Real-Time Path Planning for Humanoid Robot Navigation

Jens-Steffen Gutmann **Masaki Fukuchi** **Masahiro Fujita**
Intelligent Systems Research Laboratory
Sony Corporation, Tokyo, Japan
{steffen,fukuchi,mfujita}@pdp.crl.sony.co.jp

Abstract

We present a data structure and an algorithm for real-time path planning of a humanoid robot. Due to the many degrees of freedom, the robots shape and available actions are approximated for finding solutions efficiently. The resulting 3 dimensional configuration space is searched by the A* algorithm finding solutions in tenths of a second on low-performance, embedded hardware. Experimental results demonstrate our solution for a robot in a world containing obstacles with different heights, stairs and a higher-level platform.

1 Introduction

Path-planning is one of the fundamental problems in mobile robot navigation. It has been shown long before that the problem of moving an object through space is *PSPACE-hard* with a time complexity exponential in the degrees of freedom of the object [Latombe, 1991]. In mobile robots the degrees of freedom are usually small (three or less) which opens the application of a range of search techniques specifically tailored to the problem. If one can neglect orientation, e.g. a cylindrical holonomic system, the resulting 2 dimensional configuration space can be searched efficiently by A*, D* [Stentz, 1995], or dynamic programming [Buhmann *et al.*, 1995].

Humanoid robots while being mobile allow for more than the usual three degrees of freedom. Their feet can be placed with a greater choice and the change of body posture allows to overcome obstacles where wheeled robots fail in passing through. This enables humanoids to step onto objects [Kuffner *et al.*, 2002], climb up and down stairs [Hirai *et al.*, 1998; Gutmann *et al.*, 2004], step over obstacles [Guan *et al.*, 2004], or crawl underneath objects [Shiller *et al.*, 2001; Kanehiro *et al.*, 2004].

The higher degrees of freedom can be tackled e.g. by probabilistic roadmaps [Kavraki *et al.*, 1996]. However, if one can employ a non-probabilistic approach by finding a suitable approximation then the solution can be seen preferable due to its deterministic nature. One possible approximation is to model the robot as a cylinder or by its convex hull which allows for efficient collision checking [Okada *et al.*, 2003].

Care should be taken though not to over-simplify the problem. Humanoid robots are non-holonomic, i.e. they cannot turn in place without requiring additional space. The trajectory of the body center describes a curve similar to a car-like robot, although the turn radius is generally quite small. It is possible to account for the extra turning space in a cylinder model of the robot enlarged by twice the turn radius [Li *et al.*, 2003; Sabe *et al.*, 2004]. However such an approximation is overly pessimistic and prevents the robot from passing through narrow space as we will show.

Discretizing the possible actions of the robot into a small set of well-chosen actions, e.g. different foot placements, leads to another way for finding solutions efficiently. The induced search graph grows exponentially with the number of steps and thus, only a small number of foot steps can be explored in a real-time system [Lorch *et al.*, 2002]. Such a foot-step planning system is nevertheless a valuable tool in a hybrid system where a higher level planner provides waypoints for this sub-system [Chestnutt and Kuffner, 2004].

Shiller *et al.* [2001] discretize the actions and orientations of a digital figure that can walk forward, sideways, turn and crawl. They maintain an obstacle space for each combination of action and orientation and find solutions efficiently for trajectories of several ten steps.

We compute a trajectory for the body-center of a humanoid robot by approximating the shape of the robot using multiple cylinders and discretizing configurations into a set of orientations. When following this trajectory, the next few foot steps are computed using an analytical method. This enables a humanoid robot to find paths of up to a few meters in real-time and includes actions such as sideways walking through narrow space and climbing up and down stairs.

Our work differs from Shiller's in that we do not plan behaviors but select the behavior depending on the environment when following a found path, e.g. stair-climbing over a sill. We also maintain only one obstacle space and show how to perform collision checking efficiently. Last but not least, our system is demonstrated on a real robot where stereo range data incrementally updates a representation of the world.

Our approximations of configuration space and actions are presented in the next section. Section 3 briefly describes our approach for generating a navigation map for collision checking. Section 4 details our application of the A* algorithm for path searching. Experimental results on Sony's humanoid robot QRIO are described in Section 5. Finally, we discuss limitations and extensions of our approach in Section 6.

2 Configuration Space and Actions

The robot moves in a 2 dimensional world. Obstacles, different floor levels and stairs are projected onto one plane. We do not neglect the orientation of the non-holonomic robot, thus the configurations space C is 3 dimensional. For allowing efficient search we discretize C into a regular grid of evenly spaced cells using 8 different orientations:

$$\begin{aligned} C &= X \times Y \times \Phi \\ X, Y &= \{i \cdot cs \mid 0 \leq i < n\} \\ \Phi &= \{i \cdot 45^\circ \mid 0 \leq i < 8\} \end{aligned} \quad (1)$$

where cs defines the cell size and n the number of cells.

We restrict the movement of the robot's body center to the following six actions $A = \{$ *forward, backward, turn-left, turn-right, side-left, side-right*$\}$. For each action the transition in C is given for orientations 0° and 45° in Figure 1. For other orientations the transitions are analogous.

When concatenating turn-left (or turn-right) actions, the resulting trajectory describes a curve, see Figure 2(a). The turn actions thus model the non-holonomic constraint of a humanoid where changing the orientation requires extra space for turning. If we approximate this curve with an arc of a circle, we find that the cell size cs coincides with the turn radius of the robot and therefore cs should be chosen as such.

The shape of the robot is approximated by two cylinders sharing the same axis as shown in Figure 2(b). Cylinders allow for fast collision checking and the smaller cylinder for the legs enables the robot to pass close to low obstacles where upper body and arms are above the obstacle.

The cylinders are a tight fit to the shape of the robot standing still. When the robot moves, however, additional space is needed for the transition in C as body and legs are swinging while walking. Figure 3 shows snapshots of the robot when moving forward, turning left and stepping sideways. We account for the additional space by enlarging the cylinders at the start and end configurations depending on the action. Both, upper and lower, cylinders are enlarged by the same amount. Only when walking sideways, no additional space is needed. As we will see in Section 5 this enables the robot to pass through narrow space by choosing the sideways walk.

It is worth noting that the humanoid robot does not necessarily need to be able to exactly perform the 6 actions defined in this section. For example, the robot could well use several foot steps for performing the 45° rotation in our turn actions. The only requirement is that the enlarged cylinders cover all the space the robot requires for completing the 45° turn.

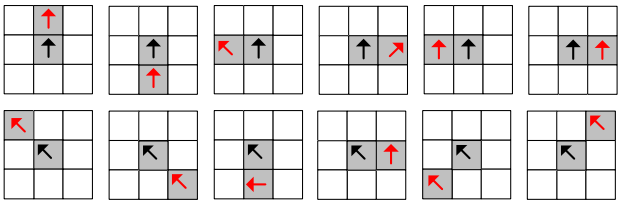
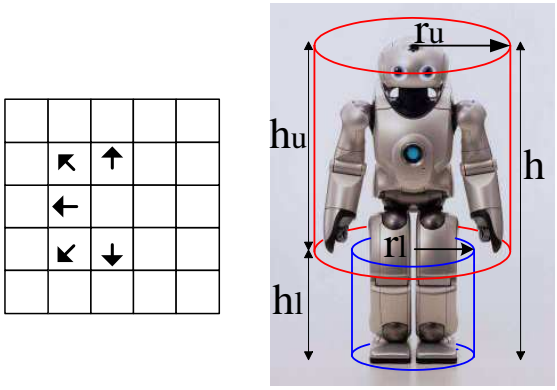


Figure 1: Change in configuration for our 6 actions.



(a) (b)

Figure 2: (a) Turn around by concatenating 4 turn-left actions. (b) Multi-cylinder model for approximating robot shape.

One might think of our approach to be specifically tailored to humanoids as the proposed actions and the cell size matching the turn radius are probably only found for this kind of robot. While this seems to be true, the contribution of this work is to demonstrate the abilities of such a system in a complex real-world environment. We will discuss the limitations and possible extensions of our approach in Section 6.

3 Navigation Map and Collision Checking

Collision checking is one of the crucial factors in path planning. Cylinder models of robots allow to perform this check in constant time if the distance to the nearest obstacle is known; the distance is simply compared to the radius of the cylinder. Our two cylinder model allows to check for collision in the same amount of time.

For efficient collision checking we make the following assumptions about the world the robot is navigating in:

1. The world is partitioned into floor and obstacles.
2. Floor is planar and horizontal (else it is an obstacle).
3. There are no multiple floor levels at the same location.
4. The robot is able to distinguish between floor and obstacles and can can estimate their relative position and height using its sensors.

Assumption 1 is general, 2 rules out inclined surfaces and treats them like obstacles, 3 is true for most but a few environments and enables the use of a 2.5D map for representing

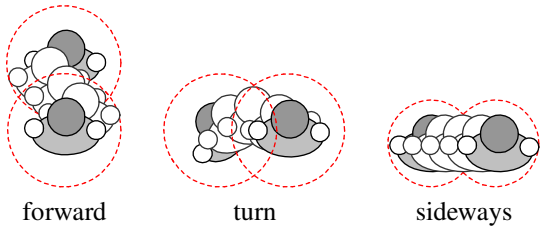


Figure 3: Space required for performing an action. Enlarged upper cylinders approximate the additional space needed.

the world. Finally, assumption 4 is a practical consideration and can be achieved e.g. by plane extraction from range data.[1]

Under these assumptions, the robot is able to build a floor and obstacle height map FOG of its environment. We employ a combination of a 3D occupancy grid with a 2.5D floor height map where range data from stereo vision is segmented into planes and integrated in a probabilistic way [Gutmann *et al.*, 2005]. Each cell in FOG holds a type and a height:

$$FOG : (x, y) \quad \mapsto \quad (t, h), \tag{2}$$

with $x \in X,\ y \in Y,\ t \in T = \{floor, obstacle, unknown\}$, and $h \in R$ is the height of the floor or obstacle.

From this map a navigation grid *NAV* is computed storing refined floor types and distances to nearest obstacles:

$$NAV : (x, y) \quad \mapsto \quad (t', d) \tag{3}$$

where $t' \in T' = T \cup \{stairs, border\}$ and $d \in R$ is the clearance, i.e. the distance to the closest obstacle for $t' \in \{floor, stairs\}$. *NAV* refines the type of $FOG(x, y) = (t, h)$ for $t = floor$ in the following way. Let $U(x, y)$ be a neighborhood around position (x, y) and $\Delta\hat{h}$ be the maximum absolute difference in floor height to cells in $U(x, y)$:

$$\Delta\hat{h} = \max_{(x',y') \in U(x,y), FOG(x',y') = (floor, h')} |h - h'|. \tag{4}$$

The refined type is then computed as:

$$\begin{aligned} refine(floor) \ &= \ \begin{cases} floor, & \Delta\hat{h} \le d_{floor} \\ stairs, & d_{floor} < \Delta\hat{h} \le d_{stairs} \\ border, & d_{stairs} < \Delta\hat{h} \end{cases} \\ refine(t) \ &= \ t, \qquad t \neq floor \end{aligned} \tag{5}$$

where d_{floor} is the maximum step height for regular walk and d_{stairs} the corresponding one for climbing up or down stairs.

The refinement of floor cells allows to treat stairs and borders differently from regular floor. We will make use of stairs in path-planning by assigning a different cost for traversal. Borders are treated similar to obstacles and as such prevent the robot from getting close to or over the end of a floor layer.

The clearance of a cell is defined in the following way. For a cell (x, y) with $FOG(x, y) = (floor, h)$ we compute the difference in height $\Delta h(x', y')$ to cell (x', y') with $FOG(x', y') = (t', h'), refine(t') \in \{obstacle, border\}$:

$$\Delta h'(x', y') \quad = \quad h' - h. \tag{6}$$

This height difference then decides which cylinder of our two-cylinder model comes into effect for computing the distance $d'(x', y')$ to this obstacle, or if it can be ignored:

$$d'(x', y') \quad = \quad \begin{cases} e - r_l, & 0 \le \Delta h'(x', y') < h_l \\ e - r_u, & h_l \le \Delta h'(x', y') \le h \\ \infty, & \text{otherwise} \end{cases} \tag{7}$$

where $e = \sqrt{(x' - x)^2 + (y' - y)^2}$ is the Euclidean distance between the cells, and r_l, r_u and h_l, h_u are the radii and heights of the two cylinders as depicted in Figure 2 (b).

[1] It should be possible to relax these assumptions and allow for inclined surfaces. However, the model for representing the world would become more complicated which is why we do not consider them in this paper.

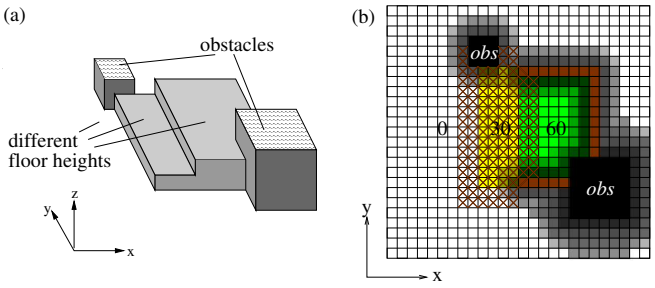


Figure 4: Example navigation map for a simulated world.

The clearance $d(floor)$ of cell (x, y) is then defined as the minimum distance to all obstacles and borders.

$$d(floor) = \min_{(x',y') \in X \times Y, NAV(x',y') = (t', \cdot), t' \in \{obstacle, border\}} d'(x', y') \tag{8}$$

The computation of this value can be implemented efficiently by *region growing* of obstacles and borders until a preset maximum distance is reached.

Note that (8) only applies for cells where $NAV(x, y) = (floor, \cdot)$. For other types $t \in T$ the clearances are set to:

$$d(obstacle) = -r_l, \qquad d(unknown) = \infty. \tag{9}$$

For the robot being able to stand at a given position (x, y) with $NAV(x, y) = (\cdot, d)$, the clearance d must be positive in order not to collide with an obstacle. For unknown terrain we are optimistic in that there could be a floor height well above all obstacles and thus ignore obstacles.

We can define the free configuration space C_{free} as

$$C_{free} = \{(x, y, \theta) \in C \mid NAV(x, y) = (\cdot, d), d > 0\}. \tag{10}$$

As an example Figure 4(a) shows a simulated world fully observed by an ideal sensor. The resulting *NAV* grid is shown in Figure 4(b). Floor heights are drawn in white, yellow and green, stairs are marked with brown crosses, the border is filled in brown, and obstacles are black. Different shades of gray show the clearance of cells where brighter cells indicate a larger clearance than darker ones. Note how the taller obstacle in the bottom right *grows* more to the bottom and right of the map (where the distance to the floor is larger than h_l) and less to the higher level floor.

4 Path-Planning

We have now the basic tools for defining the path-planning problem in our domain. Let $c, c' \in C$ be configurations. The transition $c' = succ(c, a)$ for a single action $a \in A$ is given in Figure 1. We define $succ$ for a sequence of actions $a_1 a_2 \dots a_m$ in a recursive way as

$$succ(c, a_1 a_2 \dots a_m) = succ(succ(c, a_1), a_2 \dots a_m). \tag{11}$$

In order to test whether action a is *applicable* in configuration c, we check the type and clearance of c in *NAV*:

$$\begin{aligned} app(c, a) = allow(c, a) \wedge \\ clear(c, r_a) \wedge clear(succ(c, a), r_a) \end{aligned} \tag{12}$$

where $allow(c, a)$ forbids certain actions depending on the type t' in *NAV* at the position of c (e.g. we do not allow backward and sideways walk in unknown terrain and only allow

forward walk on stairs), r_a is additional space required for execution of a (see Figure 3), and

$$clear(c, r) = d > r, \quad c = (x, y, \cdot), NAV(x, y) = (\cdot, d). \quad (13)$$

We also define app on a sequence of actions $a_1 a_2 \dots a_m$

$$app(c, a_1 a_2 \dots a_m) = app(c, a_1) \wedge app(succ(c, a_1), a_2 \dots a_m). \quad (14)$$

and call the sequence a *path* if (14) holds.

The problem of path-planning for given start and goal configurations $c_{start}, c_{goal} \in C_{free}$ is then to find a path $\pi = a_1 a_2 \dots a_m$ such that

$$c_{goal} = succ(c_{start}, \pi). \quad (15)$$

Problems for which a path exists are called *solvable*. In general there can be more than one solution for a given solvable problem. We are interested in an *optimal* path, i.e. a path $\hat{\pi}$ that minimizes a cost function g over the path:

$$\hat{\pi} = \underset{\pi}{\operatorname{argmin}}\ g(c_{start}, \pi). \quad (16)$$

In our system we found the following cost function valuable:

$$g(c, a_1) = g_a(a_1) + g_{t'}(c') + g_o(c') \quad (17)$$

$$g(c, a_1 a_2 \dots a_m) = g(c, a_1) + g_c(a_1, a_2) + g(c', a_2 \dots a_m) \quad (18)$$

where $c' = succ(c, a_1)$, g_a is a cost depending on the action (e.g. forward walk has less cost than sideways which takes more time), $g_{t'}$ a cost depending on the type t' in the *NAV* grid ($floor$ is preferred over $stairs$ and $unknown$), g_c a constant for changes in action (the robot might have to do additional foot steps), and g_o considers the clearance of c' in *NAV* in order to prefer safer paths more distant from obstacles:

$$g_o(c) = \hat{g}_o \max(d_{max} - d, 0). \quad (19)$$

Here d_{max} is the maximum clearance above which no further improvement in safety is expected, d is defined as in (13), and $\hat{g}_o$ is a predefined constant.

We employ the well-known A* algorithm for finding solutions. A* maintains a *priority queue* where candidates are ranked according to an evaluation function f composed of the cost function g and an estimated *heuristic* h to c_{goal}:

$$f(c_{start}, \pi) = g(c_{start}, \pi) + h(succ(c_{start}, \pi)). \quad (20)$$

For our chosen discretization of C we can employ the following heuristic which is *optimistic* and a larger cost estimate than the Euclidean distance:

$$h(c) = \hat{g}_a \cdot \begin{cases} \Delta x - \Delta y + \sqrt{2}\Delta y, & \Delta x > \Delta y \\ \Delta y - \Delta x + \sqrt{2}\Delta x, & \Delta x \le \Delta y \end{cases} \quad (21)$$

where $\Delta x = |x - x_g|$ and $\Delta y = |y - y_g|$ are absolute coordinate differences from $c = (x, y, \cdot)$ to $c_{goal} = (x_g, y_g, \cdot)$ and $\hat{g}_a$ is the minimum action cost.

From the properties of the A* algorithm and our chosen cost and heuristic functions it follows that A* finds the cost optimal path $\hat{\pi}$ according to (16) if a solution exists.

An important factor for the efficiency of A* is to detect whether a configuration has already been visited before. By using a regular grid, we can pre-compute the search graph (but not the optimal paths since the environment is unknown) and check for duplicates in constant time. This is an advantage over methods that build the tree at the time of search as checking for duplicates needs further processing e.g. by applying nearest-neighbor methods.

Robot specifications and grid dimensions

r_l = 60mm	h_l = 100mm	d_{floor} = 15mm	cs = 40mm
r_u =140mm	r_h = 500mm	d_{stairs} = 50mm	n = 100

Allowed actions depending on *NAV* type

$allow(c, a)$	$floor$	$unknown$	$stairs$	$border$	$obstacle$
$forward$	**true**	**true**	**true**	false	false
$turn$	**true**	**true**	false	false	false
$sideways$	**true**	false	false	false	false
$backward$	**true**	false	false	false	false

Additional clearance r_a depending on action

a	$forward$	$turn$	$sideways$	$backward$
r_a	60mm	80mm	0	60mm

Cost of action $g_a(a)$ depending on orientation

$g_a(a)$	$forward$	$turn$	$sideways$	$backward$
0°	1	1.1	1.3	2
45°	$\sqrt{2}$	1.1	$1.3\sqrt{2}$	$2\sqrt{2}$

Cost of action $a_{t'}$ depending on *NAV* type

	$floor$	$unknown$	$stairs$	$border$	$obstacle$
$g_{t'}(c)$	0	0.5	1	∞	∞

Other constants for cost function

$g_c(a, a) = 0$	$g_c(a_1, a_2) = 0.25$,	$a_1 \neq a_2$
d_{max} = 200mm	$\hat{g}_o = 3/d_{max}$	$\hat{g}_a = 1$

Table 1: Parameters used in our implementation

5 Results

We implemented our approach on QRIO, Sony's small humanoid robot with 38 degrees of freedom [Fujita *et al.*, 2003]. The robot is equipped with 3 MIPS R5000 CPUs clocked at 400MHz and a stereo camera system with a $45°$ field of view providing disparity images at 12.5 fps. This stereo data is segmented into planes from which our FOG and *NAV* grid representations are updated in real-time.

Table 1 lists parameters used in our implementation for all variables introduced in the previous sections. For stairs we restrict actions to forward walk only. Thus, the robot only finds paths on straight staircases unless there is a larger floor area where turns are possible. We forbid sideways and backward walk in unknown terrain because the robot is not able to see into its moving direction. We set a high cost for backward motion since it is generally more unsafe. Costs are added for walking in unknown terrain and on stairs which focuses the algorithm to search for a path on known floor even if the distance traveled is larger than when moving on other terrain.

For our experiments we built a 4 meters long and 1 meter wide stage where several obstacles, a 4cm high sill, a staircase with 4 steps each 3cm high, and a higher-level platform were arranged as shown in Figure 5.

We placed QRIO at the left end of the stage and let it observe the environment by rotating on the spot. We then commanded the robot to find and follow a path to a position 3 meters ahead (the center of the higher-level platform). Our path-planning system continuously evaluates the current *NAV*

Figure 5: Experimental setup. Several obstacles and a sill are placed on a stage. A staircase leads to a higher-level platform.

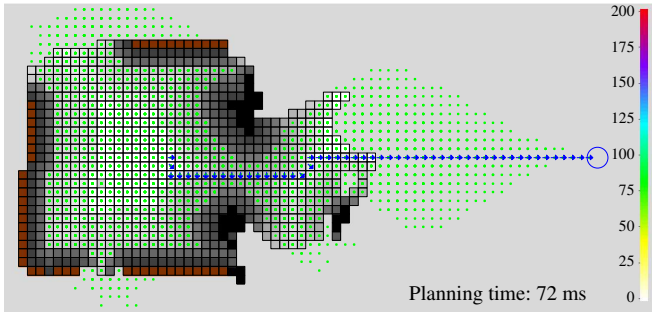


Figure 6: Path-planning in narrow environment: the robot decides to use sideways walk for passing through the obstacles.

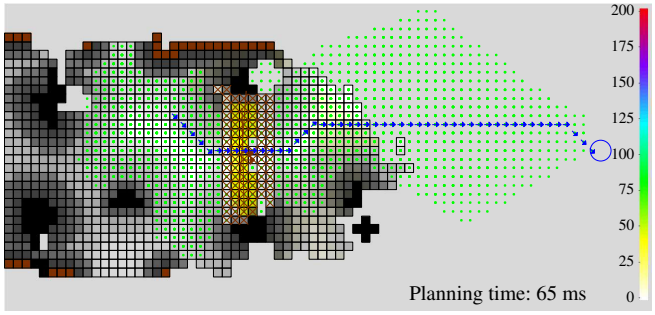


Figure 7: Path-planning over a sill: the robot finds a straight path over the sill and then moves around the next obstacle.

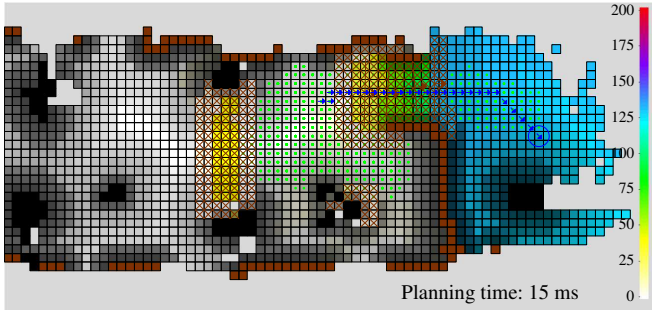


Figure 8: Path-planning to the higher platform: first the robot aligns with the staircase and then climbs up in a straight way.

grid and sends paths to a *path-following* system which computes the next two foot steps and feeds them to the robot's motion control system. Figures 6–8 show 3 snapshots of the robot while moving on the stage. We use the same encoding for floor, border, stairs and obstacles as before. The color indicator on the right shows the floor height as estimated by the robot. Unknown space is left blank. Green dots mark the search space of the A* algorithm while the path is shown with blue arrows. Brown bars on the path indicate stair transitions.

In the beginning the system has to find a path leading between two close obstacles (see Figure 6). Since space between the obstacles is narrow, forward walk is not possible due to the lack of the required clearance $r_{forward}$. Therefore, the system decides to walk sideways between the obstacles.

After passing through the narrow passage, the robot plans a path over the sill in the center of the stage (see Figure 7). For overcoming the sill, the system employs a separately developed *stair-climbing* module [Gutmann *et al.*, 2004].

Finally, as the robot walks closer to the higher platform, it finds a path on the staircase to the target location (Figure 8).

The time required for finding a path on our fixed-size configuration space depends on the environment and the path length. In the examples above the time for planning was always below 100 ms on the robot's embedded CPU. In order to investigate the run-time dependency on path length we randomly selected goal configurations in the final map after the robot reached its target. If a path to the goal could be found, we recorded the length of the path and the time needed for computation. Figure 9 shows the runtime versus path length measured on a Pentium III, 1.4GHz computer. Although the curve shows a slope at least quadratic in path length, for our chosen dimensions of C the runtime increases only slowly and does not exceed 40 ms even for longer paths.

The worst case run-time occurs when the goal is placed at an unreachable location, e.g. a free position completely surrounded by obstacles. In this case all but a few of the $8n^2$ = 80,000 configurations have to be examined. The robot's embedded processor needs close to 900 ms for exploring this search space (110 ms on a Pentium III, 1.4GHz).

6 Conclusion

We presented and evaluated a path-planning algorithm that provides collision free trajectories of up to a few meters for a humanoid robot in a layered environment in real time.

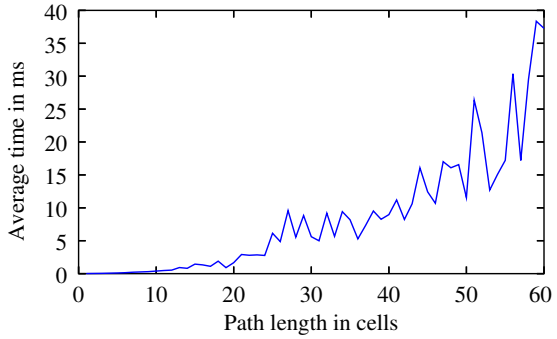


Figure 9: Time for path-planning depending on path length.

In our approach unknown terrain is treated in an optimistic way, i.e. their clearing is assumed arbitrary large. As the robot moves closer to unknown area and observes the situation, the navigation map is refined and a new path is planned automatically. This allows navigation in unknown environments.

Our perception system distinguishes between floor, border, stairs and obstacles and interprets them in a meaningful way for path planning. We believe our approach advances previous methods on simulated worlds, and real robot systems with mainly reactive, short distance navigation capabilities.

The limitations of our approach are various. Our discretization onto a grid is an approximate cell decomposition and as such is only complete up to the resolution of the grid. Furthermore, using only 8 orientations can prevent finding paths in certain environments, for example in a narrow corridor oriented at an angle of 20°.

Our approach may be extended to 2^m orientations $(m > 3)$ by changing cell size and actions such that repeated application of a turn action describes a curve with the minimum turn radius of the robot. Other actions might move the robot over more than one cell in order to align with the grid.

Our set of actions and the navigation map do not consider the possibility of stepping over an obstacle. We believe that for such a capability, a foot step planner using precise 3D information is of importance. However, our approach could be combined with such a system by marking obstacles the robot can step over similar to as we mark stairs.

At current, only straight paths are allowed on stairs. By introducing new actions *left-forward* and *right-forward* we can also find paths on spiral staircases at the cost of a higher runtime. In practice, we observed an increase of about 25% in run-time for the worst case scenario described in Section 5 when adding these two actions.

We think it is also possible to extend our approach to find areas where the robot can crawl underneath an obstacle. Our current floor and obstacle map and the navigation grid, however, do not support such a detection yet and have to be extended first. We will follow this direction in the future.

Acknowledgment

We would like to thank Stefan Edelkamp, Thilo Weigel and the anonymous reviewers for their valuable comments. We also thank all developers of the Sony QRIO robot for making this work possible.

References

[Buhmann *et al.*, 1995] J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The mobile robot RHINO. *AI Magazine*, 16(2):31–38, 1995.

[Chestnutt and Kuffner, 2004] J. Chestnutt and J.J. Kuffner. A tiered planning strategy for biped navigation. In *Int. Conf. on Humanoid Robotics (Humanoids)*, 2004.

[Fujita *et al.*, 2003] M. Fujita, Y. Kuroki, T. Ishida, and T.T. Doi. A small humanoid robot SDR-4X for entertainment applications. In *Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, Kobe, Japan, 2003.

[Guan *et al.*, 2004] Y. Guan, K. Yokoi, N.E. Sian, and K. Tanie. Feasibility of humanoid robots stepping over obstacles. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[Gutmann *et al.*, 2004] J.-S. Gutmann, M. Fukuchi, and M. Fujita. Stair climbing for humanoid robots using stereo vision. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[Gutmann *et al.*, 2005] J.-S. Gutmann, M. Fukuchi, and M. Fujita. A floor and obstacle height map for 3D navigation of a humanoid robot. In *Int. Conf. on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[Hirai *et al.*, 1998] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of Honda humanoid robot. In *Int. Conf. on Robotics and Automation (ICRA)*, 1998.

[Kanehiro *et al.*, 2004] F. Kanehiro, H. Hirukawa, K. Kaneko, S. Kajita, K. Fujiwara, K. Harada, and K. Yokoi. Loclomotion planning of humanoid robots to pass through narrow spaces. In *Int. Conf. on Robotics and Automation (ICRA)*, New Orleans, 2004.

[Kavraki *et al.*, 1996] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), 1996.

[Kuffner *et al.*, 2002] J.J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

[Latombe, 1991] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[Li *et al.*, 2003] T.-Y. Li, P.-F. Chen, and P.-Z. Huang. Motion planning for humanoid walking in a layered environment. In *Int. Conf. Robotics and Automation (ICRA)*, 2003.

[Lorch *et al.*, 2002] O. Lorch, A. Albert, J. Denk, M. Gerecke, R. Cupec, J. F. Seara, W. Gerth, and G. Schmidt. Experiments in vision-guided biped walking. In *Int. Conf. Intelligent Robots and Systems (IROS)*, 2002.

[Okada *et al.*, 2003] K. Okada, M. Inaba, and H. Inoue. Walking navigation system of humanoid robot using stereo vision based floor recognition and path planning with multi-layered body image. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, October 2003.

[Sabe *et al.*, 2004] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *Int. Conf. on Robotics and Automation (ICRA)*, New Orleans, 2004.

[Shiller *et al.*, 2001] Z. Shiller, K. Yamane, and Y. Nakamura. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. In *Int. Conf. on Robotics and Automation (ICRA)*, Seoul, Korea, 2001.

[Stentz, 1995] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. *Int. Journal of Robotics and Automation*, 10(3), 1995.