International Journal of Humanoid Robotics

A Tiered Planning Strategy for Biped Navigation

JOEL CHESTNUTT [1]
JAMES J. KUFFNER [1,2]

[1] *Robotics Institute, Carnegie Mellon University,*
5000 Forbes Ave. Pittsburgh, Pennsylvania 15213, USA
chestnutt@ri.cmu.edu, kuffner@cs.cmu.edu

[2] *Digital Human Research Center,*
National Institute of Advanced Industrial Science and Technology (AIST),
2-41-6 Aomi, Koto-ku, Tokyo, Japan 135-0064

This paper presents a three-tiered planner for biped navigation over large distances through complex environments where conventional 2D planning algorithms designed for wheeled robots fail to find a solution. The lowest tier is a footstep planner which can plan sequences of footholds for navigating through obstacles or over rough terrain. The second tier is a mobile robot planner, which plans outward from the goal to the robot's initial state, building a heuristic to aid the lowest level planner in directing the footholds toward a likely path. The highest tier chooses a long-term path to follow, ignoring the details of how it will be implemented, and directs the lower levels during execution to provide footstep sequences for the robot and notification of when re-planning is necessary. Results are demonstrated with simulated environments and execution.

1. Introduction

One current area of research involves the design of algorithms to compute robust goal-directed navigation strategies for biped humanoid robots operating in complex environments. For indoor environments designed for humans, this includes dealing with furniture, walls, stairs, doors, and previously unknown obstacles on the floor. For outdoor environments, this includes the ability to navigate on rough terrain and uneven surfaces. Because legged robots have the ability to step over and onto obstacles in their path, they are uniquely suited to overcoming these difficulties. However, existing navigation planning methods designed for wheeled mobile robots fail to consider these additional capabilities.

A biped navigation planner has been developed [2] which plans individual footsteps. By describing the capabilities of the biped as a set of possible footsteps and using a set of heuristics to quickly validate footstep locations in complex terrain, optimal paths can be found through difficult environments which will take advantage of the biped's legged abilities. Figure 1 shows the results of this planner an example terrain.

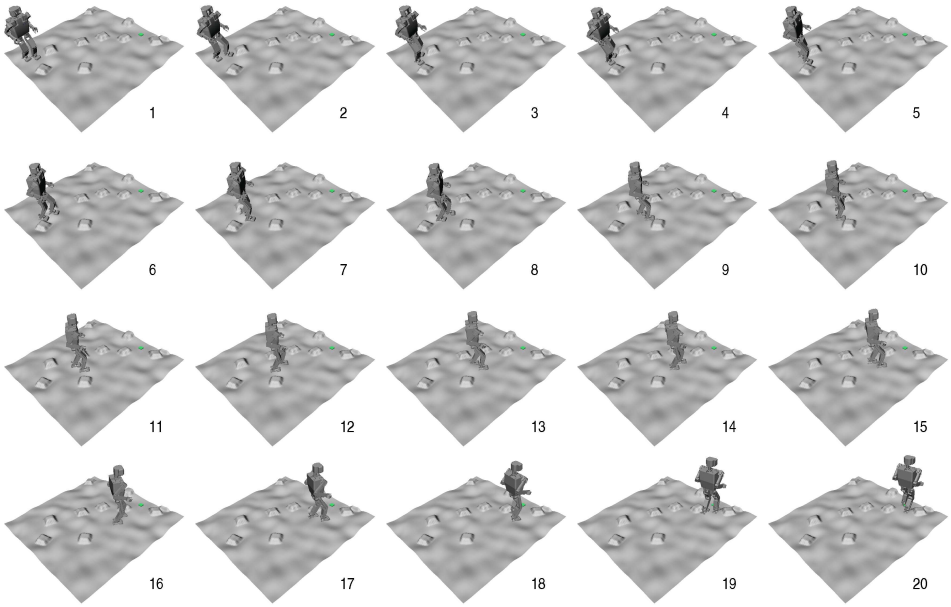


Fig. 1. The footstep planner providing a footstep path over a small but complex environment.

One drawback of this approach is that large terrains take a significant amount of memory and computation time to find solutions. In addition, plans are formulated based on a model of what the terrain looks like for the entire trip to its goal. The current implementation of this approach on physical robots limits the planning time to one or two step cycles and uses the best partial path computed in that time [2].

The idea motivating the work in this paper is that walking through a building, town, or forest type terrain does not require planning every single footstep in advance, only a rough sketch of the path to take. Toward this end, a high-level path planner is used to provide that rough sketch. This plan is then filled out during execution with the footstep planner. A mobile robot planner is used as a heuristic to direct the footstep planner, so that even in a time-limited partial plan, the footstep path will likely take the robot in the "right" direction. The high-level planner thus knows its way around an area, but does not need to know the exact details of the terrain it will be traversing. In most cases, that level of information will not be available at the start of travel. Due to this lack of information, the path the high-level planner finds may not be executable and must be monitored during execution. If the footstep planner cannot find a sequence of footsteps to reach the subgoals specified by the high-level planner, the high-level planner must re-plan and provide another route.

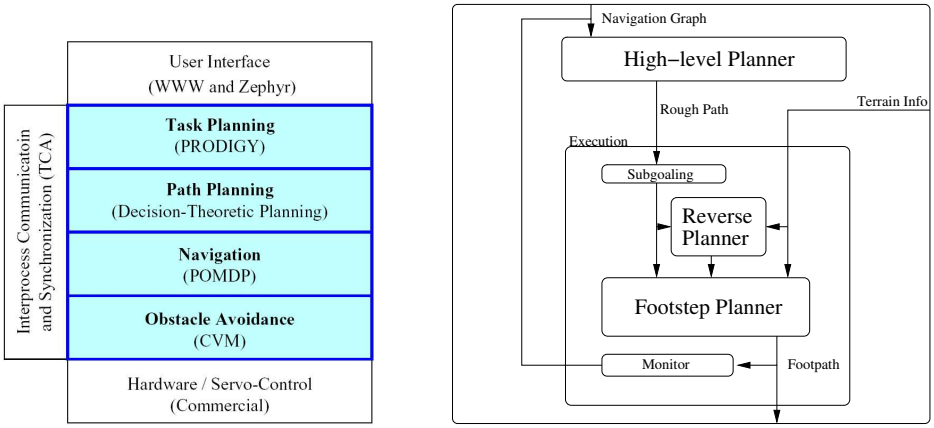
2. Background

Global path planning and obstacle avoidance strategies for mobile robots and manipulators has a large and extensive history in the robotics literature [4, 6]. Global navigation strategies for mobile robots can usually be obtained by searching for a collision-free path in a 2D environment. Because of the low-dimensionality of the search space, very efficient and complete (or resolution-complete) algorithms can be employed [17]. These techniques can be partially applied to biped humanoid robots. Conservative global navigation strategies can be obtained by choosing an appropriate bounding volume (e.g. a cylinder), and designing locomotion gaits for following navigation trajectories computed by a 2D path planner [5, 12]. However, this always forces the robot to circumnavigate obstacles. In contrast, legged robots such as biped humanoids have the unique ability to traverse obstacles by stepping over or upon them. Motion planning for spider-like legged robots with point feet has also been studied [1], where the planner finds foot placements for statically stable motion which must keep the robot's center of mass above the convex hull of its feet.

Since reliable, walking biped robots have been developed only recently, much less research attention has been focused on developing complete global navigation strategies for biped legged robots. Most research has focused on pre-generating stable walking trajectories [3, 8, 20], or on dynamic balance and control [13, 19]. Recently, techniques have been developed to generate stable walking trajectories online [9, 10], though these results do not account for obstacles.

Some recent humanoids have begun using autonomous path planning and navigation. Sony's QRIO can use its video system to recognize obstacles, and uses a mobile robot planner to navigate a path around them. The robot Johnnie from the Technical University of Munich [7] can modify a walking motion reactively, modifying step length or locomotion direction, to step over or around obstacles it senses. The H7 robot at the University of Tokyo used online footstep planning for navigation for moving goals and moving obstacles [2].

The tiered approach presented here has many similarities with the system used in the Xavier Project [16]. The planning system used for Xavier, shown in Figure 2(a), also has a layered structure. The main difference in Xavier's system is the low-level navigation control. Xavier used a POMDP control for navigation, with obstacle avoidance running beneath it. Unlike mobile robots, the steps bipeds take are made up of discrete points in the workspace, and thus do not require a continuous path. In addition, bipeds must be much more careful about their balance on the ground. In a cluttered area, a biped may be able to travel where a wheeled robot cannot, but finding the correct places to step requires some extra effort. Another difference is that Xavier's POMDP navigation system handled the robot's localization in the environment. Localization is not discussed in this paper for simplicity. To use this system on a real robot, some form of localization would be needed during execution to keep the "rough sketch" path synchronized with the corresponding real world locations. Finally, Xavier had a task-level planner for scheduling tasks



(a) Xavier's control structure

(b) The framework described in this paper.

Fig. 2. Control structures

for the navigation system. While such a planner is beyond the scope of this paper, any task scheduler which could provide destinations for the high-level planner could easily be integrated.

3. Algorithm

The structure of the tiered planner presented here is shown in Figure 2(b). The high-level planner takes as input the initial and goal locations, and a representation of the environment. Its output is a path, which is used by the execution component as a guide to direct the low level planning. The execution component choses a subgoal from the path, acquires terrain info from the sensors, and sends that information as inputs to the low-level planning.

The low-level planning is performed once each step cycle. Using the terrain data, represented as a 2.5D height map, it plans backwards from the goal toward the current state of the robot. This provides a heuristic for the footstep planning. Because the planning calculation must complete within one step cycle, its running time is limited, and the best partial plan is used if it does not find a full path in the allotted time. The mobile robot planner heuristic is not admissible for the A* search the footstep planner performs, but as it is generally an informed heuristic, it means that partial plans tend to be in the desired direction.

Once this footstep plan has been generated, the robot can begin to execute it. During the next step's planning, the unused portions of the current plan are used to seed the search queue, so the remainder of the previous effort is not completely wasted. Because the footstep path is recalculated at each step, the execution can

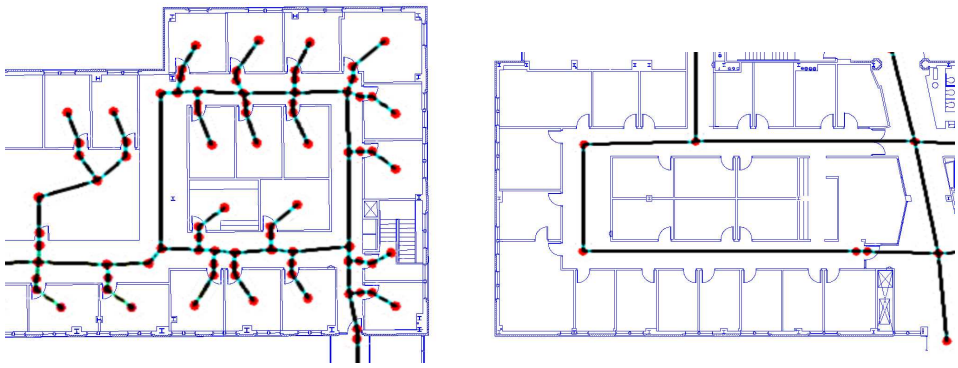


Fig. 3. Fine and coarse graphs used to generate the high level plan

quickly adapt to new terrain information or goal locations.

Finally, the progress of the robot is monitored. Should the low-level planning fail to find a path, the execution can be halted and the high-level planner re-invoked to find a new route to the goal.

3.1. *High-level Planner*

The high-level planner's function is to provide an approximate path over long distances for the robot to follow. The high-level planner implemented for this paper was simple, but servicable. The environment is represented as a graph, with each vertex representing a point in space. In the examples presented later on, they were used to represent intersections, doors, and rooms for a building's interior. This graph behaves as a floorplan of the building, allowing the robot to know its way around a particular location. Although the graphs used in the examples in this paper were manually created, these graphs can be constructed automatically from a floorplan-type description of the environment through visibility graph or voronoi techniques [14]. Topological graphs for high-level planning can also be learned through exploration of the environment [15, 18].

To find a high-level path, the start and goal locations are connected to their nearest neighbors in the graph, and then a graph search is performed to generate an optimal list of vertices to take the robot to the goal. These vertices can be interpreted as a list of goals for the low-level planning. Figure 3 shows the graph used to plan around the building shown. One corner of the example environment was covered very finely, taking into account door placement, intersections, and rooms. The opposite corner of the building was covered much more sparsely, leaving large areas fairly far from any vertex in the graph.

3.2. *Subgoaling*

Choosing the correct subgoal is the challenging part of combining these planners. Simply marching from one vertex in the path to the next will not suffice, as there is no guarantee that the vertices are not on obstacles, or placed right before obstacles, constraining the robot's path through an undesireable location.

One approach is to use the vertices as subgoals directly, but switch from one vertex to the next well before it is actually reached by the robot, thus not constraining the robot to pass directly through it or close to it. This is the approach used to generate the results in Section 4. The drawback to this approach happens when the vertices are very far apart. In that case, the reverse planning phase may have a large area to cover, which can leave little time in the one-step planning cycle for the footstep planner to generate a useful partial path. In addition, a distant subgoal can be outside the range of the robot's sensor capabilities, limiting the usefulness of the reverse planning as a heuristic.

Another method is to constantly update the subgoal to be some distance in front of the robot, along the line between subgoals. This distance can be chosen so that the robot is planning within the realm its sensors can detect, and large gaps between vertices will not slow the footstep planning down. The downside to this approach is the planning horizon introduced by the limited planning distance. The subgoal could lead the robot into a local minima, which will then need to be backtracked from when the obstacle falls within the planning distance.

3.3. *Mid-level Planner*

Once we have a subgoal to plan towards, the mid- and low-level planners generate the actual footsteps in response to the terrain. The terrain is represented as a grid of cells, each cell containing a height value. Together, these cells create a 2.5D height map describing the shape of the terrain. This representation is constructed from sensor data and is used for both the mid- and low-level planners.

The purpose of the mid-level planner is not to provide a more detailed path to the low-level planner, or to provide a reference path along which to fill in footsteps. Instead, the information generated from the planning process is saved and used by the low-level planner as a heuristic estimating the remaining distance to travel.

A mobile robot planner that plans outward from the goal state to the initial state provides a useful estimate of remaining cost, with the results stored in a grid which discretizes the workspace. During the footstep planning, the remaining cost can then be found in constant time. This heuristic takes more information about the environment into account than a Euclidean distance metric, but has several disadvantages besides the extra preprocessing time. Mobile robot planners look for a continuous path through the configuration space or workspace that connects the initial and goal states. Because the biped has the ability to step over obstacles, it does not require a continuous path through the workspace. The result of this difference is that the mobile robot planner can severely misjudge the cost of a

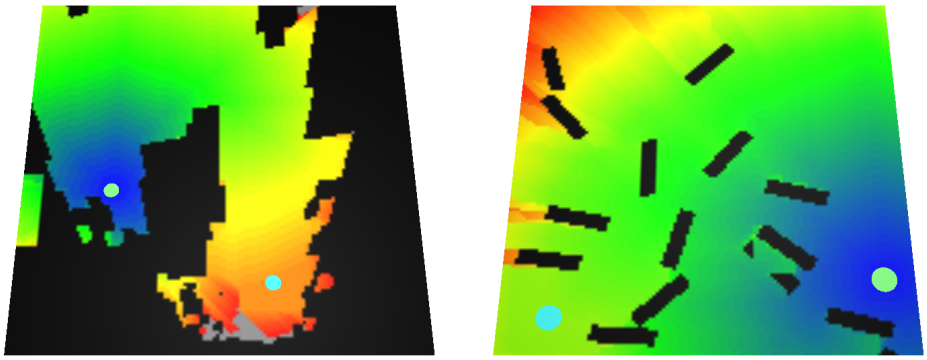


Fig. 4. The reverse planner as a heuristic. Blue is low cost, red is high. For the left figure, the goal is on the left side. For the right figure, the goal is in the lower left.

location. In an evironment with a long, low, thin obstacle, the mobile robot planner will provide lower cost to areas which send the biped the long way around instead of stepping over the obstacle, resulting in an overestimate. Also, it can underestimate when finding a path that one foot can fit though, but where there are not actually alternating footholds the robot can step on. In general, the time complexity of A* search (used in the low-level planner) is an exponential function of the error in the heuristic used [11]. So while in many environments, this heuristic performs much better than Euclidean distance, the worst case can be an arbitrarily large overestimate. Some examples of the heuristic that is generated are shown in Figure 4.

3.4. *Low-level Planner*

The footstep planner uses a fixed sample of the possible steps the robot is capable of making. Each step is described by a relative location, an allowable height change, a cost, and an obstacle clearance. The planner then uses an A* search to find an optimal sequence of footsteps from this base set. The planner evaluates three costs for each footstep location. First is the *location cost*, which evaluates the transition's destination as a potential foothold. This cost uses a variety of metrics to quickly compute how viable a location is for stepping onto. Second is a *step cost*, which computes the cost of reaching the footstep by making the chosen transition from the current state. This cost includes the transition's associated cost, a penalty for height changes, as well as an obstacle clearance check of the terrain between the foot's last position an the new foothold. Finally, the third cost is a heuristic which estimates the *remaining cost* to reach the goal state. These costs, used by the A* search to determine promising nodes to expand, are described in more detail in our

previous work [2].

3.5. *Execution Monitoring*

The footstep planner is complete up to the chosen sampling of footsteps used in the search. However, to fully exhaust all possibilities would take more time than the allotted one step-cycle. To determine whether an area is impassible to the robot, the monitoring measures its progress over its last several steps. When the robot gets into a position where the footstep planner cannot find a path to the goal, the one step-cycle planning steps back and forth in a small area, not able to find anything better in the limited time. When the monitor detects that the robot is no longer making progress over several steps, it concludes that the planning has failed.

At this point, it can pause and try a longer search time to attempt to find a difficult sequence. Alternately (or if the longer planning time still fails), the monitoring can then remove the edge in the high-level planning graph which the robot was traversing, and run the high-level planner with the new graph and new initial position. This will provide an alternate route which may not be blocked.

4. Results

For all of the results shown here, the step cycle was set to one second. All of the paths shown from the tiered planner in these figures were generated by one-second bursts of planning for each footstep. The results were generated on a 1.8 GHz Pentium 4 computer with 1 GB of memory running RedHat Linux 9.0.

4.1. *Bottom Tier Planning Results*

The lowest-level planner successfully finds footstep sequences through complex terrains, which can require the robot to step on or over obstacles, to walk up or down stairs, and to find safe footing on uneven ground. An animation the the robot H7 executing a footstep sequence found by the low-level planner is shown in Figure 1. Given detailed data about a large environment, this bottom tier can find the optimal footstep sequence, but the processing time increases from less than a second for examples such as Figure 1, to several minutes for terrain similar to that shown in Figure 5.

4.2. *Middle Tier Planning Results*

When the middle tier is layered on top of the low-level footstep planning, two advantages become immediately apparent. First, the planning process is sped up in a large variety of terrains. Local minima no longer incur large processing penalties, because the middle tier heuristic guides the path around them. The second benefit is the increased usefulness of partial paths. When a partial path is returned due to time limitations, a more informed heuristic makes the partial path more likely to be the beginning of the optimal path.

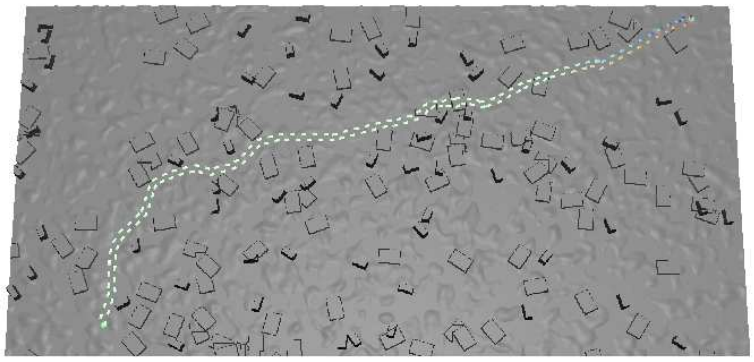


Fig. 5. Planning over longer distances using the low-level planner. While the low-level planner is capable of generating plans with hundreds of steps, the time involved in doing so precludes its use for a real-time system.

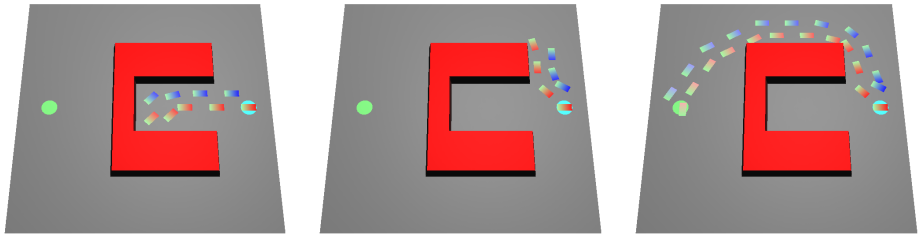


Fig. 6. *Left:* Time-limited low-level planning, *Center:* Time-limited planning with the mobile robot heuristic, *Right:* The complete path to the goal.

Figure 6 shows these advantages in the presence of a local minimum. In the left two examples, the total planning time was limited to 400ms (the mobile robot planner finishes approximately 350ms into the planning process). Even with such a short time to generate footsteps, the path found when using the mobile robot planner heuristic is a useful path to begin executing. Given enough time, both planners return the path shown on the right in Figure 6. However, the bottom tier takes 117 seconds to find this path by itself, but when combined with the middle tier, it finds that path in only 560 milliseconds.

4.3. *Three Tiered Planning Results*

By recording the path taken by the simulated robot during execution, we can compare it to the optimal path generated by the footstep planner which has full knowledge of the environment in advance and unlimited time and memory for planning.

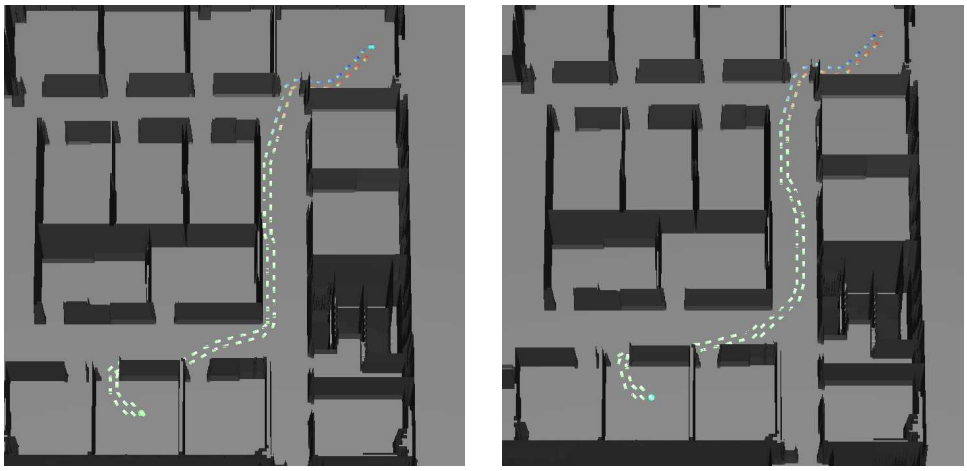


Fig. 7. *Left:* The footstep sequence planned from the start to the goal using the bottom two planning tiers. *Right:* The results of a simulated run using all three tiers, with plans made at each step to subgoals.

Figure 7 shows the optimal path and simulated execution runs for traveling between two offices. The paths are mostly similar, the only significant difference is that the simulated run stays more to the center of the hallway. The simulated run was guided by vertices in the center of the halls, while the optimal planner was minimizing distance. Figure 7 also shows that the closely spaced vertices work well and result in a path that does not significantly differ from the optimal. The paths generated in the presence of sparse high-level data can closely resemble an optimal path as well, as shown in Figure 8. While the nodes are placed far apart, and none of them enter the rooms, the footstep planner can still find a sequence between them. Sparse vertices cause suboptimal solutions when the wrong vertex is chosen to start or end from. In Figure 8, the bottom result shows a suboptimal path due to the high-level planner starting off in the wrong direction. With more intelligent choice of start and end connections in the graph, very loosely defined floorplans can be used to navigate through large environments without significant loss of optimality.

With the simulated runs, we can attempt much larger distances than are feasible for the pre-planned optimal trajectories. Figure 9 shows the results of walking from one corner of a building to the other with obstacles scattered along the way. The obstacles in halls will not have any effect on the high-level plan, as it has no knowledge of those details, but will affect the executed path. The robot must step around and onto these obstacles to proceed.

Finally, Figure 10 shows the results when one of the corridors the high-level plan chose was blocked. The footstep planner was unable to find a way past the blockage. The execution monitor detected that the robot was not making progress, so it replanned and used an alternate route to the goal. Discovering a blockage

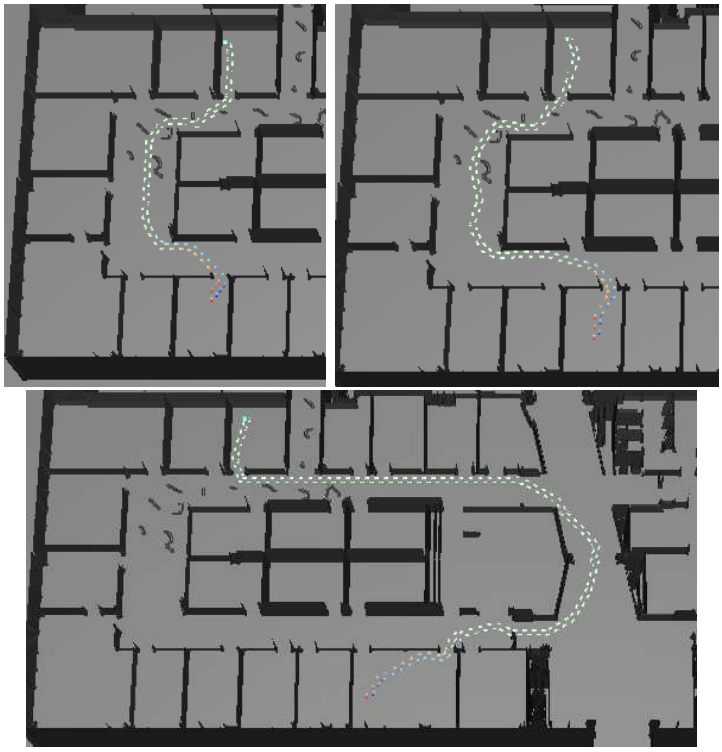


Fig. 8. Simulation runs over the portion of the map with sparse high-level planning data. The path significantly deviates from optimal only when the wrong node is chosen to start or end from.

can cause the planner to backtrack along previously reached subgoals, as shown in Figure 11. In this example, the robot backtracks twice along the path it came when discovering blockages before finding an unblocked path around to the goal.

5. Discussion

This combination of high and low level planners allows for much larger distances to be traversed, using only step-cycle planning times, and still providing the same safe footing available for smaller terrains. However, it is easy to see that optimality has been lost. Figures 10 and 11 show clearly how the resulting path is not optimal in some situations. This loss of optimality is acceptable when taking into account the fact the the information to generate an optimal footstep sequence for the whole trip is in practice not available at the start, and the required processing to find the optimal sequence can be prohibitive.

The examples shown involved the interior of a building, where paths are already fairly constrained, and choosing the vertices and edges for the high-level graph is a simple matter. This graph-based approach works well in environments where the available paths are constrained, such as building interiors, city sidewalks and

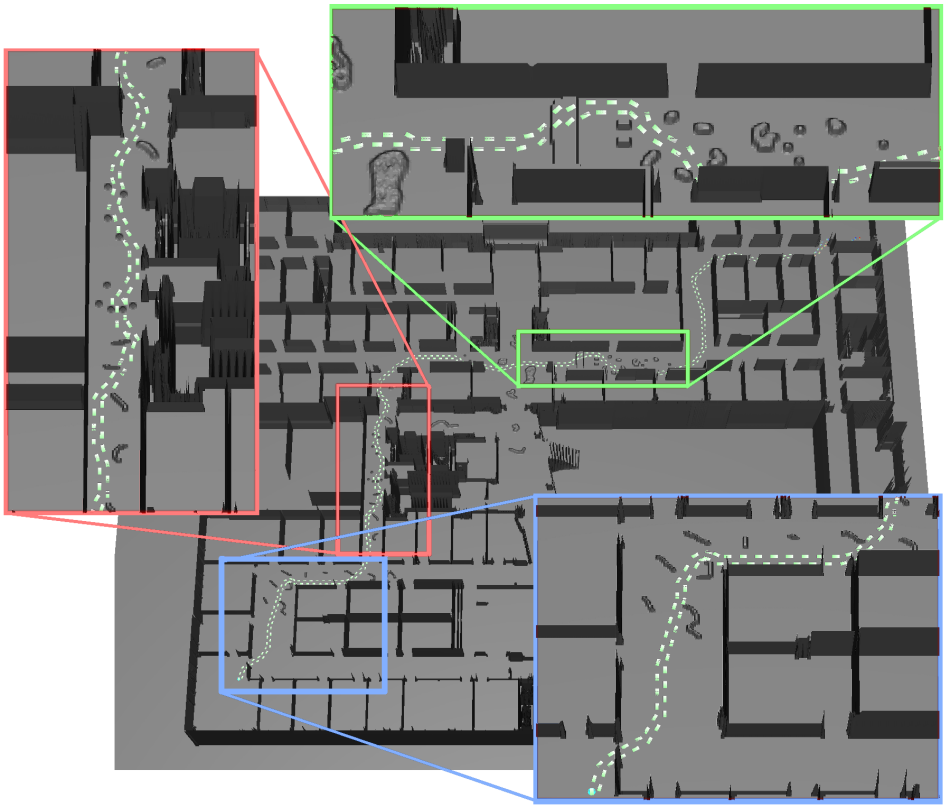


Fig. 9. A simulation run with lots of obstacles scattered along its desired path. The high-level path completely disregards the presence of any of the obstacles, leaving the middle and bottom tiers to find a way past them. Insets show details of the path used during execution.

crosswalks, or forest paths. However, this approach may not be appropriate for open, unconstrained spaces where the problems of vertices placed far apart become more apparent.

The performance of this graph-based reqresentation was satisfactory, even when the available graph data was sparse. The better the performance of the middle on bottom tiers of the planning, the more freedom there is in the density of high-level vertices. This is hopeful for the automatic generation of planning graphs for real environments, as the waypoints do not need to be very precise. In fact, the only observed errors due to sparse high-level graph information occured in linking the start and goal locations into the graph.

The use of a mobile robot planner as a heuristic can significantly speed up the planning process, but *only insofar as the optimal path can be followed by a traditional mobile robot*. As a result, the more an environment requires the biped's capabilities to step over or onto obstacles, the less informed this heuristic will be.

Fig. 10. A simulation run with a desired hallway blocked, requiring replanning from the high-level planner.

The detection of an impassable location during execution is based on the idea of forward progress. If the robot stops making progress, the execution monitoring assumes a blockages and reacts accordingly. However, this form of detection knows nothing about *why* no forward progress was made. It cannot distinguish between a temporary blockage, a permanent blockage, or some obstruction it could deal with itself (e.g. a group of people that the robot could ask to move, or a cart or box that could be pushed aside). This limitation means that the long term implications of removing an edge in the high-level planning graph cannot be properly dealt with. The system determines that an edge is impassable *at the moment*, but needs to identify the reason behind the impassibility to make an informed decision about whether to use that edge in the future or not.

Acknowledgements

This research was partially supported by NSF grants ECS-0325383, ECS-0326095, and ANI-0224419.

Fig. 11. A simulation run with mutliple desired hallways blocked, requiring backtracking to previously reached subgoals.

References

1. Jean-Daniel Boissonnat, Olivier Devillers, and Sylvain Lazard. Motion planning of legged robots. In *The Workshop on the Algorithmic Foundations of Robotics*, 1994.
2. J. Chestnutt, J.J. Kuffner, K. Nishiwaki, and S. Kagami. Planning biped navigation strategies in complex environments. In *Proc. IEEE Int. Conf. on Humanoid Robotics (Humanoids'03)*, Munich, Germany, October 2003.
3. K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'98)*, pages 1321–1326, May 1998.
4. Y. K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Trans. Robot. & Autom.*, 8(1):23–32, February 1992.
5. J.J. Kuffner. Goal-directed navigation for animated characters using real-time path planning and control. In *Proc. CAPTECH '98 : Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, pages 171–186, 1998.
6. J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
7. Klaus Löffler and Michael Genger. Sensors and control concept of walking 'Johnnie'. *International Journal of Robotics Research*, 22(3-4):229–240, 2003.
8. K. Nagasaka, M. Inaba, and H. Inoue. Walking pattern generation for a humanoid robot based on optimal gradient method. In *Proc. IEEE Int. Conf. on Systems, Man,*

and Cybernetics, 1999.

9. K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02)*, pages 96–101, 2002.
10. K. Nishiwaki, T. Sugihara, S. KAGAMI, M. Inaba, and H. Inoue. Online mixture and connection of basic motions for humanoid walking control by footprint specification. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'01)*, Seoul, Korea, May 2001.
11. J. Pearl. *Heuristics*. Addison-Wesley, Reading, Ma., 1984.
12. J. Pettre, Jean-Paul Laumond, and Thierry Simeon. A 2-stages locomotion planner for digital actors. In *Proc. SIGGRAPH Symp. on Computer Animation*, 2003.
13. J. Pratt and G. Pratt. Exploiting natural dynamics in the control of a 3d bipedal walking simulation. In *In Proc. of Int. Conf. on Climbing and Walking Robots (CLAWAR99)*, September 1999.
14. Vachirasuk Setalaphruk, Takashi Uneno, Yasuyuki Kono, and Masatsugu Kidode. Topological map generation from simplified map for mobile robot navigation. In *Conference of Japanese Society for Arti cial Intelligence,*, 2002.
15. Hagit Shatkay and Leslie Pack Kaelbling. Learning topological maps with weak local odometric information. In *IJCAI (2)*, pages 920–929, 1997.
16. Reid Simmons, R. Goodwin, K. Haigh, S. Koenig, Joseph O'Sullivan, and Maria Manuela Veloso. Xavier: Experience with a layered robot architecture. In *Agents '97*, 1997.
17. A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'94)*, pages 3310–3317, 1994.
18. S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.
19. M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic. *Biped Locomotion: Dynamics, Stability, Control, and Applications*. Springer-Verlag, Berlin, 1990.
20. J. Yamaguchi, S. Inoue, D. Nishino, and A. Takanishi. Development of a bipedal humanoid robot having antagonistic driven joints and three dof trunk. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98)*, pages 96–101, 1998.