Online, interactive user guidance for high-dimensional, constrained motion planning

Fahad Islam[†], Oren Salzman[†] and Maxim Likhachev[†*]

Abstract

We consider the problem of planning a collision-free path for a high-dimensional robot. Specifically, we suggest a planning framework where a motion-planning algorithm can obtain guidance from a user. In contrast to existing approaches, we suggest to seek user guidance only when the planner identifies that it ceases to make significant progress towards the goal. User guidance is given in the form of an intermediate configuration $\hat{q}$ which, in turn, is used to bias the planner to go through $\hat{q}$. We demonstrate our approach for the case where the planning algorithm is Multi-Heuristic A* (MHA*) and the robot is a 34-DOF humanoid. We show that using this general approach allows to compute highly-constrained paths with little domain knowledge. Without our approach, solving such problems require carefully-crafted domain-dependent heuristics.

1 Introduction

Motion-planning is a fundamental problem in robotics that has been studied for over four decades [Choset *et al.*, 2005; LaValle, 2006; Sharir, 2004]. However, efficiently planning paths in high-dimensional, constrained spaces remains an ongoing challenge. One approach to address this challenge is to incorporate user input to guide the motion-planning algorithm. While there has been much work on planning using human demonstration (see, e.g., [Argall *et al.*, 2009; Holladay and Srinivasa, 2016; Phillips *et al.*, 2016; Schulman *et al.*, 2016; Ye and Alterovitz, 2017]), there has been far less research incorporating guidance as an interactive part of the planning loop.

Broadly speaking, interactive planning has been typically used in the context of sampling-based motion-planning algorithms [LaValle, 2006]. User guidance was employed by biasing the sampling scheme of the planner. This was done by having the user mark regions in the *workspace* that should be avoided or explored [Denny *et al.*, 2014; Mehta *et al.*, 2015; Yan *et al.*, 2015; Ranganeni *et al.*, 2017]. Alternatively, interactive devices such as a 3D mouse or a haptic arm have been used to generate paths in a (low-dimensional) configuration space. This path was then used by a planner to bias its sampling domain [Blin *et al.*, 2016; Flavigne *et al.*, 2009; Taïx *et al.*, 2012].

We are interested in planning in high-dimensional, constrained spaces such as those encountered by a humanoid robot (see Fig. 1 and Sec. 2.1). In such settings, workspace regions often give little guidance to the planner due to the high dimension of the configuration space as well as the physical constraints of the robot. Additionally, obtaining user guidance in the configuration space is extremely time consuming, even for expert users. Thus, while beneficial, user guidance should be employed scarcely.

Our key insight is that carefully chosen individual configurations suggested by a user can be used to effectively guide the planner when in need. Transforming this insight into a planning framework requires addressing three fundamental questions:

Q1. When should the planner ask the user for guidance?

Q2. What form should the user's guidance take?

Q3. How should the guidance be used by the planner?

Identifying *when* to obtain guidance (Q1, Sec 4.1) comes in stark contrast to existing approaches—we suggest to only employ user guidance when the planner *identifies* that it ceases to make significant progress towards the goal. The specific type of guidance given (Q2, Sec 4.2), as previously mentioned, is configuration-space based and *not* workspace-based. This deviation from previous work is due to our specific setting of high-dimensional, constrained systems. Finally, guidance is used to *bias* the search algorithm towards regions that are likely to be beneficial (Q3, Sec 4.2). It is worth emphasising that this is done without requiring the user to understand the underlying search algorithm.

While our approach is general and can be incorporated with any motion-planning algorithm (see discussion in Sec. 6) it is especially suitable for search-based planning algorithms (see, e.g., [Cohen *et al.*, 2014]) that perform a systematic search guided by heuristic functions. Specifically, we demonstrate its effectiveness for the case where the motion-planning algorithm is multi-heuristic A* (MHA*) [Aine *et al.*, 2016; Narayanan *et al.*, 2015] which we detail in Sec. 2.2.

[*†]Robotics Institute, Carnegie Mellon University, Pittsburgh, USA {fi,osalzman} at andrew.cmu.edu, maxim@cs.cmu.edu

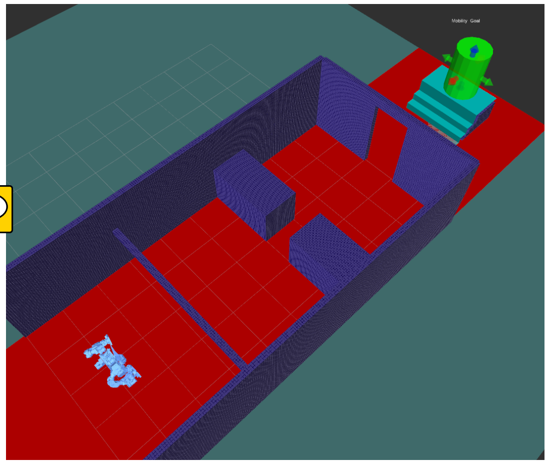


Figure 1: Planning domain—Humanoid robot needs to plan a path in a challenging environment that involves circumventing obstacles and passing through tight spaces.

After describing our approach at high-level (Sec. 3) we demonstrate how it can be applied to the case of MHA* (Sec. 4). We continue by showing the effectiveness of our planner (Sec. 5). Specifically, we show that this general approach allows to compute highly-constrained paths such as climbing stairs with little domain knowledge. Without our approach, solving such problems require carefully-crafted domain-dependent heuristics. We conclude this paper with a discussion and an outline of possible future work (Sec. 6).

2 Related work and algorithmic background

2.1 Motion planning for humanoid robots

Humanoid robots, for which we demonstrate our approach, often have several dozens of degrees of freedom making planning a challenging task, especially when taking additional constraints into account such as stability and contact constraints. One approach to plan the motion for such systems, is to use predefined, carefully chosen fixed gaits [Kaneko *et al.*, 2004]. However, when the terrain is uneven, such planners are inadequate at computing stable motions [Hauser *et al.*, 2008]. Another approach is to reduce the search space by decomposing the degrees of freedom into functional groups such as locomotion and manipulation. Then functional-specific algorithms such as footstep planning are applied to the low-dimensional space (see, e.g., [Chestnutt *et al.*, 2005; Kuffner *et al.*, 2001; Perrin *et al.*, 2012; Xia *et al.*, 2009; Jr. *et al.*, 2002] for a partial list). A high-dimensional planner is then used to "track" the plan generated in the low-dimensional space.

User guidance has been intensively incorporated in controlling the motion of humanoid robots, especially in complex tasks as those presented in the Darpa Robotics Challenge (DRC) [Spenko *et al.*, 2018]. In such settings, guidance ranged from teleoperating the robot through shared autonomy to high-level task guidance [McGill *et al.*, 2017; Zucker *et al.*, 2015; Gray *et al.*, 2017; DeDonato *et al.*, 2017; Marion *et al.*, 2017]. However, as far as we are aware of, in all the aforementioned cases, *identification* of when to use guidance was done by a human operator and not by the system (except for relatively simple metrics where the system asks for help when it fails to complete some given task). Furthermore, the human guidance used was used as a "hard" constraint forcing the system to make use of the guidance. In contrast, in our work the system automatically and independently identifies when it is in need of guidance. This guidance is then seamlessly incorporated as a soft constraint— the system biases the search towards the guidance but also continues to explore the search space as if the guidance was not given.

2.2 Multi Heuristic A* (MHA*)

Multi Heuristic A* (MHA*) [Aine *et al.*, 2016; Narayanan *et al.*, 2015] is a search-based planning algorithm that takes in multiple, possibly inadmissible heuristic functions in addition to a single consistent heuristic termed the *anchor* heuristic. It then uses the heuristic functions to simultaneously perform a set of weighted-A* [Pohl, 1970]-like searches. Using multiple searches allows the algorithm to efficiently combine the guiding powers of the different heuristic functions.

Specifically, for each search, MHA* uses a separate priority queue associated with each heuristic. The algorithm iterates between the searches in a structured manner that ensures bounds on sub-optimality. This can be done in a round-robin fashion, or using more sophisticated approaches that allow to automatically calibrate the weight given to each heuristic [Phillips *et al.*, 2015].

Part of the efficiency of MHA* is due to the fact that the value of the cost-to-come (the g-value) computed for each state is shared between all the different searches[1]. Sharing cost-to-come values between searches implies that if a better path to a state is discovered by any of the searches, the information is updated in all the priority queues. Moreover, if one search ceases to make progress towards the goal (a state which we call "stagnation region" and which will be formally defined in Sec. 4) it can use "promising" states found by other searches to escape this stagnation region ~~(see also [Hernández and Baier, 2012; Ishida, 1992])~~. Furthermore, path sharing allows MHA* to expand each state at most twice. ~~Finally, in graph search, heuristic functions give a principled way to identify when the planner ceases to make progress (see, e.g., [Vats *et al.*, 2017]).~~

2.3 Identifying progress in search-based planners

A key component in our work is automatically detecting when our planner requires user guidance. This requires *characterizing* regions where the planner ceases to make progress and algorithmic tools to *detect* such regions. These two requirements are closely related to the notion of *heuristic depressions* [Ishida, 1992] and *expansion delays* [Dionne *et al.*, 2011; Burns *et al.*, 2013].

[1]To be precise, Aine et al. [Aine *et al.*, 2016] define two variants of MHA*: Independent and Shared MHA* where the queues do not share and do share the g-values of states, respectively. In this paper when we use the term MHA*, it refers to the latter (shared) variant.

Algorithm 1 User-guided planning ($\mathcal{A}$)

1: **while** $\neg\mathcal{A}$.`is_solution_found()` **do**
2: **while** $\neg\mathcal{A}$.`is_in_stagnation_region()` **do**
3: $\mathcal{A}$.`run()` ▷ no user guidance
4: $g \leftarrow$ `get_user_guidance()` ▷ $\mathcal{A}$ is in a stagnation region
5: $\mathcal{A}$.`update_user_guidance(`g`)` ▷ account for guidance
6: **while** $\mathcal{A}$.`is_in_stagnation_region()` **do**
7: $\mathcal{A}$.`run()` ▷ $\mathcal{A}$ uses guidance to escape stagnation region
8: $\mathcal{A}$.`update_user_guidance(`$\neg g$`)` ▷ remove guidance

A heuristic depression region is a region in the search space where the correlation between the heuristic values and the actual cost-to-go is weak. It is defined as a maximal connected component of states $\mathcal{D}$ such that all states in the boundary of $\mathcal{D}$ have a heuristic value that is greater than or equal to the heuristic value of any state in $\mathcal{D}$.

Such regions often occur in real-time search algorithms such as LRTA* [Korf, 1990] and LSS-LRTA* [Koenig and Sun, 2009] where the heuristic function is updated as the search progresses. Subsequently, current state-of-the-art algorithms guide the search to avoide states that have been marked as part of a heuristic depression [Hernández and Baier, 2012]. As we will see, we will use a slightly different characterization of when the planner ceases to make progress which we call *stagnation regions*. For details, see Sec. **??**.

Expansion delay, defined as the average number of node expansions from when a node is generated until it is expanded, is a tool used to estimate the average progress that a planner makes along any single path. When the heuristic used by a planner is perfect, the expansion delay equals one, while when performing uniform-cost search, the expansion delay can grow exponentially. We will use the notion of expansion delays to identify when the planner is in a stagnation region (i.e., when it ceases to make progress towards the goal). For details, see Sec. **??**.

3 Algorithmic approach—user-guided planning

To employ our planning framework, we assume that we are given a motion-planning algorithm $\mathcal{A}$ that is endowed with two non-standard procedures which are planner dependent. The first, `is_in_stagnation_region()`, identifies when it is in a *stagnation region*, namely when $\mathcal{A}$'s search does not progress towards the goal. The second, `update_user_guidance()`, incorporates (or removes) the user guidance provided to $\mathcal{A}$.

Equipped with these functions, we can describe our planning framework, detailed in Alg. 1. The framework runs as long as no solution is found (line 1). It runs the planner $\mathcal{A}$ (lines 2-3) as long as it continuously makes progress towards the goal (namely, it is not in a stagnation region). Once a stagnation region is identified, user guidance is invoked (line 4) and $\mathcal{A}$ is updated to make use of this guidance (line 5). It is then run while using the guidance as long as it is still in the stagnation region (lines 6-7). Once it escapes the stagnation region, $\mathcal{A}$ is updated to remove the guidance that was provided by the user (line 8).

4 User-guided planning via MHA*

We demonstrate our general planning framework described in Sec. 3 for the case where the motion-planning algorithm $\mathcal{A}$ is multi-heuristic A* (MHA*) [Aine *et al.*, 2016]. We assume that MHA* has a set of possibly inadmissible heuristics in addition to the anchor heuristic. We will refer to these heuristics as *baseline heuristics*. The user guidance will be used to generate *dynamic heuristics*. We start (Sec. 4.1-4.3) by describing how we answer the three questions we posed in Sec. 1. We then continue to detail how they are used in our planning framework. This implementation is slightly more complicated than that described in Alg. 1 and is detailed in Sec. 4.4.

4.1 Invoking user guidance (Q1)

The heuristic functions of search-based planning algorithms, such as MHA*, can be used to estimate in a principled manner when the planner is in a stagnation region (Alg 1, lines 2, 6). We suggest to identify when the planner is in a stagnation region as follows: Let $\mathcal{Q}$ be a priority queue ordered according to some heuristic function $h(\cdot)$, s_i be the node expanded from $\mathcal{Q}$ at the i'th iteration and ω_1, ω_2 be parameters such that $\omega_1 > \omega_2$. We define $\kappa_{\mathcal{Q}}(i, \omega) = \min_{i-\omega \leq j \leq i}\{h(s_j)\}$. Namely, $\kappa_{\mathcal{Q}}(i, \omega)$ denotes the minimal value attained by h in the past ω states.

Definition 1 *A heuristic h associated with a queue $\mathcal{Q}$ is defined to be in a stagnation region if* $\kappa_{\mathcal{Q}}(i, \omega_1) \geq \kappa_{\mathcal{Q}}(i - \omega_2, \omega_1 - \omega_2) - \varepsilon$.

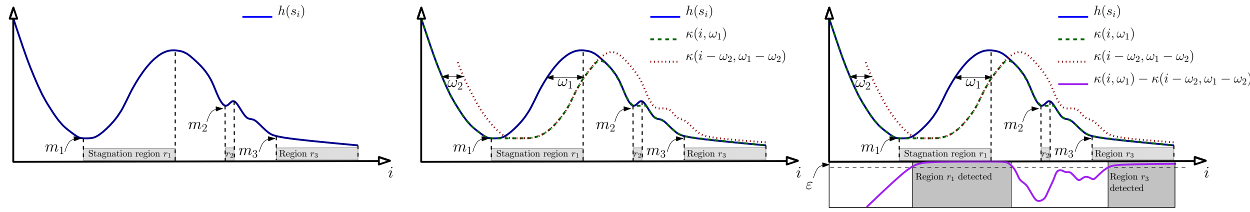
Namely, $\mathcal{Q}$ is in a stagnation region if looking at the previous ω_1 iterations, there was no reduction (by more than some threshold ε) in the minimum value of h in the last ω_2 states expanded from $\mathcal{Q}$.

Thus, our algorithm is given three additional parameters ω_1, ω_2 and ε and maintains for each queue $\mathcal{Q}$ the values $\kappa_{\mathcal{Q}}(i, \omega_1)$ and $\kappa_{\mathcal{Q}}(i - \omega_2, \omega_1 - \omega_2)$. Note that testing if a single queue is in a stagnation region takes $O(1)$ time. For a visualization, see Fig. 2.

4.2 Form of user guidance (Q2)

We chose to obtain user guidance (Alg 1, line 4) in the form of an intermediate configuration $\hat{q}$ that is used to guide the planner. We discuss alternative options in Sec. 6.

The framework includes a graphical user interface (GUI) (Fig. 3) capable of depicting the robot and the workspace. Once user guidance is invoked, a configuration in the stagnation region is obtained and the robot is placed in that configuration (as well as the start configuration and target region) in the GUI. This allows the user to intuitively try and understand where the planner faces difficulty and how to guide it out of the stagnation region. The user then suggests the guidance $\hat{q}$ by moving the robot's joints and end effectors. The tool runs a state validity checker in the background which restricts the user from providing invalid configurations, e.g. with respect to collision, joint limits and stability. Note that the user is *not* required to be familiar with the search algorithm.



(a) (b) (c)

Figure 2: Visualization of the way stagnation regions are detected. (a) The heuristic value $h(s_i)$ (solid blue) has three local minima, m_1, m_2 and m_3 followed by three stagnation regions (light grey). Local minima m_2 is very small while m_3 is not a local minimum per se, yet the progress made between consecutive steps is smaller than the predefined threshold ε. (b) The function $\kappa(i, \omega_1)$ (dashed green) returns the minimal value $h(s_i)$ attained over the past ω_1 iterations (referred to as the "recent history"). The function $\kappa(i - \omega_2, \omega_1 - \omega_2)$ (dotted red) returns the minimal value $h(s_i)$ attained over the beginning of the recent history. (c) The difference between the two functions (solid purple) indicates if there was significant progress (i.e. more than ε) made at the end of the recent horizon when compared to the beginning of the recent history. If not, then the planner detects a stagnation region (dark grey). Notice that the hysteresis parameters ω_1 and ω_2 (i) induce a lag from the time that the stagnation region starts until it is detected, (ii) allow to avoid detecting stagnation region r_2.

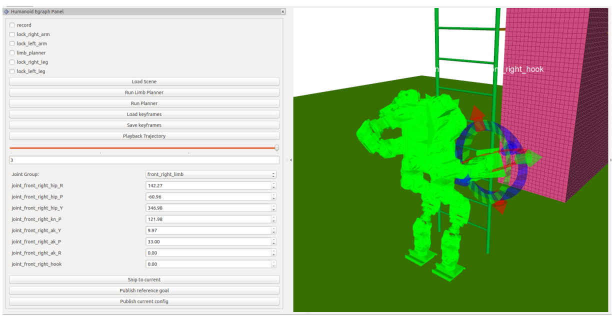


Figure 3: The graphical user interface used to provide guidance to the planner. The panel on the left hand side can be use to select different joint groups, move the joints around and pass the guidance to the planner. The interactive marker shown on the right hand side can be used to move the robot end effectors.

4.3 Using user guidance (Q3)

We assume that MHA* has at least one baseline heuristic h_{goal} which approximates the cost to reach the goal from every state. Furthermore, we assume that there exists a family of (possibly inadmissible) heuristic functions $\mathcal{H}$, such that for every configuration q, there exists a heuristic $h_q \in \mathcal{H}$ where $h_q(s)$ estimates the cost to reach q from state s.

Given user guidance in the form of a configuration $\hat{q}$, we dynamically generate a new heuristic

$$\hat{h}(s) = \begin{cases} h_{\hat{q}}(s) + h_{\text{goal}}(\hat{q}), & \text{if } \hat{q} \text{ is not an ancestor of } s, \\ h_{\bullet goal}(s), & \text{if } \hat{q} \text{ is an ancestor of } s. \end{cases}$$

Namely, $\hat{h}$ estimates the cost to reach the goal (via the term h_{goal}) by passing through $\hat{q}$ (via the term $h_{\hat{q}}$). If the state was reached by passing through $\hat{q}$, then the value of $\hat{h}$ is simply the estimation of the cost to reach the goal.

Equipped with the heuristic $\hat{h}$, we add a new queue to the MHA* algorithm prioritized using the heuristic $\hat{h}$. States expanded using this queue will be biased towards $\hat{q}$ (see also [Islam *et al.*, 2015] for more details on adding heuristics and queues dynamically to MHA* and [Narayanan *et al.*, 2015] for more details on dealing with calibrating the different values used by different heuristic functions). Note that in MHA*, nodes are shared between the different queues. Thus, once a state has been found that can be used to get the planner out of the stagnation region, it will be expanded by the other queues using their heuristics. Once this is detected the newly-added queue is removed.

We note that we can add a dynamic queue for every baseline heuristic if there are more than one. However, for simplicity, in this work we use a single baseline heuristic and add one dynamically generated queue when the user provides guidance.

4.4 User-guided MHA*

We are now ready to describe how we apply our general framework of user-guided planning to the case of MHA*. This differs slightly from the general approach described in Sec. 3 due to our ability to identify which queue is in stagnation and to detect if the configuration $\hat{q}$ (i.e., the guidance) was reached.

Specifically, if the baseline heuristic escaped a stagnation region but the configuration $\hat{q}$ was *not* reached, we suspend the dynamic queue but do not discard it. This is done to first try reusing the last guidance before asking for a new one. When the planner will detect that is in a stagnation region, it will first resume the suspended dynamic heuristic (if one exists).

If the baseline heuristic escaped a stagnation region and the configuration $\hat{q}$ was reached it will no longer be useful again and hence will be discarded. Finally, if the dynamic heuristic is in a stagnation region then it is discarded and the user will be queried for a new guidance.

For pseudo-code describing the algorithm, describing the implementation of each specific function of our general pseudocode (Alg. 1), see Alg. 2. For a visualization of the way the algorithm progresses, see Fig. 4. Note that although for the illustrated example, the search happens to pass through the guidance, in general it is not a constraint in the framework.

5 Evaluation

We evaluated the performance of our planning framework on a 34-DOF WAREC humanoid robot [Matsuzawa *et al.*,

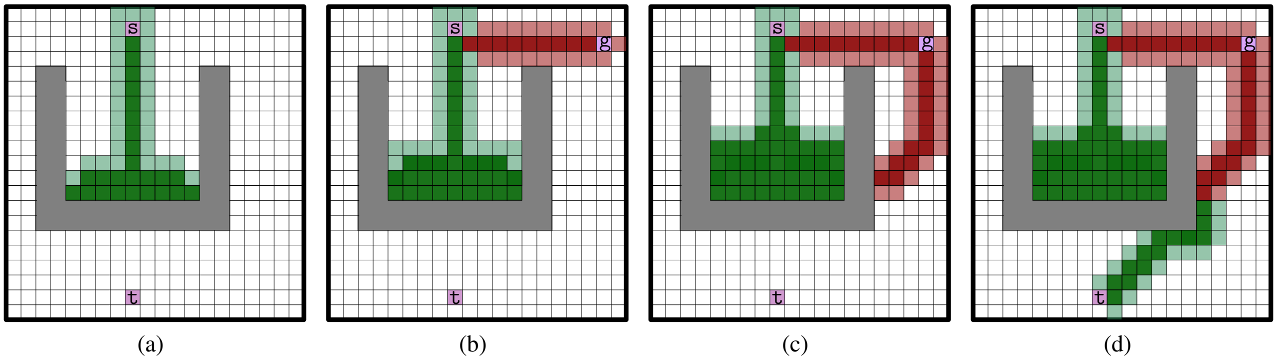


Figure 4: Algorithm progression. States popped from a priority queue are depicted using dark colors while states still in a queue are depicted by a light color. Start, target and user-guided states are depicted in purple with the letters s, t, g, respectively. In this example MHA* alternates between queues in a round-robin fashion. Furthermore, heuristics are inflated by a weight of $w = \infty$. Namely, the priority queues are ordered according to the heuristic-value of the states. (a) MHA* starts with a single baseline heuristic (green) which is the Euclidean distance to goal and a local minimum is identified. (b) User provides guidance and the algorithm automatically generates an additional heuristic (red) that drives the search towards the guidance (notice that the anchore heuristic continues to search the local minimum). (c) After passing through the guidance, the additional heuristic (red) drives the search towards the goal. (d) After the additional heuristic found states that are placed at the top of the priority queue of the anchor heuristic, the additional heuristic is deleted and the anchor heuristic drives the search towards the goal.

Algorithm 2 User-guided MHA*

```
function is_in_stagnation_region()
  if baseline heuristic not in stagnation then
    return true
  else
    return false

function get_user_guidance()
  if exists suspended dynamic heuristic then
    return suspended dynamic heuristic
  else
    get new user guidance and add dynamic heuristic
    return new dynamic heuristic

function remove_user_guidance()
  if dynamic heuristic is in local minimum then
    remove dynamic heuristic                ▷ guidance is not useful
  else                        ▷ dynamic heuristic is not in local minimum
    if states passed through guidance then
      discard dynamic heuristic             ▷ will not be useful in future
    else
      suspend dynamic heuristic             ▷ may be useful in future
```

2015] which is constrained to maintain static stability at all times. The robot has four symmetric limbs each having seven revolute joints. The other six dimensions come from the pose of the robot with respect to some global frame of reference. In one of the experiments (Sec. **??**), we employ the recently-introduced adaptive dimensionality framework [Gochev *et al.*, 2011; Gochev *et al.*, 2012; Gochev *et al.*, 2013]. Roughly speaking, it consists of two stages: an adaptive planning phase which attempts to plan in a low-dimensional space when possible and a tracking phase which plans in the high-dimensional space. Our user-guided planner is integrated within the high-dimensional planner. Specifically, the planner attempts to find a plan in the full-dimension configuration space of the robot in order to track the footstep plan generated in the planning phase.

5.1 Implementation details

We used a set of motion primitives which are short kinematically feasible motion sequences to generate the actions, or the successors, of a state during the search. These primitives included clockwise and counter-clockwise rotation of individual joints. Specifically, four different types of motion primitives are used: (i) primitives for the free limbs of the robot i.e. the limbs whose end effectors are not constrained by any contact surface, (ii) primitives to move the torso of the robot, (iii) primitives to latch onto the target end-effector poses and (iv) primitives to latch onto the user-provided configuration once it is within a small threshold distance to the state being expanded. Notice that motion primitives (ii) and (iii) require solving closed-chain inverse kinematics for the constrained limbs.

For each experiment we used a very small set of fairly generic heuristics. This is done to demonstrate that our approach allows to significantly reduce the engineering effort required in carefully crafting domain-specific heuristics. We conducted two experiments which largely differ in the nature of mobility and thus employ different baseline heuristic functions. The dynamic heuristic function in both the experiments is the Euclidean distance in the joint 34-DOF space. The parameter values that we used consistently in the two experiments are $\omega_1(200), \omega_2(50)$ and $\varepsilon(50)$. Finally in our final experiment, we show that while our approach uses several parameters, it is highly robust to the choice of parameters for our domain.

5.2 Mounting onto a ladder

The first task we considered was mounting onto a ladder while starting from a standing position (see Fig. 3). The initial state of the robot here is a full-dimensional configuration. The goal state is defined as 6-DOF poses for all the four end effectors (hands and feet) of the robot. These poses are predefined contact positions on the ladder rungs which ensure a reasonable degree of robot stability. This is a challenging problem as the planner is required to compute the order in which each of these target contacts has to be reached. Moreover, the targets need to be approached in a certain direction to establish the desired contacts while avoiding collisions. The inability of heuristic functions to model such task-specific constraints introduce stagnation regions in the search. The baseline heuristic function that we used in this experiment is computed as the summation of four six-dimensional euclidean distances between each of the four end effectors and their respective target poses.

Results demonstrating the effectiveness of our framework are depicted in Fig. 6(a) Specifically, we plot the heuristic value (a proxy to the algorithm's progress) as a function of the number of queue expansions. For the same setting, we ran our algorithm with and without user guidance. Namely, if a stagnation region was detected, we recorded the state of the planner and then continued to run it once without asking the user for guidance and once using our approach. This was done every time a stagnation region was detected. Results show that without guidance, the planner is not able to make any significant progress and the heuristic value does not change. On the other hand, when guidance is given, then the algorithm escapes the stagnation region and resumes to make progress towards the goal.

5.3 Bipedal locomotion

The second task we considered is bipedal locomotion of the robot while avoiding collisions with the obstacles in the environment as well as adhering to the physical constraints of the robot. The initial state of the robot again is the full dimensional 34-DOF configuration of the robot but the goal state in this experiment is represented just as a cylindrical region which the robot needs to enter (depicted in Fig. 1).

As briefly discussed earlier, for this experiment we use the adaptive planning framework. The planning phase generates the footstep plan first followed by the tracking phase which runs the full high-dimensional planner in order to track those footsteps. We integrate our approach into the latter. A single baseline heuristic is used in this experiment which assists the search with a general walking or stepping capability for the robot. For each step the footprint of the target footstep is visualized for the user to be able to better understand where the planner is stuck and how to provide the guidance. In relatively easier scenarios this heuristic would suffice in guiding the search. However if the search encounters harder situations such as those depicted in Fig. 5, then it would get trapped into local minima. Our approach invokes the user for guidance at four different instances where the baseline heuristic fails to make progress. In all of these instances the user can assist the planner by providing the guiding configuration after analysing the respective scenario. When the user provides the guidance, the dynamic heuristic guides the search to pass through that configuration until the baseline heuristic escapes the stagnation region.

We evaluated the performance by running the experiments on the two environments. The results are shown in table 1. In all the experiments the planner succeeds to reach the goal with the help of a user. The deviations in the experiments come from how good or bad the guidances were.

	Ladder	Staircase
Success Rate(percent)	100	100
Mean planning time(s)	113 ± 36	101 ± 17.5
Mean total time(s)	222 ± 55	246 ± 53.5
Mean state expansions	708 ± 189	1219 ± 242
Mean num of guidances	5.5 ± 0.85	8 ± 2
Mean time per guidance(s)	20 ± 3.6	18.5 ± 3

Table 1: Experimental results for the the staircase and ladder scenarios averaged over 10 trials

5.4 Robustness to algorithmic parameters

We introduced three parameters in section Sec. 4.1 i.e ω_1, ω_2 and ε which govern where the user guidance will be invoked. We can show that our approach is not very sensitive to the change in these parameters and thus they don't require fine tuning. The results presented in Fig. 6 are for some nominal values of the parameters.

For the experiments that we conducted, the results don't vary with changing ω_1 because the heuristic functions that we used don't show dips as illustrated in the hypothetical illustration in Fig. 2 but in general, such a trend can occur in an arbitrarily inadmissible heuristic. ε is a domain dependent parameter and its magnitude should be set relative to the scale of the corresponding heuristic function. For instance if we use n baseline heuristics, then the value of ε for each of these heuristics should be proportional to the scale of that heuristic. Having said that one can fix ε and tune the parameter ω_2 until you get the desired behaviour, as both of them are correlated and measure the rate of progress being made by a heuristic.

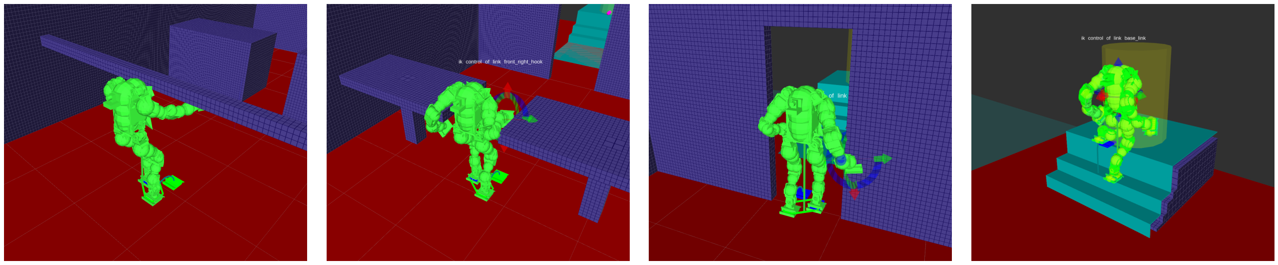
To show robustness to ω_2, we linearly swept its value over a window centred at the nominal value used for experiments and checked how many of the stagnation region detections matched with the nominal results. Fig. 7 shows the evaluation for the two experiments. For the values of ω_2 where the matching ratio is 1, the planner would have invoked the user exactly for the same stagnation regions as the ones for which the guidance was called using the nominal parameters.

6 Discussion and future work

6.1 Discussion

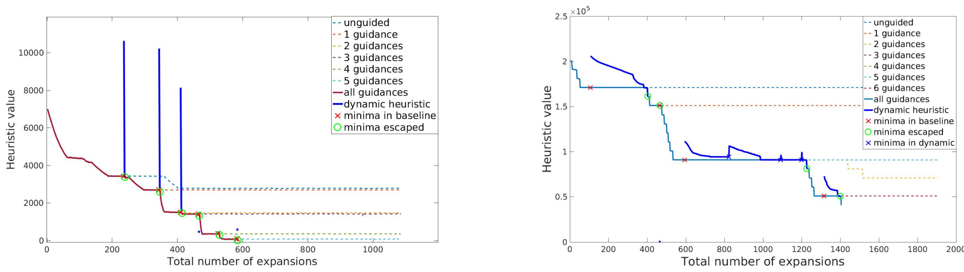
In Sec. 5 we demonstrated how using user guidance allows to solve highly-constrained motion-planning in high-dimensional spaces with only simple baseline heuristics. An alternative approach would be to add domain-dependent carefully-crafted heuristics [Vijayakumar, 2017] which allow to faster solve the same problems completely autonomously.

When a new type of problem is encountered (say, climbing a ladder, crawling on uneven terrain etc.) we can either design additional heuristics to tackle this problem or shift the burden to the user to provide guidance in an online fashion. If our problem requires planning in multiple, diverse types of domains this problem is accentuated—should we use a large ar-



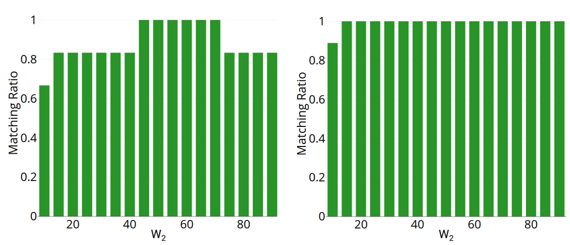
(a) (b) (c) (d)

Figure 5: Biped locomotion in challenging scenarios: (a) shows an instance where the robot has to squat or bend down to pass under a beam. (b) depicts a situation where the robot has to pass through two tables by may be lifting the elbows high above the tables. Progressing further, (c) is a situation where the robot has to squeeze through narrow doorway by tucking in its arms. (d) shows a scenario where the robot has to step onto a relatively high platform to reach the goal for which it has to lean a little bit more to it's right side to be able to lift the left foot further up.



(a) Mounting onto the ladder (b) Climbing up the staircase

Figure 6: Progress of MHA* with and without user guidance—heuristic values as a function of the number of queue expansions. When a stagnation region is detected (red cross) in the baseline heuristic (cyan), a dynamic heuristic is generated (blue) until the stagnation region is escaped (green circle) or until a new stagnation region is detected (blue cross) in the dynamic heuristic. Plots depicting the baseline heuristic values when only a limited amount of guidance is given demonstrate that without guidance, the planner remains stuck in the stagnation region (we note that as MHA* is complete, it will eventually escape all such regions).



(a) Mounting onto the ladder (b) Climbing up the staircase

Figure 7: The plot shows the ratio of the number stagnation region detections matched to the total number of regions detected by the nominal parameter values, for ω_2 values swept around the nominal value (50) by $\pm$ 40 expansions with a step of 5.

senal of heuristics that can address each domain or should we have a small number of baseline heuristics that will (roughly) address all domains and rely on user guidance when these baseline heuristics fail? There is no clear answer to this question and our approach simply offers a general alternative to existing approaches.

6.2 Future work

While providing promising initial results, our framework is far from being complete. We are interested in experimenting with alternative forms of user guidance such as providing a *constrained sub-manifold* of the configuration space which can be more informative for the planner instead of providing it with a single joint configuration and could potentially reduce the number of user invocations. Finally, once a guide is given, we want our planner to be able to *generalize* the guidance obtained to future stagnation regions that are similar in nature to the ones encountered (e.g., climbing up multiple stairs).

Finally we are interested in implemented our approach using sampling-based planners. Here, once the user will provide guidance, it can be used to bias the sampling procedure. An open challenge remains how to automatically detect that the planner is not progressing toward the goal.

References

[Aine *et al.*, 2016] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic A*. *IJRR*, 35(1-3):224–243, 2016.

[Argall *et al.*, 2009] Brenna Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[Blin *et al.*, 2016] Nassime Blin, Michel Taïx, Philippe Fillatreau, and Jean-Yves Fourquet. I-RRT-C: interactive motion planning with contact. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 4244–4249, 2016.

[Burns *et al.*, 2013] Ethan Burns, Wheeler Ruml, and Minh Binh Do. Heuristic search when time matters. *J. Artif. Intell. Res*, 47:697–740, 2013.

[Chestnutt *et al.*, 2005] Joel E. Chestnutt, Manfred Lau, German K. M. Cheung, James Kuffner, Jessica K. Hodgins, and Takeo Kanade. Footstep planning for the honda ASIMO humanoid. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 629–634, 2005.

[Choset *et al.*, 2005] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, June 2005.

[Cohen *et al.*, 2014] Benjamin J. Cohen, Sachin Chitta, and Maxim Likhachev. Single- and dual-arm motion planning with heuristic search. *IJRR*, 33(2):305–320, 2014.

[DeDonato *et al.*, 2017] Mathew DeDonato, Felipe Polido, Kevin Knoedler, Benzun PW Babu, Nandan Banerjee, Christoper P Bove, Xiongyi Cui, Ruixiang Du, Perry Franklin, Joshua P Graff, et al. Team WPI-CMU: Achieving reliable humanoid behavior in the DARPA robotics challenge. *J. Field Robotics*, 34(2):381–399, 2017.

[Denny *et al.*, 2014] Jory Denny, Read Sandström, Nicole Julian, and Nancy M. Amato. A region-based strategy for collaborative roadmap construction. In *WAFR*, pages 125–141, 2014.

[Dionne *et al.*, 2011] Austin J. Dionne, Jordan Tyler Thayer, and Wheeler Ruml. Deadline-aware search using on-line measures of behavior. In *Symposium on Combinatorial Search (SoCS)*, 2011.

[Flavigne *et al.*, 2009] David Flavigne, Michel Taïx, and Etienne Ferre. Interactive motion planning for assembly tasks. In *IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN)*, pages 430–435, 2009.

[Gochev *et al.*, 2011] Kalin Gochev, Benjamin J. Cohen, Jonathan Butzke, Alla Safonova, and Maxim Likhachev. Path planning with adaptive dimensionality. In *Symposium on Combinatorial Search (SoCS)*, 2011.

[Gochev *et al.*, 2012] Kalin Gochev, Alla Safonova, and Maxim Likhachev. Planning with adaptive dimensionality for mobile manipulation. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2944–2951, 2012.

[Gochev *et al.*, 2013] Kalin Gochev, Alla Safonova, and Maxim Likhachev. Incremental planning with adaptive dimensionality. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2013.

[Gray *et al.*, 2017] Steven Gray, Robert Chevalier, David Kotfis, Benjamin Caimano, Kenneth Chaney, Aron Rubin, and Todd Danko. An architecture for human-guided autonomy: Team TROOPER at the DARPA robotics challenge finals. *J. Field Robotics*, 34(5):852–873, 2017.

[Hauser *et al.*, 2008] Kris K. Hauser, Timothy Bretl, Jean-Claude Latombe, Kensuke Harada, and Brian Wilcox. Motion planning for legged robots on varied terrain. *IJRR*, 27(11-12):1325–1349, 2008.

[Hernández and Baier, 2012] Carlos Hernández and Jorge A Baier. Avoiding and escaping depressions in real-time heuristic search. *J. Artif. Intell. Res*, 43:523–570, 2012.

[Holladay and Srinivasa, 2016] Rachel M Holladay and Siddhartha S Srinivasa. Distance metrics and algorithms for task space path optimization. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 5533–5540, 2016.

[Ishida, 1992] Toru Ishida. Moving target search with intelligence. In *AAAI*, pages 525–532, 1992.

[Islam *et al.*, 2015] Fahad Islam, Venkatraman Narayanan, and Maxim Likhachev. Dynamic multi-heuristic A*. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2376–2382, 2015.

[Jr. *et al.*, 2002] James J. Kuffner Jr., Satoshi Kagami, Koichi Nishiwaki, Masayuki Inaba, and Hirochika Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

[Kaneko *et al.*, 2004] Kenji Kaneko, Fumio Kanehiro, Shuuji Kajita, Hirohisa Hirukawa, Toshikazu Kawasaki, Masaru Hirata, Kazuhiko Akachi, and Takakatsu Isozumi. Humanoid robot HRP-2. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1083–1090, 2004.

[Koenig and Sun, 2009] Sven Koenig and Xiaoxun Sun. Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems (AAMS)*, 18(3):313–341, 2009.

[Korf, 1990] Richard E Korf. Real-time heuristic search. 42(2-3):189–211, 1990.

[Kuffner *et al.*, 2001] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Footstep planning among obstacles for biped robots. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 500–505, 2001.

[LaValle, 2006] Steven M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[Marion *et al.*, 2017] Pat Marion, Maurice Fallon, Robin Deits, Andrés Valenzuela, Claudia Pérez D'Arpino, Greg Izatt, Lucas Manuelli, Matt Antone, Hongkai Dai, Twan Koolen, et al. Director: A user interface designed for robot operation with shared autonomy. *J. Field Robotics*, 34(2):262–280, 2017.

[Matsuzawa *et al.*, 2015] Takashi Matsuzawa, Kenji Hashimoto, Xiao Sun, Kazuhiro Uryu, Ayanori Koizumi, Shinya Hamamoto, Tomotaka Teramachi, and Atsuo Takanishi. Development of disaster response robot with commonly structured limbs and experiment in climbing vertical ladder. In *Int. Conf. on Advanced Mechatronics (ICAM)*, pages 142–143, 2015.

[McGill *et al.*, 2017] Stephen G McGill, Seung-Joon Yi, Hak Yi, Min Sung Ahn, Sanghyun Cho, Kevin Liu, Daniel Sun, Bhoram Lee, Heejin Jeong, Jinwook Huh, et al. Team THOR's entry in the DARPA Robotics Challenge Finals 2015. *J. Field Robotics*, 4(34):775–801, 2017.

[Mehta *et al.*, 2015] Siddhartha S Mehta, Chau Ton, Michael J McCourt, Z Kan, Emily A Doucette, and W Curtis. Human-assisted RRT for path planning in urban environments. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 941–946, 2015.

[Narayanan *et al.*, 2015] Venkatraman Narayanan, Sandip Aine, and Maxim Likhachev. Improved multi-heuristic A* for searching with uncalibrated heuristics. In *Symposium on Combinatorial Search (SoCS)*, pages 78–86, 2015.

[Perrin *et al.*, 2012] Nicolas Perrin, Olivier Stasse, Leo Baudouin, Florent Lamiraux, and Eiichi Yoshida. Fast humanoid robot collision-free footstep planning using swept volume approximations. *IEEE Trans. Robotics*, 28(2):427–439, 2012.

[Phillips *et al.*, 2015] Mike Phillips, Venkatraman Narayanan, Sandip Aine, and Maxim Likhachev. Efficient search with an ensemble of heuristics. In *International Joint Conference on Artificial Intelligence, (IJCAI)*, 2015.

[Phillips *et al.*, 2016] Mike Phillips, Victor Hwang, Sachin Chitta, and Maxim Likhachev. Learning to plan for constrained manipulation from demonstrations. *Autonomous Robots*, 40(1):109–124, 2016.

[Pohl, 1970] Ira Pohl. First results on the effect of error in heuristic search. *Machine Intelligence*, 5:219–236, 1970.

[Ranganeni *et al.*, 2017] Vinitha Ranganeni, Oren Salzman, and Maxim Likhachev. Effective footstep planning for humanoids using homotopy-class guidance. *CoRR*, abs/1712.00531, 2017.

[Schulman *et al.*, 2016] John Schulman, Jonathan Ho, Cameron Lee, and Pieter Abbeel. Learning from demonstrations through the use of non-rigid registration. In *Robotics Research*, pages 339–354. 2016.

[Sharir, 2004] Micha Sharir. Algorithmic motion planning. In *Handbook of Discrete and Computational Geometry, Second Edition.*, pages 1037–1064. 2004.

[Spenko *et al.*, 2018] Matthew Spenko, Stephen Buerger, and Karl Iagnemma. *The Darpa Robotics Challenge Finals: Humanoid Robots To The Rescue*, volume 121 of *Springer Tracts in Advanced Robotics*. Springer, 2018.

[Taïx *et al.*, 2012] Michel Taïx, David Flavigné, and Etienne Ferré. Human interaction with motion planning algorithm. *J. Intell. Robot. Syst.*, 67(3):285–306, 2012.

[Vats *et al.*, 2017] Shivam Vats, Venkatraman Narayanan, and Maxim Likhachevand. Learning to avoid local minima in planning for static environments. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2017.

[Vijayakumar, 2017] Karthik Vijayakumar. A multi-heuristic framework for humanoid planning. Master's thesis, Robotics Institute, Carnegie Mellon University, 2017.

[Xia *et al.*, 2009] Zeyang Xia, Guodong Chen, Jing Xiong, Qunfei Zhao, and Ken Chen. A random sampling-based approach to goal-directed footstep planning for humanoid robots. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 168–173, 2009.

[Yan *et al.*, 2015] Yu Yan, Emilie Poirson, and Fouad Bennis. An interactive motion planning framework that can learn from experience. *Computer-Aided Design*, 59:23–38, 2015.

[Ye and Alterovitz, 2017] Gu Ye and Ron Alterovitz. Demonstration-guided motion planning. In *Robotics Research*, pages 291–307. 2017.

[Zucker *et al.*, 2015] Matt Zucker, Sungmoon Joo, Michael X Grey, Christopher Rasmussen, Eric Huang, Michael Stilman, and Aaron Bobick. A general-purpose system for teleoperation of the DRC-HUBO humanoid robot. *J. Field Robotics*, 32(3):336–351, 2015.