

Planning Biped Navigation Strategies in Complex Environments

Joel Chestnutt¹, James Kuffner^{1,3}, Koichi Nishiwaki², and Satoshi Kagami³

¹ Robotics Institute, School of Computer Science, Carnegie Mellon University, USA

² School of Information Science and Technology, University of Tokyo, Japan

³ National Institute of Advanced Industrial Science and Technology (AIST), Japan

Abstract. We present an algorithm for planning goal-directed footstep navigation strategies for biped robots through obstacle-filled environments and uneven ground. Planning footsteps is more general than most existing navigation methods designed for wheeled robots, since the options of stepping over or upon obstacles are available. Given a height map of the terrain and a discrete set of possible footstep motions, the planner uses an A* search to generate a sequence of footstep locations to reach a given goal state. The planner evaluates footstep locations for viability using a collection of heuristic metrics designed to encode the relative safety, effort required, and overall motion complexity. We show preliminary results of the planner over several simulated terrains, as well as a simplified, online version of the algorithm running on the H7 humanoid robot. In the latter case, a stereo vision system is used to sense obstacles in the immediate environment and identify a target goal location, which is used to update the current optimal footstep sequence to the goal from the robot's present location.

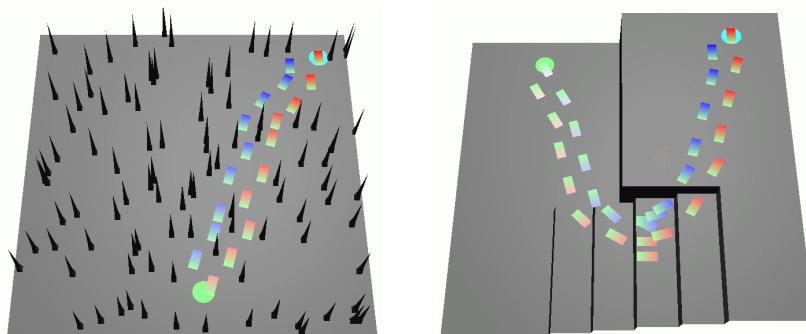


Fig. 1. Example footstep navigation strategies for complex scenes output by our planner (*left*: a room with spikes. *right*: two platforms connected by stairs.) The lighter end of each footstep signifies the front of the foot.

1 Introduction

As humanoid robots continue to advance and gain new capabilities, software for high-level control and autonomous motion generation will be required. One important area of research involves the design of algorithms to compute robust navigation strategies for humanoids in complex human environments. For indoor environments, this includes dealing with furniture, walls, stairs, doors, and possible objects 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.

The planner described in this paper allows the humanoid to take advantage of its bipedal capabilities and navigate through difficult environments where conventional 2D planning algorithms designed for wheeled robots would be unable to find a solution. Heuristics designed to minimize the number and complexity of the step motions are used to encode cost functions used for searching a footstep transition graph. If successful, the planner returns an optimal sequence of footstep locations according to the cost functions and plausible sets of footstep locations defined.

The rest of the paper is organized as follows: Section 2 gives an overview of related research, Section 3 describes our biped stepping model and the planning algorithm, Section 4 shows some results from both simulation and an online implementation using the humanoid H7, and Section 5 concludes with a summary discussion and directions for future research.

2 Background

Patla and colleagues have studied the problem of how humans perform local planning over irregular terrain based on visual feedback [PAM⁺96, Pat98, PNS00]. For humanoid robots, most existing research has focused on pre-generating stable walking trajectories (e.g. [HHHT98, YINT98, NII99]), or on dynamic balance and control (e.g. [VBSS90, PP99]). Recently, techniques have been developed to generate stable walking trajectories online [NSK⁺01, kNa], though these results do not account for obstacles. For quadruped robots, adaptive gait generation and control on irregular terrain and among obstacles has been previously studied [Hir84]. This method has not yet been applied to biped robots. Sensor-based obstacle-avoidance techniques have been developed for bipeds navigating in unknown environments [YL99, LDS⁺00]. However, such reactive methods can become trapped in local loops or dead-ends, since they do not consider global information. Other related techniques in computer animation that use footprint placement for motion specification have been developed for bipeds [Gir87, vdP97], and quadrupeds [KMB95, TvdP98]. We previously implemented a global planning approach for bipeds that generated statically-stable walking motions for a simulated H6 robot in cluttered 2D environments [KNK⁺]. In this paper, we handle complex height maps, uneven terrain, and allow stepping motions that

are not statically-stable. The most complex examples still run only in simulation, but we have created a version of the planner that uses a simplified robot walking model and obstacle collision checking routine to reduce the computational cost. Our current implementation is fast enough to enable continuous replanning during each step cycle, and has been run successfully on a real robot.

Our approach is to build a search tree from a discrete set of feasible footstep locations corresponding to candidate stable stepping motion trajectories. Using standard dynamic programming techniques, we compute a sequence of footstep placements to reach a goal region in an environment with obstacles. This has the advantage of planning a *global navigation strategy* for bipeds that includes the ability to step over obstacles. Encoded heuristics minimize the number and complexity of the steps taken, as well as guide the search. Given a proper admissible heuristic cost function, the planner can compute *optimal* sequences of footstep locations to reach a goal, according to the cost function defined. For the simplified online version, this strategy can be computed efficiently on standard PC hardware (usually in a fraction of a second for paths involving less than 20 steps to reach the goal). We have connected this version to an online stereo vision system and walking trajectory generator for the humanoid robot H7. The robot continuously updates a *walking area map* computed by the stereo vision system, and replans an optimal footstep sequence to reach the current goal.

3 Algorithm

The planner takes as input a map representing the terrain to plan over, an initial and goal state, and a discrete set of possible footsteps that can be taken. If a path is found the planner returns the solution as an ordered list of the footsteps that should be taken to reach the goal.

Biped Robot Model: The robot capabilities are modeled by storing a symmetric collection of candidate footstep transitions for each foot. The footsteps are represented as a displacement from the robot’s current state, which consists of $Q = (x, y, \theta, s)$. The state variables x , y , and θ denote the relative position and orientation of the footstep, and the binary variable $s \in \{R, L\}$ denotes which foot is currently the support foot (right or left). Four examples of footstep transition sets that we used in our experiments are illustrated in Figure 2.

State Transitions: Each footstep transition $T = (Q, cost, HC_{upper}, HC_{lower}, clearance)$ consists of a relative candidate footstep displacement Q , an associated *cost*, an upper and lower allowable height change HC_{upper} and HC_{lower} , and an obstacle clearance value, *clearance*, representing the largest obstacle that the transition can step over. Each transition is assumed to be independent of all previous transitions and the robot’s current state. Transition displacements, height change allowances, and obstacle clearances are chosen manually based on the actual robot’s capabilities in a way so as to cover the robot’s range of motion as well as desired. The associated costs are chosen manually to have the planner prefer certain transitions over others.

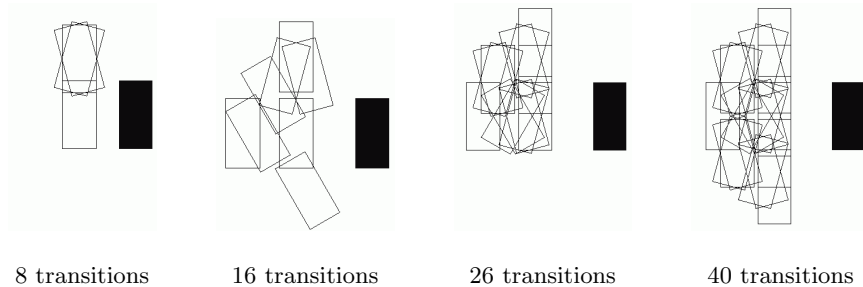


Fig. 2. Footstep transition sets. The transitions displayed are only those for the left foot (relative to the right foot shown in black).

Environment: The terrain map M is represented by a grid of cells. Each cell $c = (x, y, height, info)$ has an associated location in the grid (x, y) , a height value $height$, and an information value $info$. Together, the cells create a height map representing the form of the terrain the planner must overcome. The information values provide extra knowledge of the terrain, such as places which appear to be safe to step on but should be considered unsafe.

State Evaluation: The planner evaluates all transitions from a state, generating three costs for each one. First is the *location cost* $L(Q)$, 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 to. Second is a *step cost* $S(Q, T, Q_c)$, which computes the cost of reaching the state Q by making the transition T from the current state Q_c . 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 and the new foothold. Finally, the third cost is a heuristic $R(Q, Q_g)$ which estimates the *remaining cost* to reach the goal state, based on the distance to the goal, the angle of the robot to the goal, and the height of the robot relative to the goal. These costs can then be used by the planner in one of several different algorithms: best-first search, A* search, breadth-first search, dynamic programming, etc. Note also that the location cost is independent of the transition or current state of the biped, and thus can be precomputed for all states if desired, rather than computed for the needed states at run-time.

3.1 Location Metrics

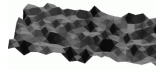
To evaluate a location’s cost, we would like to know exactly how the foot would come to rest on the surface, which parts of the foot would be supported, and be able to build a support polygon of the foot based on which parts of the foot are touching the ground and evaluate how stable that support polygon is. Unfortunately, this is very expensive to exactly compute. Instead, we use a set of metrics which can be quickly computed and serve to approximate this ideal

location cost. To be useful, a metric should be quick to compute, invariant to the number of cells at the location, and should eliminate/penalize an unwanted form of terrain while not eliminating or heavily penalizing good terrain.

To compute the metrics, we first determine the relevant cells to include. Based on the foot's positions and orientation, we test every cell in the rectangular box around it. We rotate these cells' coordinates into the foot's coordinate system and then test them against the foot's shape. This gives us a set of cells, \mathcal{C} , to use with each of the metrics defined below:



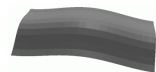
The slope angle of the surface at the candidate location. Perfectly horizontal surfaces are desired. The slope angle is computed by fitting a plane $h(x, y)$ to the cells in the location.



The "roughness" of the location. A measure of the deviation of the surface from the fitted plane. Computed by averaging the difference in height of each cell to the plane's height at the that cell.

$$\frac{1}{N} \sum_c^{c \in \mathcal{C}} |c.height - h(c.x, c.y)| \quad (1)$$

The "stability" of the location. This metric evaluates the curvature of the location. While perfectly flat is desired, curving down at the edges is penalized more than curving up at the edges. This metric is computed using a weighted sum of the heights (convolving with a dome-shaped filter).



$$\frac{1}{N} \sum_c^{c \in \mathcal{C}} \{[c.height - h(c.x, c.y)]g(c.x, c.y)\} \quad (2)$$

$g(x, y)$ is the dome-shaped filter. Restrictions on $g(x, y)$ are that it should be higher in the center than on the edges, and it should sum to zero over the area of the foot. The filter we used (in the foot's coordinates) was:

$$g(x, y) = \cos\left(\frac{2\pi x}{w}\right) + \cos\left(\frac{2\pi y}{l}\right) \quad (3)$$

w and l are the length and width of the foot.



The largest bump of the location. Bumps above the fitted plane are much worse for a location than holes in the plane. This metric finds the largest deviation above the plane.

$$\max\{c.height - h(c.x, c.y) | c \in \mathcal{C}\} \quad (4)$$



The "safety" of the location. This refers to the area around the location. It's purpose is to take into account the possible inaccuracy of foot positioning. This can be computed using the roughness and largest bump metrics, using the cells around the foot location.

Each of the metrics has both a weight and a cutoff value associated with it. If any metric passes its cutoff value, the location is discarded. Otherwise, the location's cost is the weighted sum of these metrics.

3.2 Step Cost

The cost of taking a step is given by:

$$S(Q, T, Q_c) = T.cost + w_h |H(Q, Q_c)| \quad (5)$$

$T.cost$ is the cost of the transition T . $H(Q, Q_c)$ is the height change between the state Q and the current state of the robot Q_c . w_h is the penalty for height changes, chosen manually based on the user's desire to try to avoid paths that go up or down. If the height change is outside the transition's allowable range, the step is discarded. An obstacle collision check is also done to determine if the foot can safely clear all the cells along the path from its previous location to its new location. Because this path may include many cells, quadtrees that encode the maximum height are used to quickly check for collisions (see Figure 3). If collisions are found, the step is discarded.

3.3 Estimated Cost Heuristic

The estimated remaining cost to reach the goal is calculated by:

$$R(Q) = w_d D(Q, Q_g) + w_\theta |\Theta(Q, Q_g)| + w_h |H(Q, Q_g)| \quad (6)$$

$D(Q, Q_g)$ is the euclidean distance from the state Q to the goal state Q_g . $\Theta(Q, Q_g)$ is the relative angle of the robot in state Q to the goal. $H(Q, Q_g)$ is the difference in height between the state Q and the goal state. The weights w_d and w_θ are chosen based on the costs associated with the transitions. w_h is the penalty for height changes applied during step cost calculations.

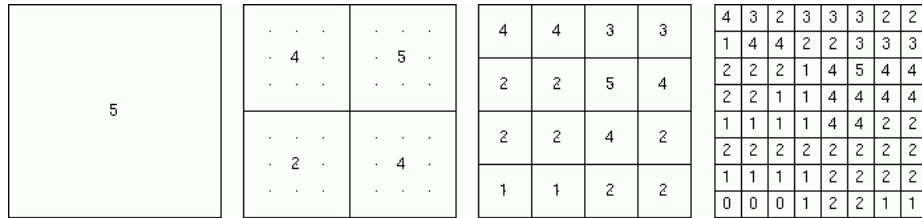


Fig. 3. An example quadtree constructed so that the value of a cell is the maximum value of its four children on the next level. This allows large amounts of the terrain to quickly be eliminated when searching for obstacle collisions.

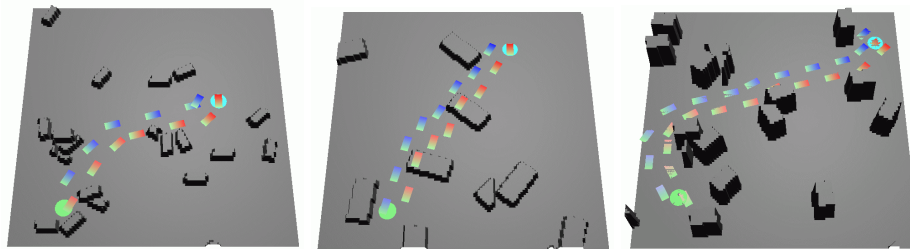


Fig. 4. Cluttered terrain with obstacles that can be stepped over (*left*), stepped onto (*center*), or large obstacles that must be stepped around (*right*)

4 Results

This section presents some preliminary results using prototype planners running in both a simulation environment, and on the humanoid 'H7'. The simulation results were computed on a 1.8 GHz Pentium4 PC with 1GB RAM running RedHat Linux 7.1. The planner was able to find suitable paths for a wide variety of terrain types.

Figure 4 shows examples involving cluttered terrain with obstacles that can be stepped over, stepped onto, or large obstacles that must be stepped around. Figure 5 illustrates two different multi-level terrains with planned footsteps, including an environment with clear stairs, and an environment with obstacle-filled stairs and platforms of varying heights. The two examples in Figure 6 show the planned output over terrain with obstacles on uneven ground.

4.1 Distance to Goal

Over flat terrain, the running time of best-first search scales linearly with the distance to goal. A* and truncated A* also scale roughly linearly with distance to goal. Figure 7 compares the performance of A* and BFS for walking towards a distant goal located at various distances along a forward diagonal line. The fluctuations in the total calculation times are due to the goal state falling in between two preferred footstep transitions.

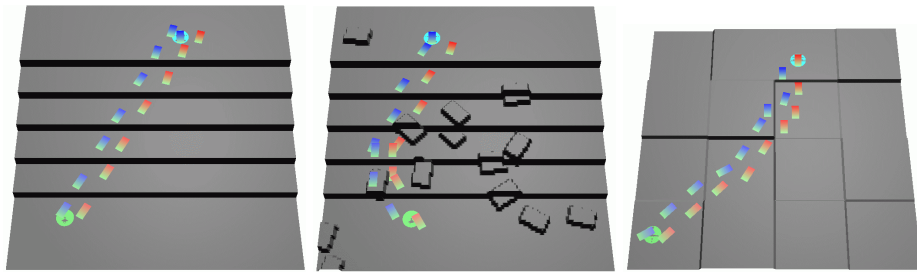


Fig. 5. Multi-level terrain with footsteps planned over clear stairs (*left*), obstacle-filled stairs (*center*), and platforms of varying heights.

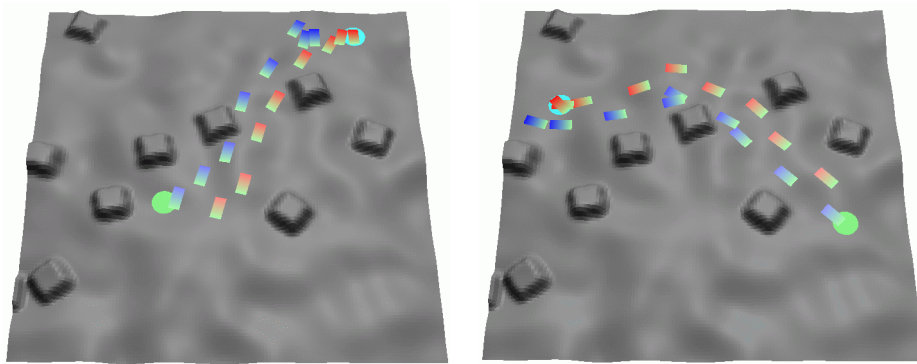


Fig. 6. Planning over terrain with obstacles on uneven ground.

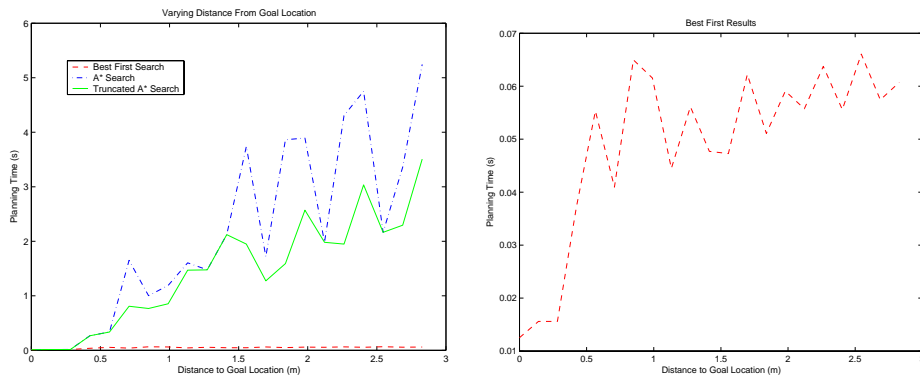


Fig. 7. Performance comparison of A* and BFS for walking towards a distant goal located at various distances along a forward diagonal line.

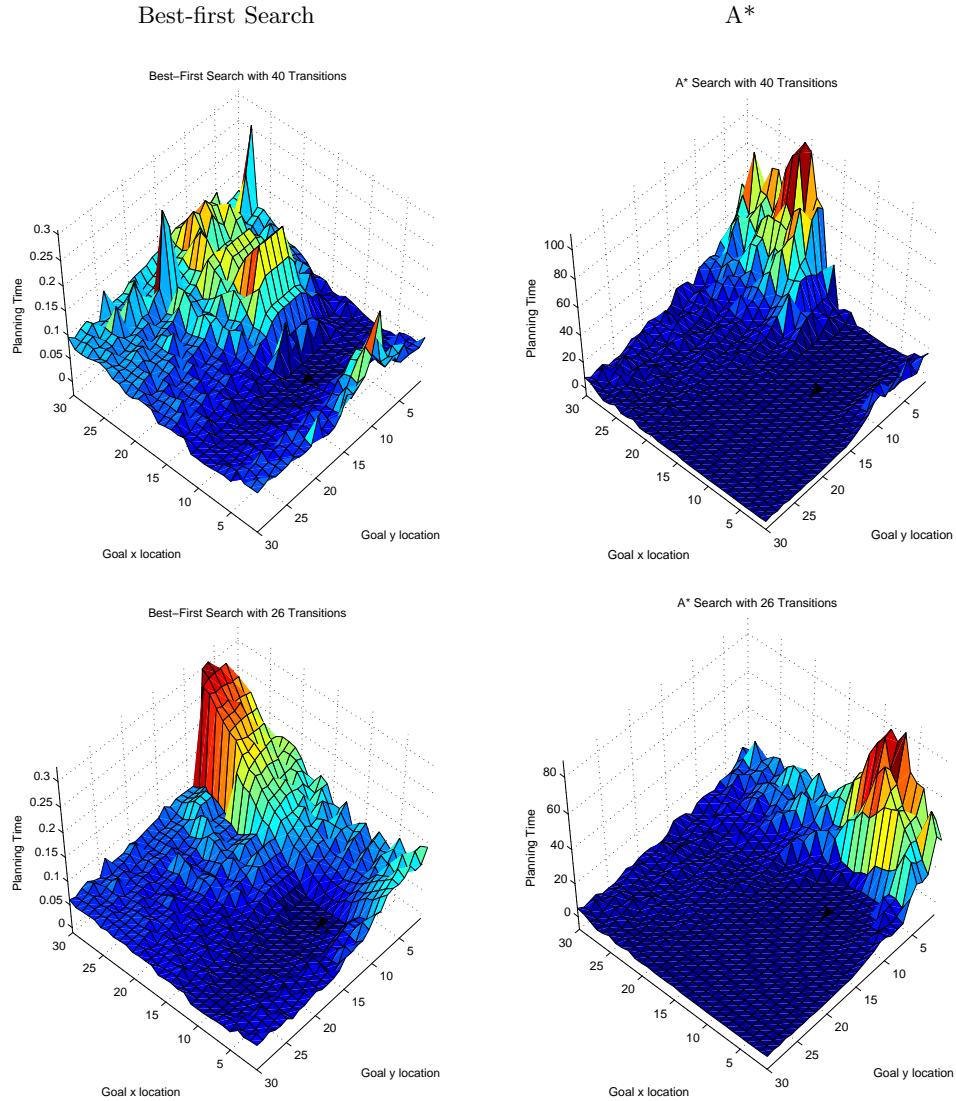


Fig. 8. Performance comparison of BFS (*left*) and A* (*right*) for different sets of available footstep transitions. Each row shows plots of the total computation time required for various goal locations for a robot that can step forwards, sideways, and backwards (*top row, transitions=40*), and for a robot that can only step forwards and sideways (*bottom row, transitions=26*). The initial state of the robot is at location (10,10) in each graph (as indicated by the small arrow), facing along the positive y-axis.

The plots in Figure 8 compare the performance of BFS (*left*) and A* (*right*) for different sets of available footstep transitions. Each row shows plots of the total computation time required for various goal locations for a robot that can step forwards, sideways, and backwards (*top row, transitions=40*), and for a robot that can only step forwards and sideways (*bottom row, transitions=26*). The initial state of the robot is at location (10,10) in each graph (as indicated by the small arrow), facing along the positive y-axis. Interestingly, the regions to the back and right relative to the robot’s current location can be planned significantly faster using the forward-only transitions. The reason for this difference appears to arise from having fewer movement options that actually make progress towards a goal at that location, whereas having more movement options causes the planner to expand many more relatively equal-cost paths.

4.2 Transitions and Obstacle Effects

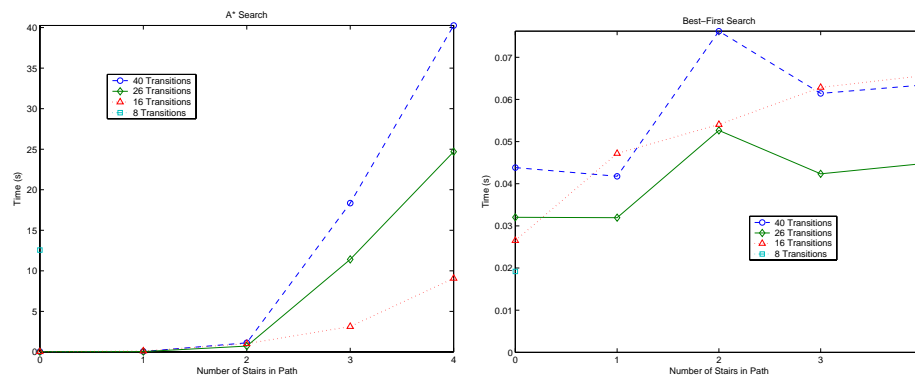


Fig. 9. Performance comparison of A* (*left*) and BFS (*right*) for increasing numbers of stairs along the path from the initial to goal state. The transition set containing only 8 transitions is fails to find a path as soon as the first step is added.

In general, decreasing transitions decreases runtime, since the branching factor is smaller. However, this smaller number of options decreases the set of solvable maps. The example in Figure 9 compares the performance of A* (*left*) and BFS (*right*) for increasing numbers of stairs along the path from the initial to goal state. Here, the smallest number of transitions was unable to get past the first stair, and could only solve the empty map.

In the presence of local minima, Best-first search can increase significantly the running time and often generates highly undesirable paths. Figure 10 shows a comparison of the output of BFS versus A* on environments with local minima. In the example with deep local minima, A* search outperforms Best-first in running time as well as producing optimal footstep paths.

Best-first Search

A*

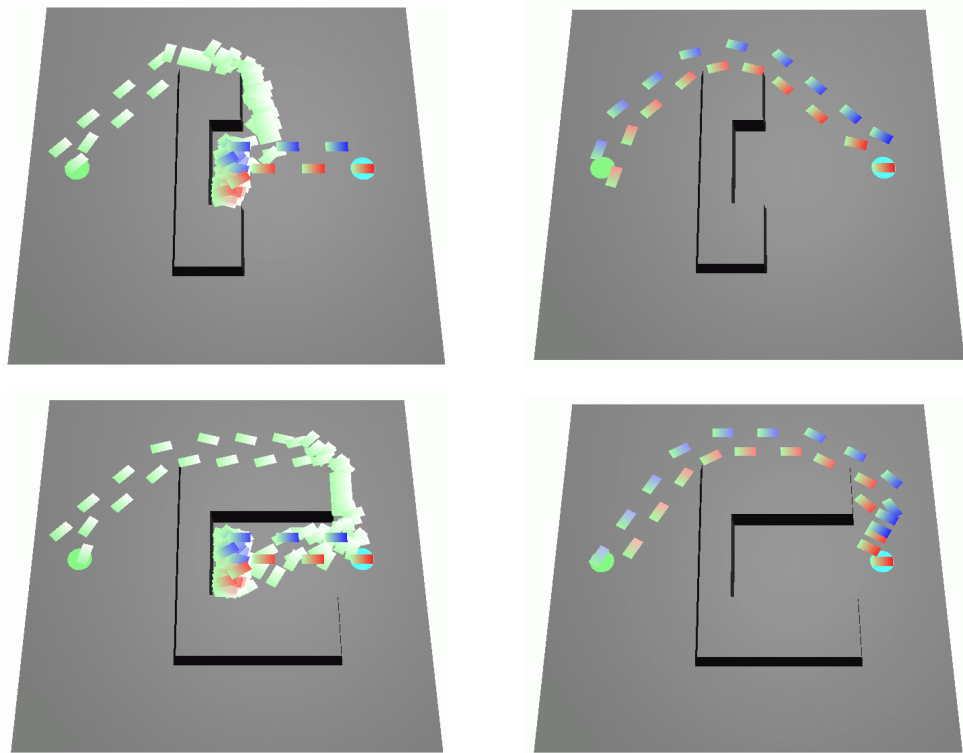


Fig. 10. Comparison of the output of BFS versus A* on environments with local minima. With deep local minima (*bottom*), A* search outperforms Best-first in running time as well as optimality.

4.3 Metric Weights

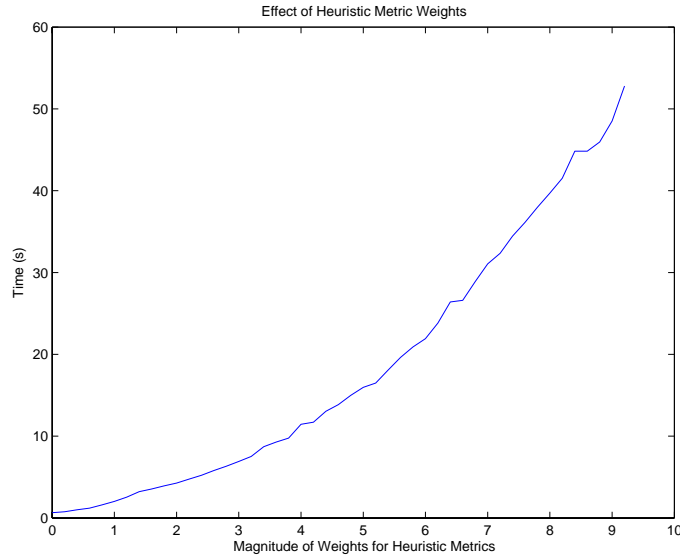


Fig. 11. Performance effect of scaling the weights used for the heuristic metrics. Magnitude zero corresponds to zero weights. Magnitude 1 corresponds to the manually chosen weights for each metric.

By scaling the weights of the metrics applied to each location, we can see that they bring about an exponential increase in the running-time of the algorithm. The increase in the weight of the location cost encourages the planner to try more locations, in the hope of finding a more optimal path. Figure 11 shows that carefully choosing the weights for the different metrics is very important for the runtime of the algorithm. In many cases, it may be better to simply use the metrics for discarding locations via their cutoff values, and lower the metric weights to zero.

4.4 Online Footstep Planning Experiments

We have developed a simplified, online version of the algorithm running on the H7 humanoid robot. A stereo vision system is used to sense obstacles in the immediate environment and identify a target goal location, which is used to update the current optimal footstep sequence to the goal from the robot's present location. This strategy can be computed efficiently on standard PC hardware (usually in a fraction of a second for paths involving less than 20 steps to reach the goal). The output of the footstep planner is sent to a walking trajectory generator for the humanoid robot H7. The robot continuously updates a *walking*

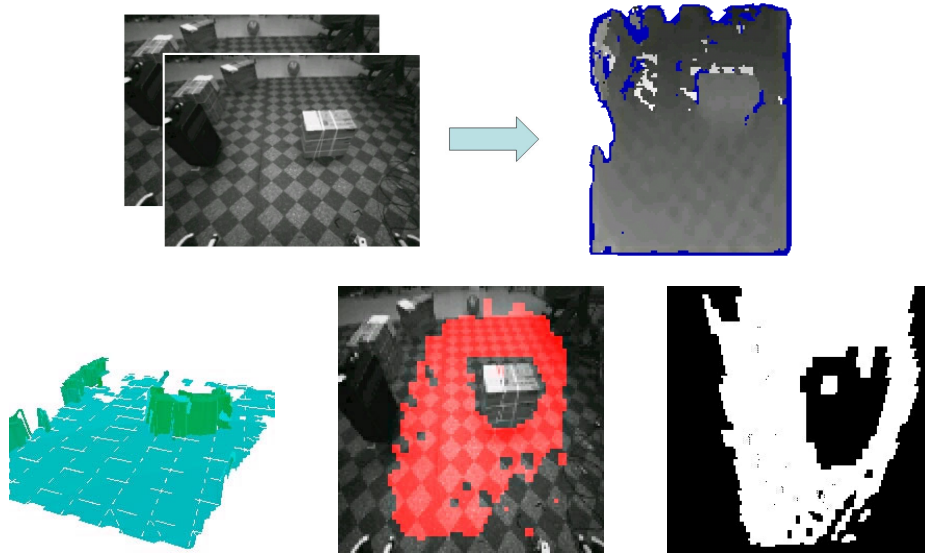


Fig. 12. Vision Processing steps. (*top row:*) raw camera images and resulting 3D Depthmap; (*bottom row:*) mesh model, planar surface identification, and final walking area map.



Fig. 13. Online footstep planning: (*top row:* H7 humanoid and candidate walking surface area extracted from 3D stereo depth map; *bottom row:* distortion-corrected obstacle map after thresholding, and calculated optimal footstep sequence to target.



Fig. 14. Snapshots of the humanoid H7 calculating and executing online the footstep sequence from Figure 13

area map computed by the stereo vision system, and replans an optimal footstep sequence to reach the current goal.

Figure 12 shows the output of the 3D vision system. The vision information is used to calculate the walking area map, which is used as input to the planner. Figure 13 shows a calculated footstep sequence for a model of the humanoid robot H7 navigating in the laboratory. The sequence is computed to reach a circular goal region determined by a red ball that is tracked in real time. Figure 14 shows snapshots taken while running the goal following planning program. There were a total of 19 discrete foot placements considered for each foot in the online version of the planner.

5 Discussion

Due to the computation time involved in planning some of the longer paths, a tiered planned can be used which first simply plans for a robot-sized circle to move through to environment. If a path is found, footsteps can be filled in along this path using heuristics. This allows the planner to leverage the substantial research for mobile robot planning, and fall back to this more detailed planning whenever the first round of fast planning fails.

Careful thought needs to go into designing transition sets. Lower numbers of transitions result in a lower branching factor and faster run times, sometimes much faster, but can result in slower times and sometimes no solution when the set of transitions does not sufficiently cover the space of possible transitions. Also, it is not just the number of transitions that is important, but their arrangement and associated costs.

Best-first search consistently outperforms A* search except in cases of severe local minima, but can yield undesirable paths in the way it circumvents obstacles

in the path. By increasing the weight of the estimated cost heuristic relative to the location cost and step cost, A* can be brought closer to best-first by sacrificing some optimality to lower the running time, depending on the user's requirements for both optimality and speed.

Many improvements could be built upon this work to increase the capabilities of the algorithm, including:

1. State- or Transition- dependent transitions, allowing for different gaits or actions to be encoded.
2. Calculation of Waypoints, to break a large planning problem into shorter problems to avoid the exponential growth that comes with more obstacles.
3. Dynamically generated transitions which can increase or decrease the number of transitions depending on the environment, or adjust the locations of transitions to better match the terrain (e.g. set stride length to match stair length).

We are currently working on extending to algorithm in these directions, as well as conducting more complex, vision-guided footstep planning experiments with the H7 humanoid robot.

References

- [Gir87] M. Girard. Interactive design of computer-animated legged animal motion. *IEEE Computer Graphics & Applications*, 7(6):39–51, June 1987.
- [HHHT98] 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.
- [Hir84] S. Hirose. A study of design and control of a quadruped walking vehicle. *Int. J. Robotics Research.*, 3(2):113–133, Summer 1984.
- [KMB95] E. Kokkevis, D. Metaxas, and N. I. Badler. Autonomous animation and control of four-legged animals. In *Proc. Graphics Interface*, pages 10–17, May 1995. ISBN 0-9695338-4-5.
- [kNa] k. Nishiwaki and. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02)*.
- [KNK⁺] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'01)*.
- [LDS⁺00] O. Lorch, J. Denk, J. F. Seara, M. Buss, F. Freyberger, and G. Schmidt. ViGWaM - an emulation environment for a vision guided virtual walking machine. In *Proc. IEEE Int. Conf. on Humanoid Robotics (Humanoids 2000)*, 2000.
- [NII99] 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.
- [NSK⁺01] 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.

- [PAM⁺96] A.E. Patla, A. Adkin, C. Martin, R. Holden, and S. Prentice. Characteristics of voluntary visual sampling of the environment for safe locomotion over different terrains. *Exp. Brain Res.*, 112:513–522, 1996.
- [Pat98] A.E. Patla. How is human gait controlled by vision? *Ecological Psychology*, 10:287–302, 1998.
- [PNS00] A.E. Patla, E. Niechwiej, and L. Santos. Local path planning during human locomotion over irregular terrain. 2000.
- [PP99] 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.
- [TvdP98] N. Torkos and M. van de Panne. Footprint-based quadruped motion synthesis. In *Proc. Graphics Interface*, pages 151–160, 1998.
- [VBSS90] M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic. *Biped Locomotion: Dynamics, Stability, Control, and Applications*. Springer-Verlag, Berlin, 1990.
- [vdP97] M. van de Panne. From footprints to animation. In *Proc. Computer Graphics Forum*, volume 16, pages 211–223, October 1997.
- [YINT98] 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.
- [YL99] M. Yagi and V. Lumelsky. Biped robot locomotion in scenes with unknown obstacles. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA '99)*, pages 375–380, Detroit, MI, May 1999.