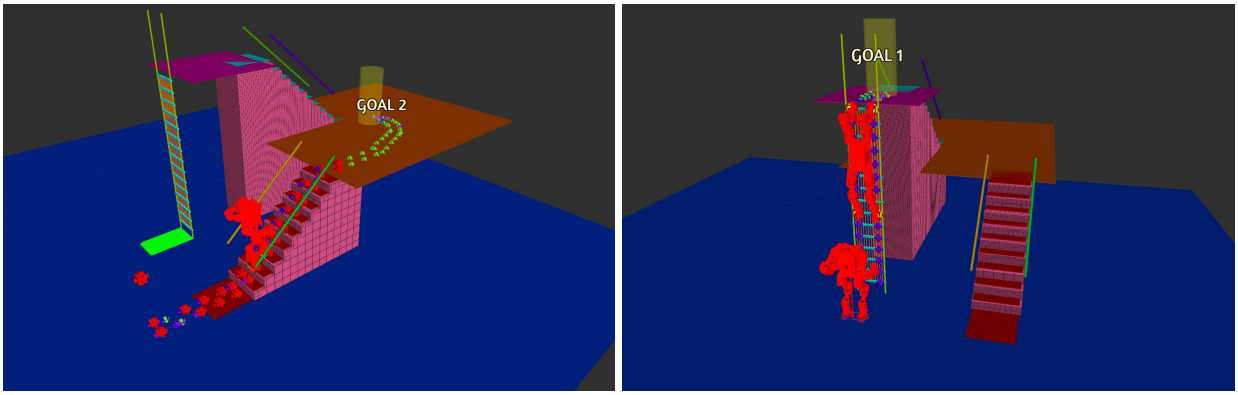
Planners	Goal 1		Goal 2	
	Planning Time(s)	Tracking Time(s)	Planning Time(s)	Tracking Time(s)
MR-MHA*	27.33	67.22	25.7	48.91
wA*	7.87	3.03	-	-

Table 2.2: Comparison of Planning phase and tracking phase times for MR-MHA* and wA*

Planners	Goal 1		Goal 2	
	Planning Exps	Tracking Exps	Planning Exps	Tracking Exps
MR-MHA*	5796	856	3583	640
wA*	146	74	-	-

Table 2.3: Comparison of Planning phase and tracking phase expansions for MR-MHA* and wA*

MR-MHA*.We use an example of bipedal footprint planning on stairs for the test. wA* is used with a consistent Dijkstra heuristic, MR-MHA* is used with a consistent Dijsktra heuristic as anchor, and quadratic inflated heuristics for stairs. We test Sharing-on-Demand with MR-MHA* with the same set of heuristics for the bipedal footprint planning. We compare the planning times and expansions of the three planners for a total of 20 randomly generated starts, and statistics averaged over all of them. We see that Sharing-on-Demand performs better than the vanilla MR-MHA* variant Fig. 2.11.



(a) A plan to Goal 2 through the stairs

(b) A plan to Goal 1 through the ladder

Figure 2.10: Example planning expansions in the test environment.

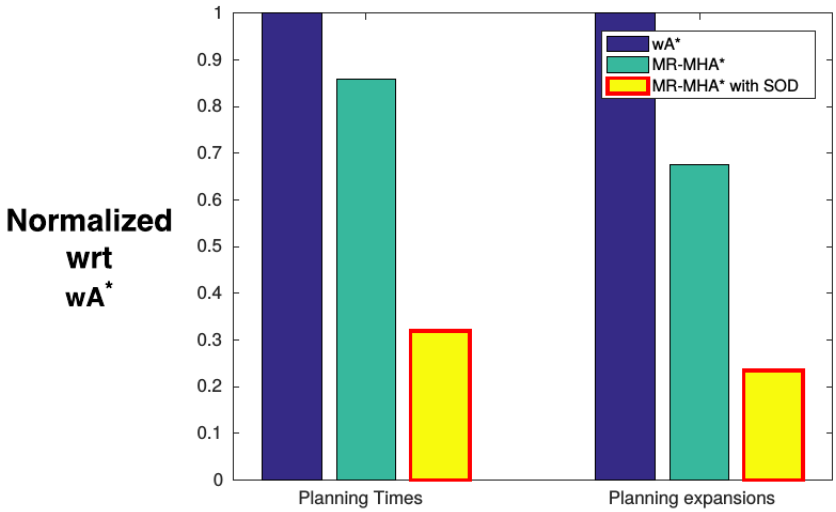


Figure 2.11: Comparison of planning times and expansions between wA*, MR-MHA* and MR-MHA* with Sharing-on-Demand(SOD) for a footprint planning example

2.6 Conclusion

In this chapter we presented a Multi-Heuristic framework that can work with multiple representations (state-spaces) in an environment. The representations own their set of heuristics and the search selectively explores across different representations to reach the goal. We also presented a extension to the algorithm, namely Sharing-on-Demand, where the heuristics for different representations share information only when needed. The framework was tested in Planning for Humanoid mobility in complex industrial environments.

Chapter 3

Learning Macro-Actions to guide Planning

3.1 Introduction

In the previous chapter, we presented a Multi-Heuristic algorithm that can work with multiple statespace representations. This algorithm built on top of Shared Multi-Heuristic A*[4], holds one key property that you can have any arbitrarily inadmissible heuristic among the set of heuristics as long as you have one consistent heuristic to satisfy sub-optimality bounds. This gives us freedom in designing and incorporating several different kinds of heuristics which can possibly help the search, especially for high DOF planning problems such as 12DOF mobile manipulation planning for PR2[4] [16][20], 35DOF mobility planning for humanoids as dealt in this thesis etc.

One major challenge of using a multi-heuristic framework for Humanoid planning is that designing several different heuristics that share information with each other fluently without introducing local minima for the search might be difficult. For example, the current set of heuristics used for Humanoid full representation, as defined in Chapter.2, include Euclidean distance between COM, Dijkstra for the feet, fitting different curves for feet movement etc. Though Sharing-on-Demand can be helpful in this regard to some extent, the problem of designing complex heuristics in the first place is challenging.

Another problem with search-based planning for high DOF planning problems such as for humanoids is that the action-set is not representative enough. A general set of actions would include incremental changes for each DOF in the planning space. Though such an action-set is representative of all movement that the robot would make, it would be more beneficial to have actions that move several different joints of the robot at the same time. Also, a humanoid operating in an industrial environment will repeat several actions over and over to complete any task assigned to it. It has been well studied in literature that a lot of these problems are intrinsically lower dimensional [7]. For example, a humanoid might execute an action trajectory several times to walk from one location in space to another, climbing a ladder might involve similar motions to reach from one rung to another, executing similar arm motion to hold handrails on stairs to balance itself etc

which might involve coordinating a lower dimensional space of movements[24], [19]. This motions essentially comprise of similar set of actions which move a set of joints together to achieve intermediary goals , like making a step. These set of actions or macro-actions can be very useful in planning for humanoids for tasks which involve a lot of repetitive motions. While it is straightforward to look at executed motions from previous plans and use them for future planning, it can difficult to generalize this to similar macro-actions that satisfy new constraints.

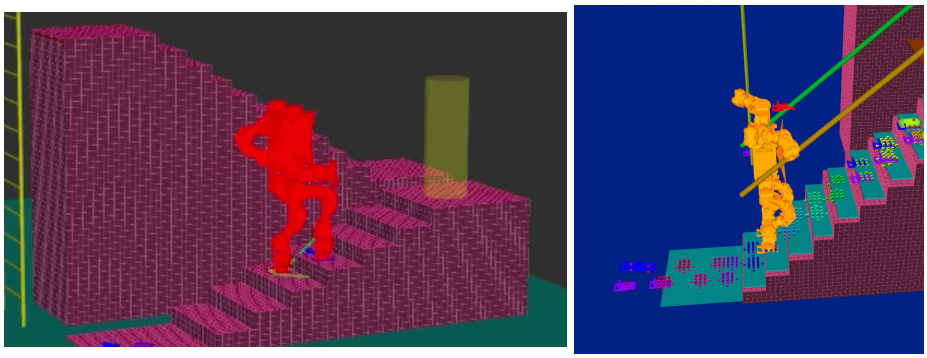


Figure 3.1: Repetive humanoid motions like stepping and staircase climbing

In this chapter, we present an approach to learn new macro-actions from previous planning data that adapt to the new constraints while maintaining temporal and spatial structure of similar motions executed previously. We look to a recently developed method called bilinear SpatioTemporal Basis models [5] that exploits spatial and temporal regularity in data while generalizing to new sequences at the same time. We use this bilinear model to learn new macro-actions that are relevant to the planning problem in hand, from spatiotemporal bases vectors learnt several such similar motions from previous planning solutions. We attempt to use these newly learnt macro-actions both as an action for the search to select and as one of the heuristics in the framework of MR-MHA* to help the search get to the solution faster.

3.2 Related Work

A good amount of previous work in the past have looked at learning movement primitives for humanoids. [25] present a method to create fundamental action blocks called Dynamic Movement primitives(DMPs) for humanoid control using nonlinear attractor systems. DMPs are a formulation of movement primitives with nonlinear differential equations, whose time evolution creates smooth kinematic control policies. Model-based control theory is used to convert the outputs of these policies into motor commands. There were natural extensions to the framework to learn from demonstrations and imitation learning[27][15][26][21].

There has also been a lot of work on the statistical side, especially in the graphics community to develop determination of motion primitives using dimensionality reduction techniques like PCA.

[19] represent a way to generate natural looking motions in an efficient manner at a dynamic level by extracting basis functions from observed human movements. [24] uses a similar approach to exploit the fact that human motions are intrinsically lower dimensional and use PCA to look at a reduced search space to perform the optimization. The difference between our approach and a PCA-based approach is that we use a Bilinear SpatioTemporal Model presented in [5], but we also don't take dynamics into account. [28] uses a Gaussian Process Models to extract human motions in lower dimensional space from high dimensional mocap data.

3.3 Bilinear SpatioTemporal Models

We summarize the model presented in [5] in this section.

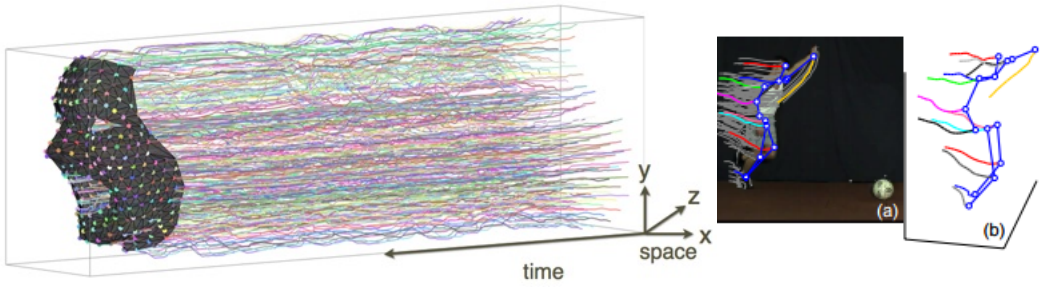


Figure 3.2: Bilinear Spatio-Temporal Models

Consider a set of P n-dimensional points sampled at F time instances. Such as a set of points can be represented in the form of the matrix as below

$$S = \begin{bmatrix} X_1^1 & X_2^1 & . & . & X_P^1 \\ X_1^2 & X_2^2 & . & . & X_P^2 \\ . & . & . & . & . \\ . & . & . & . & . \\ X_1^F & X_2^F & . & . & X_P^F \end{bmatrix}$$

where each $X = [p^1, p^2, ...p^n]$.

The spatial regularity in the data represented in S can be represented by concatenating each n-D shape in a row-wise manner.

$$S = \sigma B^T$$

where B is a $nP \times K_s$ matrix containing K_s basis vectors for the spatial direction and σ corresponds to the spatial coefficients of the data of size $F \times K_s$.

An alternative representation is to model the temporal regularity in the data S. This is done by concatenating P n-D trajectories in a column-wise manner.

$$S = \theta A^T$$

where θ is a $F \times K_t$ matrix containing K_t basis vectors for the temporal direction and A corresponds to the temporal coefficients of the data of size $nP \times K_t$.

The key idea presented in [5] is that a spatial or temporal basis individually fails to generalize spatiotemporal regularities. If S can be represented as $S = \sigma B^T$ and as $S = \theta A^T$ then there exists a factorization

$$S = \theta C B^T$$

where $C = \theta\sigma = A^T B$ is $K_t \times K_s$ matrix of spatiotemporal coefficients. This allows to specify a trajectory in both the spatial and temporal bases through a single set of coefficients in C [5].

3.4 Learning Macro-Actions

In this section, we present our method to use Bilinear SpatioTemporal models to learn macro-actions suitable to Humanoid motions which can be further used in Planning. As mentioned earlier we know that the humanoid executes several similar motions when performing a task. Through this approach we aim to learn several bilinear models for several distinct behaviors of a humanoid, such as stepping, holding handrails, etc. Once we extract the bases, we optimize for a new set of coefficients that corresponds to a new macro-action which satisfies the constraints of the environment. This optimization as we will show later is much faster than optimizing over the entire trajectory space. Also the bases vectors in the spatial and temporal direction makes the resulting motion similar to the ones we have seen before satisfying the newer constraints. We summarize our approach below

3.4.1 Approach

Training Phase : Learning the SpatioTemporal basis

1. We use a database of several previous plans of humanoid mobility. Each plan consists of several different behavioral motions such as stepping, arm motions etc. We manually segment the various behaviors and group them accordingly. This process can also be automated through a probabilistic PCA or using Gaussian Mixture Models [6].

2. Each clustered motion behavior consists of a set of similar trajectories. We make sure that all trajectories are of the same dimension in the temporal dimension, we uniformly sample the same number of time steps in each trajectory. For n behavioral motions M, each with a set of trajectories we have

$$M_1 = T_1^1, T_1^2, T_1^3 ...$$
$$M_2 = T_2^1, T_2^2, T_2^3 ...$$
$$.$$
$$.$$
$$M_n = T_n^1, T_n^2, T_n^3 ...$$

where T_i^j would represent the j^{th} trajectory in the i^{th} motion.

3. For each trajectory T_i^j, we extract the actions in each trajectory so as to obtain a set- of actions that define each kind of behavior. This is done so that the behaviors we learn the bases from are not subject to the pose of the robot, and are only concerned with the actions that each DOF is subject to. To extract the macro-action corresponding to each trajectory, we compute the differences between consecutive waypoints in each trajectory. This can represented through a simple linear transform.

$$S = RT$$

$$R = \begin{bmatrix} 0 & 0 & . & . & 0 \\ -1 & 1 & 0 & . & 0 \\ 0 & -1 & 1 & . & 0 \\ . & . & . & . & . \\ 0 & 0 & . & -1 & 1 \end{bmatrix}$$

4. For each of these macro-action trajectories S_i^j we do a polar mapping for values in the circular dimension. This is because certain DOFs of the trajectory such as rotational joints are values in the circular dimensions. Hence we project this angular dimension into the real dimension through the following projection into the $\mathbb{R}$ Fig. 3.3.

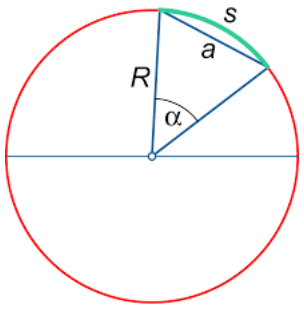


Figure 3.3: Projecting circular dimension in the Real space

The projection is represented through the equation as stated below

$$a = 2Rsin(\alpha/2)$$

where α is the value in the circular dimension projected to a in the real dimension. R can be set as 1.

5. We have a set of macro-actions S_i^j defined for each behavior motion, where each DOF of every time step are mapped onto the $\mathbb{R}$ space. Hence for any behavior M_i we have

$$M_i = S_i^1, S_i^2, S_i^3 ..., S_i^P$$

We concatenate these macro-actions in the spatial and temporal direction and extract Bilinear SpatioTemporal bases for each motion, predefining the temporal bases as DCTs.

$$S_i^1 = \theta_i C_i^1 B_i^T$$
$$S_i^2 = \theta_i C_i^2 B_i^T$$
$$.$$
$$.$$
$$S_i^P = \theta_i C_i^P B_i^T$$

For all macro-actions corresponding to a behavior motion, we have extarcted a single temporal basis θ_i, a single spatial basis B_i and a set of coefficient matrices C_i^j each corresponding to a macro-action S_i^j. It is an important point to note that we always select the most important basis vectors that explain the data in the spatial and temporal direction. Hence if each macro-action is a matrix of size $F \times n$ where F is the number of time steps and n is the number of DOFs, then the number of spatial basis vectors $K_s \leq n$ and number of temporal basis vectors $K_t \leq F$. Usually the number of bases vectors is much less than the dimension of the space. The matrix C representing the coefficients is of size $K_t \times K_s$.

Test Phase : Generating a new macro-action

In the test phase, the goal is to generate a new macro-action given the learnt bases vectors and set of constraints. hence we need to optimize for a new coefficient matrix C which satisfies the new constraints provided during the test phase.

$$\min_C \sum_j \|\mathbf{C} - \mathbf{C_i^j}\|_F \tag{3.1}$$

subject to

$$g(C) \leq 0 \tag{3.2}$$

$$h(C) = 0 \tag{3.3}$$

where $\|A\|_F$ represents the Frobenius norm of the difference between optimized coefficient matrix C and the coefficient matrices C_i^j of trajectories of behavior motion i from which the basis have been learnt before. Hence the optimization objective in Eq.(3.1) tries to converge to a coefficient matrix C that is similar coefficients of trajectories seen before. $g(C)$ represents inequality constraints such as joint limits of the humanoid robot. $h(C)$ represents the equality constraints such as end-effector position of certain limbs, contact constraints etc.

3.5 Macro-Actions in Planning

The macro-actions generated in the optimization procedure mentioned above is used as a pre-computation step before planning starts. From previously generated planning data, spatiotemporal basis pertaining to different behaviors are learnt. When a new planning request is generated in a different environment, the constraints from the environment are used to optimize for a new C_i for every behavior i. These new constraints can be different stair heights, step lengths, handrail height and orientation etc.

The most common approach to using macro-actions in planning itself is to use them as one of the available options in the action-set. While this can be useful, the macro-actions generated may often be invalid since we don't have constraints on collision with the world, self-collision etc. Hence we follow an approach similar to E-graphs [22] in effectively using these macro-actions with the MR-MHA* framework. E-graphs use prior plans to accelerate the plans of new planning requests by using heuristics to bias the search towards these prior experience and avoiding to search large parts of the state-space.

3.5.1 Heuristics

We consider a nominal standing pose of the humanoid and apply the learnt macro-actions to these nominal poses. The trajectories generated out of these are used as heuristics for the search in the multi-heuristic framework. We follow a pure-pursuit approach in using trajectories as heuristics. Fig. 3.4

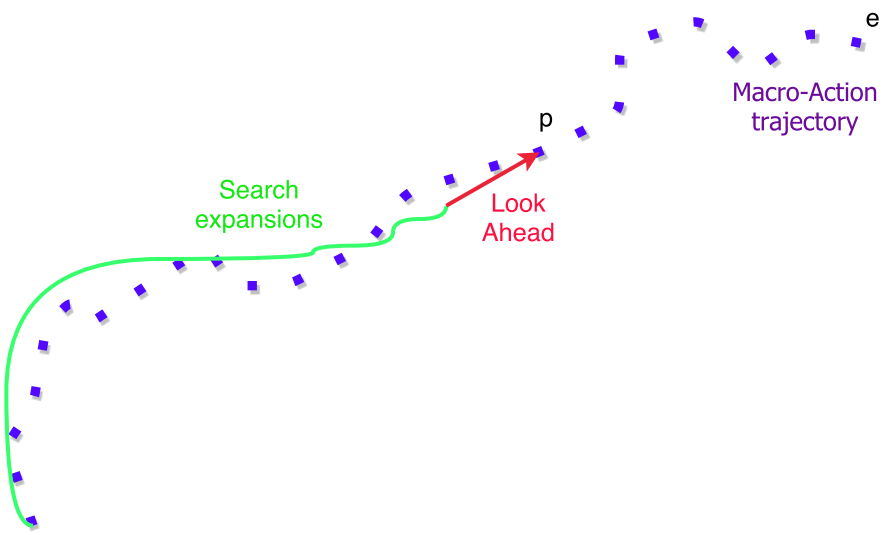


Figure 3.4: Pure pursuit approach to using trajectories as heuristics

The heuristic of a state is calculated as

$$h(state) = dist(state, p) + dist(p, e)$$

where p is the lookahead waypoint and e is the end waypoint on the macro-action trajectory. We get the distance to the macro-action trajectory shown as distance to point p. To follow the trajectory we also add the distance along the trajectory from point p to end point e to guide the search to follow the trajectory.

3.5.2 Adaptive Motion Primitives

As discussed earlier, depending on the current state of expansion during the search, a trajectory for each macro-action is generated from a chosen nominal pose and added as an experience in the search graph. Hence while selecting the next best action, the search also considers snapping to one of these states on the trajectory in addition to other actions available in its action-set. The heuristic, explained above, in addition to these snapping actions, guides the search to execute the macro-action trajectories appropriately.

3.6 Experiments

We perform preliminary experiments to extract macro-actions for stepping actions of humanoid robot and testing it in planning for climbing stairways. We collected 100 different stepping motions from previous plans with weak heuristics where the environment step height ranged from

Optimization statistics			
Macro-action	Time(s)	Spatial Basis vectors	Temporal Basis vectors
Left(Both feet start at the same height)	15.2	6	6
Right(Both feet start at the same height)	8.4	6	6
Left(Both feet start at a step height difference)	9.17	6	6
Right(Both feet start at a step height difference)	17.31	6	6

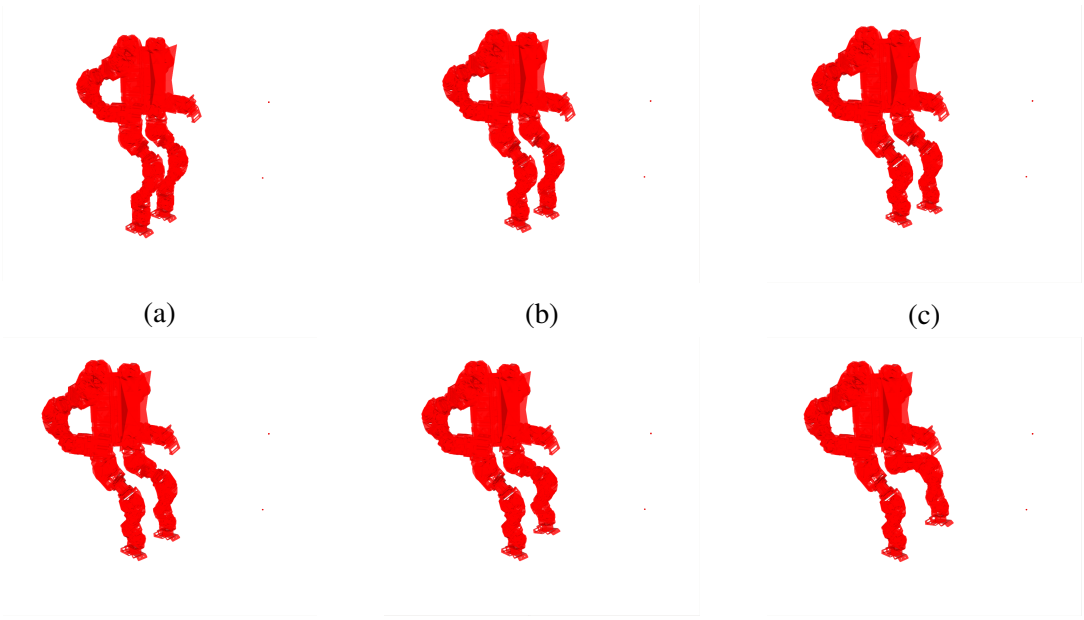
Table 3.1: Optimization statistics for stepping macro-actions.

$0.1 - 0.2m$ and step length of $0.2m$. The left stepping motions and right stepping motions were segmented into separate behaviors and spatiotemporal bilinear basis were learnt for both the stepping behaviors.

Each action trajectory was of size 50×17 where 50 is the number of time steps and 17 is the number of DOFs considered. Note that we did not consider the entire 35DOF since we already know that the arms did not move at all during any of these plans. The 17 DOF corresponds to 7 DOF for each leg, and 3 for the COM(x, y, z). From our experiments we determined that 6 basis vectors each for the spatial and temporal basis, (i.e) $K_s = 6$ and $K_t = 6$, worked well in reconstructing the data and generating new macro-actions. Note that the number of basis selected is much less than the full dimensionality of the spatial and temporal direction and helps speed up the optimization of coefficient matrix C.

The environment of the new planning problem consisted of a step height of $0.25m$ and step length of $0.3m$ both of which have not been seen before. Hence we solve the optimization presented in Eq.(3.1) to determine a new C. The inequality constraints are the joint limits of the robot and equality constraints correspond to the new step length and step height.

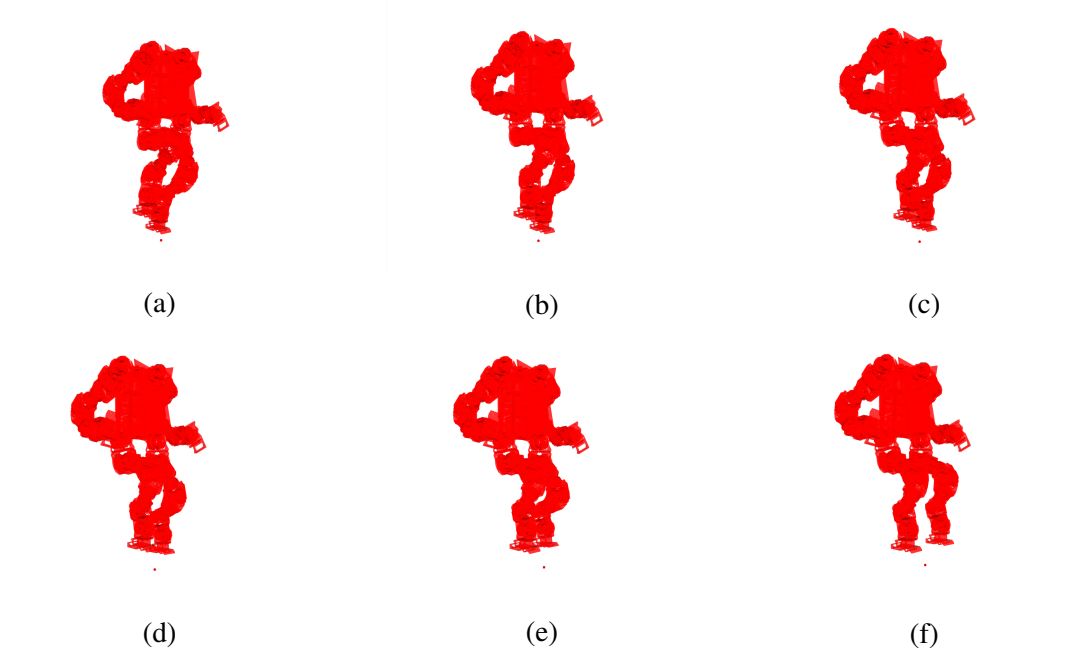
We use the Sequential Quadratic Programming(SQP) implementation in Matlab to determine the C which satisfies the constraints. Four macro-actions were generated two each for the right feet and left feet step-up. The optimization time taken to generate new macro-action and the number of basis vectors used for each macro-action is specified in the table.



(a) (b) (c)

(d) (e) (f)

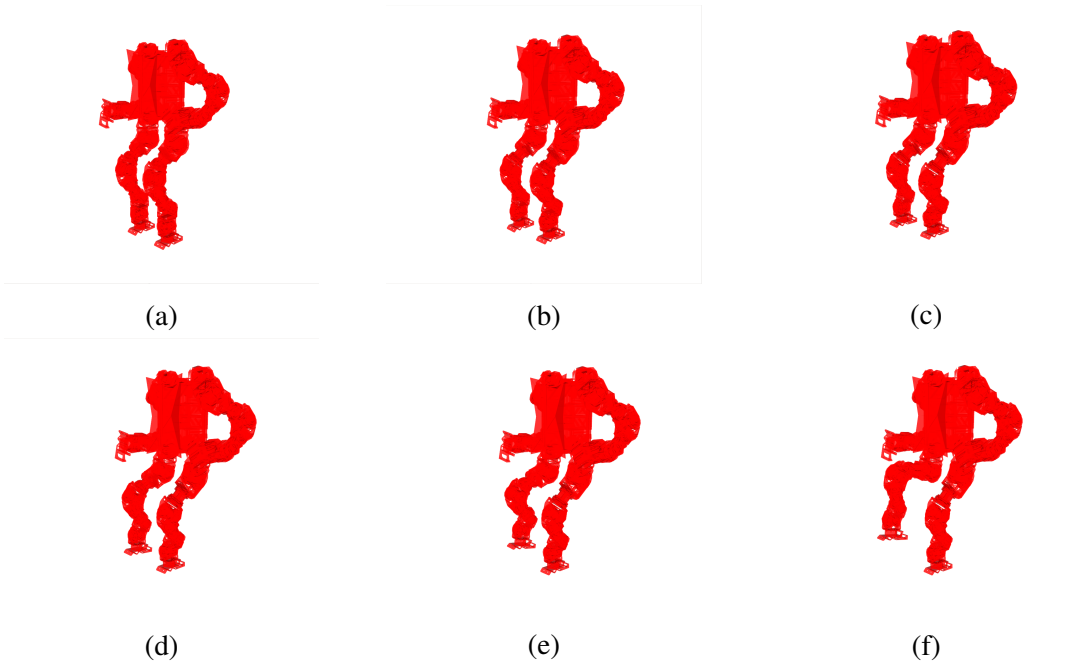
Figure 3.5: Macro-action for the left foot. Both feet start at the same level and the left foot is moved up and forward to satisfy the step length and step height.View clockwise.



(a) (b) (c)

(d) (e) (f)

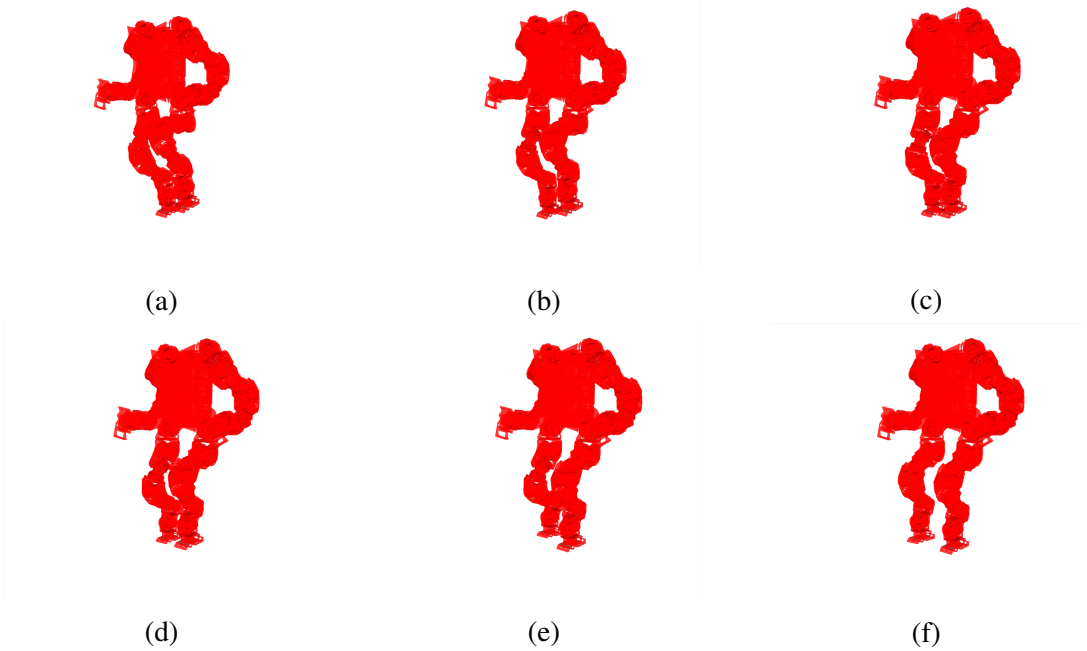
Figure 3.6: Macro-action for the left foot. Both feet start at difference of step height and the left foot is moved up and forward to satisfy the step height and step length. View clockwise.



(a) (b) (c)

(d) (e) (f)

Figure 3.7: Macro-action for the right foot. Both feet start at the same level and the right foot is moved up and forward to satisfy the step length and step height.View clockwise.



(a) (b) (c)

(d) (e) (f)

Figure 3.8: Macro-action for the right foot. Both feet start at difference of step height and the right foot is moved up and forward to satisfy the step height and step length.View clockwise.

As mentioned before the macro-actions are generated as a pre-computation step. Once these

macro-actions are generated, they are used in the planning as heuristics and snapping actions as mentioned before. We perform an experiment of climbing on top of a single flight of stairs with and without macro-actions. In the case of planning without macro-action heuristics, the search times out in the tracking phase, failing to find a plan for a timeout of $180s$. With the presence of macro-actions, the search in the tracking phase is able to find a solution in $44s$. Fig. 3.9 shows the expansions(red) during planning along with the heuristic suggestions(yellow).

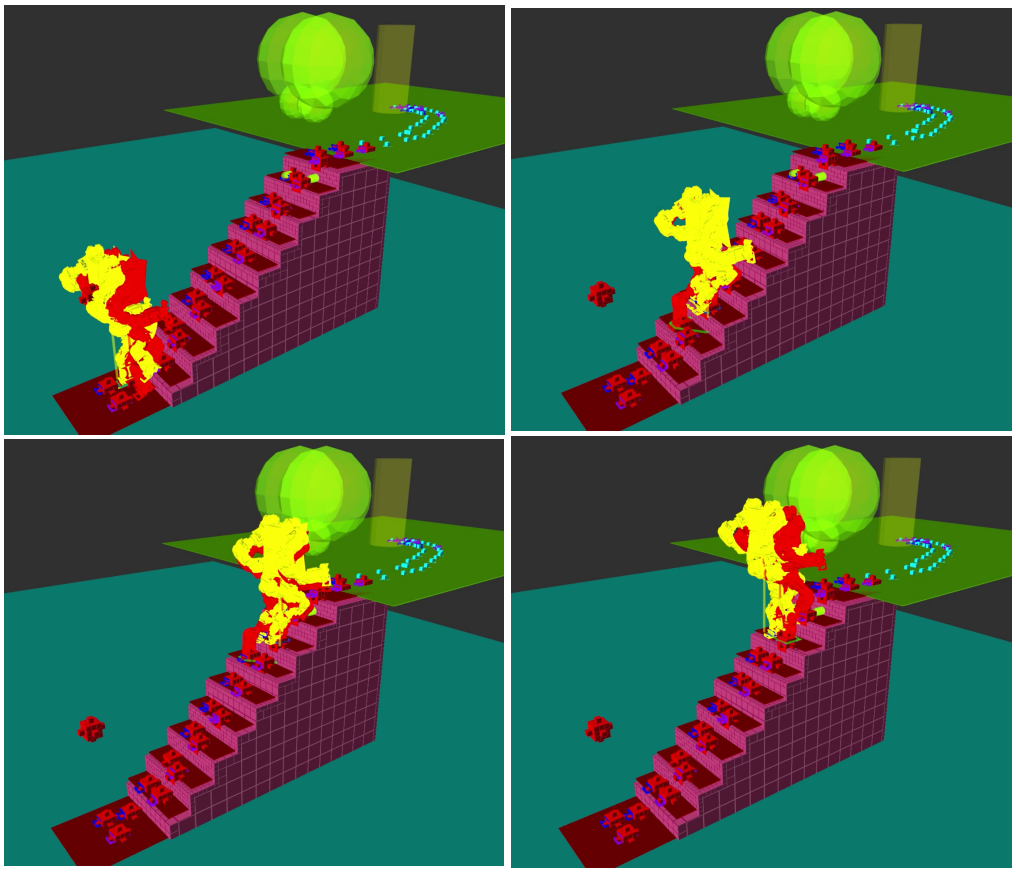


Figure 3.9: Planning with macro-actions as heuristics and snap motions. Red colour shows the expansions and yellow color shows the macro-action heuristic suggestions.

3.7 Conclusion

In this chapter we presented an approach to learn useful macro-actions from previous planning data using Bilinear SpatioTemporal basis models. We also presented a way to use these macro-actions as snapping motions and heuristics in the search. We showed some preliminary results for humanoid mobility planning on a staircase.

Chapter 4

Conclusion

In this thesis we have presented a Multi-heuristic framework for planning hor humanoid mobility in industrial environments. Industrial environments present a challenge of multiple state-space representations that we have to plan in. In this thesis we introduce MR-MHA*, as a generalization of the MHA* algorithm, that is used on top of the adaptive dimensionality framework to produce a plan that searches among all these state-space representations simultaneously. We also talk about an extension to MHA* called Sharing-on-Demand where we discuss how limited sharing between several heuristic queues can be useful while having a set of heuristics which might potentially pollute each other.

We also discuss an approach to learning macro-actions for full-body planning for humanoids. We extract a bilinear bases models in both the spatial and temporal direction to represent several different humanoid behaviors. Using these bases models we optimize for new macro-actions subject to newer constraints. We then show how these macro-actions can be used in motion planning for humanoids and provide preliminary results on full-body planning.

Bibliography

[1] Honda motor corp. the honda humanoid robot asimo. URL http://world.honda.com/ASIMO. 1

[2] Yaskawa electric corp. motoman-sda10. URL http://www.yaskawa.co.jp/en/newsrelease/2007/02.htm. 1

[3] Honda motor corp. the honda humanoid robot asimo. URL http://world.honda.com/ASIMO. 1

[4] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic a*. Berkeley, USA, July 2014. doi: 10.15607/RSS.2014.X.056. 2.5, 2.5.2, 3.1

[5] Ijaz Akhter, Tomas Simon, Sohaib Khan, Iain Matthews, and Yaser Sheikh. Bilinear spatiotemporal basis models. *ACM Transactions on Graphics*, 31(2):17:1–17:12, April 2012. doi: 10.1145/2159516.2159523. 3.1, 3.2, 3.3, 3.3

[6] Jernej Barbič, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K. Hodgins, and Nancy S. Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004*, GI '04, pages 185–194, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society. ISBN 1-56881-227-2. URL http://dl.acm.org/citation.cfm?id=1006058.1006081. 1

[7] Matei T. Ciocarlie and Peter K. Allen. Hand posture subspaces for dexterous robotic grasping. *Int. J. Rob. Res.*, 28(7):851–867, July 2009. ISSN 0278-3649. doi: 10.1177/0278364909105606. URL http://dx.doi.org/10.1177/0278364909105606. 3.1

[8] Marco Cognetti, Pouya Mohammadi, and Giuseppe Oriolo. Whole-body motion planning for humanoids based on com movement primitives. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 1090–1095. IEEE, 2015. 1.1

[9] Johannes Garimort, Armin Hornung, and Maren Bennewitz. Humanoid navigation with dynamic footstep plans. *2011 IEEE International Conference on Robotics and Automation*, pages 3982–3987, 2011. 1.1

[10] Kalin Gochev. Planning with adaptive dimensionality. In *Publicly Accessible Penn Dissertations. 1739.*, 2016. URL http://repository.upenn.edu/edissertations/

1739. 2.4, 2.4

[11] Kalin Gochev, Benjamin J. Cohen, Jonathan Butzke, Alla Safonova, and Maxim Likhachev. Path planning with adaptive dimensionality. In *Proceedings of the Fourth Annual Symposium on Combinatorial Search, SOCS 2011, Castell de Cardona, Barcelona, Spain, July 15.16, 2011*, 2011. URL http://www.aaai.org/ocs/index.php/SOCS/SOCS11/paper/view/4037. 2.1, 2.2, 2.3

[12] Neha Gupta, Ole-Christoffer Granmo, and Ashok Agrawala. Thompson sampling for dynamic multi-armed bandits. In *Proceedings of the 2011 10th International Conference on Machine Learning and Applications and Workshops - Volume 01*, ICMLA '11, pages 484–489, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4607-0. doi: 10.1109/ICMLA.2011.144. URL http://dx.doi.org/10.1109/ICMLA.2011.144. 2.5.2

[13] Armin Hornung, Daniel Maier, and Maren Bennewitz. Search-based footstep planning. 1.1

[14] Armin Hornung, Andrew Dornbush, Maxim Likhachev, and Maren Bennewitz. Anytime search-based footstep planning with suboptimality bounds. In *Humanoids*, 2012. 1.1

[15] J. A. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *International Conference on Robotics and Automation (ICRA2002)*, Washinton, May 11-15 2002, 2002. URL http://www-clmc.usc.edu/publications/I/ijspeert-ICRA2002.pdf. 3.2

[16] Fahad Islam, Venkatraman Narayanan, and Maxim Likhachev. Dynamic multi-heuristic a. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2376–2382. IEEE, 2015. 3.1

[17] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. *Motion Planning for Humanoid Robots*, pages 365–374. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31508-7. doi: 10.1007/11008941_39. URL https://doi.org/10.1007/11008941_39. 1.1

[18] James J. Kuffner, Satoshi Kagami, Koichi Nishiwaki, Masayuki Inaba, and Hirochika Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, Jan 2002. ISSN 1573-7527. doi: 10.1023/A:1013219111657. URL https://doi.org/10.1023/A:1013219111657. 1.1

[19] Bokman Lim, Syungkwon Ra, and Frank C Park. Movement primitives, principal component analysis, and the efficient generation of natural motions. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4630–4635. IEEE, 2005. 3.1, 3.2

[20] Venkatraman Narayanan, Sandip Aine, and Maxim Likhachev. Improved multi-heuristic a* for searching with uncalibrated heuristics. In *Eighth Annual Symposium on Combinatorial Search*, 2015. 2.5, 3.1

[21] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Proceedings of the 2009 IEEE*

International Conference on Robotics and Automation, ICRA'09, pages 1293–1298, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2788-8. URL http://dl.acm.org/citation.cfm?id=1703435.1703645. 3.2

[22] Mike Phillips, Benjamin J Cohen, Sachin Chitta, and Maxim Likhachev. E-graphs: Bootstrapping planning with experience graphs. 2012. 3.5

[23] Mike Phillips, Venkatraman Narayanan, Sandip Aine, and Maxim Likhachev. Efficient search with an ensemble of heuristics. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 784–791. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/116. 2.5.2, 2.5.2

[24] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 514–521, New York, NY, USA, 2004. ACM. doi: 10.1145/1186562.1015754. URL http://doi.acm.org/10.1145/1186562.1015754. 3.1, 3.2

[25] S. Schaal. Dynamic movement primitives - a framework for motor control in humans and humanoid robots. In *The International Symposium on Adaptive Motion of Animals and Machines*, Kyoto, Japan, March 4-8, 2003, March 2003. URL http://www-clmc.usc.edu/publications/S/schaal-AMAM2003.pdf. 3.2

[26] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Control, planning, learning, and imitation with dynamic movement primitives. In *IROS 2003*, pages 1–21. Max-Planck-Gesellschaft, October 2003. 3.2

[27] S. Schaal, J. Peters, J. Nakanishi, and A.J. Ijspeert. Learning Movement Primitives. In *International Symposium on Robotics Research (ISRR2003)*, 2003. 3.2

[28] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models. In *In NIPS*, pages 1441–1448. MIT Press, 2006. 3.2

[29] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. doi: 10.1145/1275808.1276509. URL http://doi.acm.org/10.1145/1275808.1276509. 2.4