32

Planning Versatile Motions for Humanoid in a Complex Environment

Tsai-Yen Li, Pei-Zhi Huang
Computer Science Department, National Chengchi University
Taipei, Taiwan

1. Introduction

The potential market of service and entertainment humanoid robots has attracted a great amount of research interests in the recent years. Several models of humanoid robots have been designed in research projects. However, it remains a great challenge to make humanoid robots move autonomously. Motion planning is one of the key capabilities that an autonomous robot should have. An autonomous robot should be able to accept high-level commands and move in a real-life environment without colliding with environmental obstacles. A high-level command is something like "Move to location A on the second floor" while the robot is currently at some location B on the first floor, for example. It is an interesting problem for humanoid robots since locomotion capability possessed by a humanoid robot is usually much better than a mobile robot. Like a human, a humanoid robot should be able to step upstairs or downstairs and stride over small obstacles or a thin deep gap in a complex environment such as the one shown in Fig. 1. Similar needs for the planning capability also arise in the domain of computer animation in generating motions for autonomous characters.

The motion for a humanoid robot to achieve a given goal is typically very complex because of the degrees of freedom involved and the contact constraint that needs to be maintained. Therefore, it is common to take a two-level planning approach to solve this problem. In our previous work (Li et al, 2003), we have been able to plan efficient humanoid walking motions in a layered environment. The approach used in the planner decomposed the planning problem into subproblems of global and local planning, each of which is easier to solve. The global planner assumes some basic properties of a humanoid and uses an approximated geometric shape to define the path planning problem. The path generated by the global planner is then passed to the local planner to realize the path with appropriate walk motions. However, in previous work, the locomotion of a humanoid is usually limited to forward walking only, and a humanoid cannot pass a deep gap even if it can stride over it. In this chapter, we will describe a motion planning system adopting the two-level approach and extending the work to overcome the above two limitations. We will present an efficient implementation of the planner in terms of space and time such that it can be used in an on-line manner.

The rest of the chapter will be organized as follows. In the next section, we will review the researches pertaining to our work. In the third section, we will give a formal description of the problem we consider in this chapter. We will then propose the planning algorithm in the

fourth section and present some implementation details and experimental results in the fifth section. In the last section, we will conclude the chapter with some future directions.

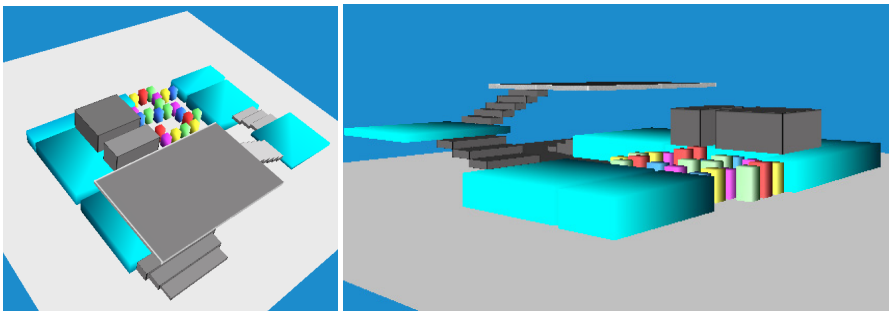


Fig.1. Example of a layered virtual environment with deep narrow gaps (colored columns between two large platforms)

2. Related Work

Motion planning problems have been studied for more than three decades. An overview of motion planning algorithms can be found in Latombe's book (Latombe, 1991). The researches pertaining to humanoid motion planning or simulation can be found in the fields of robotics and computer animation (Kuffner et al., 2001; Polland et al., 2002; Sun & Dimitris, 2001). These researches differ mainly on the way they perform global path planning and how locomotion is taken into account. For example, early work in computer animation focused on generating human walking motion to achieve a high-level goal (Bruderlin & Calvert, 1989). However, no planning was done to generate the global path automatically. Since the application was mainly for computer graphics, how to simulate human walking with realistic looking was the main concern (Sun & Dimitris, 2001). A dynamic filtering algorithm was proposed in (Yamane & Nakamura, 2000) to ensure that the generated motions could be transformed into a dynamically feasible one.

In an early paper by Kuffner (1998), a gross motion planner utilizing graphics hardware was proposed to generate humanoid body motions on a flat ground in real time. Captured locomotion was used in this case to move the humanoid along the generated global path. In (Kalisiak & Panne, 2001), a stochastic search approach with versatile locomotion was proposed to generate humanoid motions in an unstructured environment where a set of predefined grasp points served as contact constraints. In (Choi et al, 2003), sequences of valid footprints were searched through augmented probabilistic roadmap with a posture transition graph, and then each pairs of footprints were substituted by corresponding motion clips. Since the motion clips were captured in advance, the locomotion may not be flexibly adapted to uneven terrain to avoid collisions. In (Shiller et al, 2002), a multi-layer grid was used to represent the configuration space for a humanoid with different types of locomotion such as walking and crawling, and the humanoid may change its posture along the global path.

In (Pettre et al., 2003), a digital actor was modelled with active (all degrees of freedom attached to the legs) and reactive (attached to the upper parts of the body) degrees of freedom. A collision-free trajectory was computed for the active part of the digital actor. Then the motion warping technique (Witkin & Poppvic, 1995) was applied to the motion

capture data along the path when the reactive part of the digital actor collided with obstacles. The planner proposed in (Chestnutt et al., 2003) evaluated footstep locations for viability using a collection of heuristic metrics of relative safety, effort required, and overall motion complexity. At each iteration of the search loop, a feasible footstep was selected from the footstep transition sets. The global path of this approach was a sequence of footstep locations toward a given goal state.
In (Kuffner et al., 2001a; Kuffner et al., 2001b), a humanoid robot with real-time vision and collision detection abilities was presented. The robot could plan its footsteps amongst obstacles but could not step onto them. Considering locomotion directly in global path planning may generate more complete result but, on the other hand, it limits the flexibility of locomotion. In our previous work (Li et al., 2003), we were able to plan humanoid motions in real time with a given general description of the objects in the workspace. A more detailed description of this approach will be given in the next two sections.

3. Problem Description

3.1 Problem Overview

According to motion granularity, the motion-planning problem usually can be classified into global (gross) motion planning and local (fine) motion planning. In the global motion-planning problem we concern with working out the body logistics, such as planning a collision-free path amongst trees in a forest to reach some destination. In contrast, for the local motion-planning problem we focus more on limb logistics, such as planning hand motion to grab an awkwardly placed object. For the problem of walking on a layered complex environment for a humanoid robot, both types of planning needs to be considered in order to ensure that the desired task can be accomplished with feasible motion plans.
The inputs to a typical motion planner for a humanoid robot include the initial and goal configurations of the robot, the kinematics description and locomotion abilities of the robot, and a geometric description of objects in the environment. In our approach, the planning problem is decomposed into two subproblems: the global motion planning for moving the body trunk of a humanoid and the local motion planning for realizing the global motion plan with the chosen locomotion. The planners at the two levels can be linked together to solve the problem in sequence as well as to feed back failure situation for further replanning as shown in Fig. 2. The global motion planning will be the main concern of this chapter.

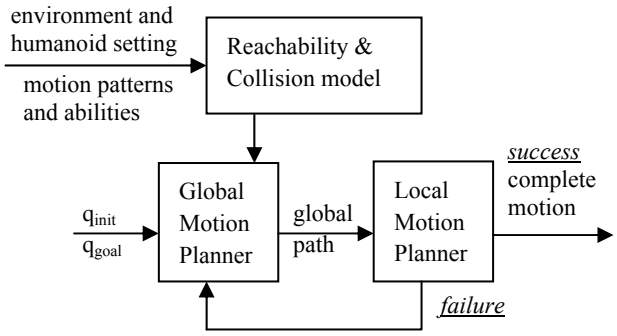


Fig.2. Planning loop for a typical query of humanoid motion.

3.2 Modeling the Humanoid

The kinematics description of a humanoid robot includes the maximal gait size, the maximal step height (h), and the bounding cylinder for each type of locomotion that the robot can adopt. Unlike traditional motion planning problems where the obstacles are given explicitly, the definition of obstacles in our case depends on the kinematics properties of the humanoid as well as the geometry of the objects. For example, an object is an obstacle to a humanoid if there is no way for the humanoid to step onto or pass under the object for a given locomotion due to the height and leg length of the humanoid.

In the global motion planning for a humanoid, we assume that the humanoid can be modeled as a bounding cylinder to simplify collision detection. In our system, we uses a large and a small cylinders to model a humanoid for frontal and lateral walking, respectively, as shown in Fig. 3. The radius of the inner cylinder is the size of the minimal region for a stable stance. The radius of the outer cylinder is determined according to the type of locomotion. We use the largest lateral width orthogonal to the moving direction of the humanoid to determine the radius. For example, the cylinder used in planning the side-walk motion is smaller than the one for regular walking motion. The height of the cylinders, denoted by H, is also related to the height and locomotion type of the humanoid. If we allow the humanoid to bend its upper body during the walking cycle, the actual height may be lower than the height of the humanoid. When using cylinders to model the geometry of a humanoid, we actually ignore its orientation at the planning time. We assume that we can recover the orientation of a humanoid in a postprocessig step according to the generated path and the adopted locomotion.

We assume that a humanoid robot can perform several types of locomotion and choose the most appropriate one according to the environment. Although many types of locomotion can be considered and implemented, only frontal walking, side walking, and jumping are demonstrated in this work. In addition, we assume that the local planner can generate necessary motion transition from one type of locomotion to another at a given configuration along a path.

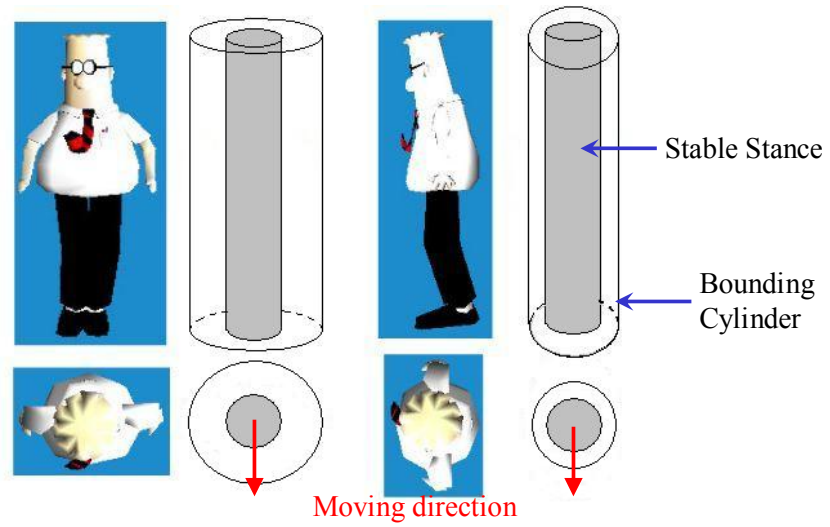


Fig.3. Frontal and lateral walking model for the humanoid.

4. Motion Planning for Humanoid

As described in the previous section, we use a decoupled approach to reduce the complexity of the motion-planning problem. In this section, we will focus on addressing the global path-planning problem for a humanoid with multiple types of locomotion including striding. We will first describe how we model the environment and prepare the search space according to the given environmental description and the data of a humanoid. Then we will present the planning algorithm that is used to search for a feasible path with these abilities.

4.1 Modeling the Environment

We assume that we are given the geometric description of the objects in the workspace such as the one shown in Fig. 4(a) and 4(b). Each object is described by a height and an offset from a reference level (such as the ground) in addition to its polygonal description. According to this geometric description, one can discretize the workspace into a grid of cells with an appropriate resolution. The resolution is chosen such that the area of a cell can allow a foot of the humanoid to step onto. Each cell in the workspace is given an offset value representing the distance from the bottom of an object to the referenced ground. Each cell is also assigned a height value above the offset. According to the offset values of the objects, we separate the 3D workspace into multiple connected 2D layers. For each layer, we compute a height map containing the elevation value of each cell above the layer in the workspace grid as shown in Fig. 4(c) and 4(d). Each cell in the layer l is referred to the same offset value d^l representing the distance from the base of the layer l to the referenced ground. Each cell i is also assigned a height value α_i^l above the layer l. The cell i of the height map for the layer l is represented as $c_i^l = \alpha_i^l$. Therefore, in a layered environment, a point may have different offset and height values on these different layers.

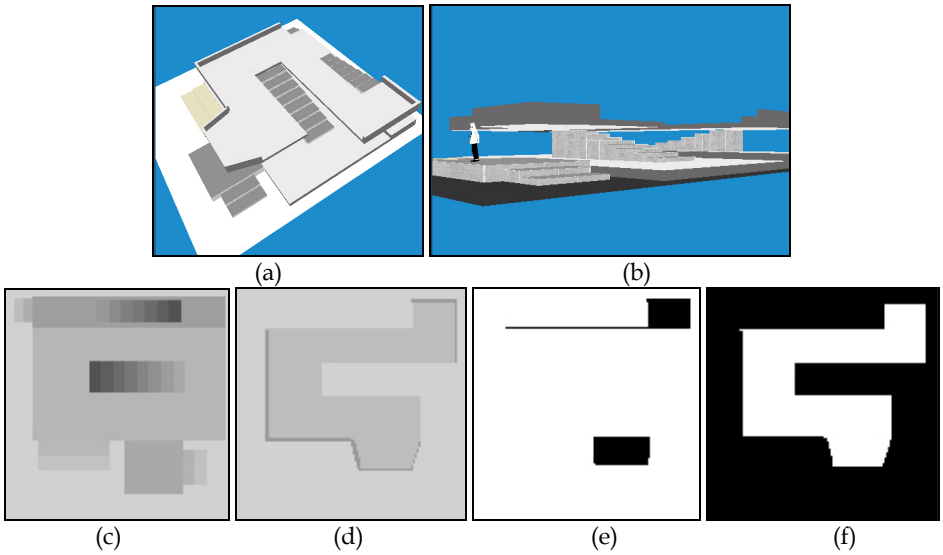


Fig. 4. (a) Top view of the workspace, (b) side view of the workspace, (c) and (d) are the height maps (the darker, the higher), (e) and (f) are the reachability maps of first and second layers (reachable regions in white).

The layered representation approach is appropriate for the gross motion-planning problem of a humanoid robot since the humanoid needs to stay on the ground all the time, even when climbing up or stepping down stairs. Therefore, a configuration q in workspace can be represented as (x, y, l), which means that the humanoid is standing at position (x, y) and on the "ground" of layer l.

4.2 Basic Reachable Region

Given an initial configuration of the humanoid, we compute a map, called *reachability map*, to represent the reachable regions from the initial configuration. The unreachable cells are marked as obstacle regions in this map. A reachability map may consist of several slices with one slice for each layer. For example, Fig. 4(e) and 4(f) show the reachability map for the two layers of the scene in Fig. 4(a). The black regions are the unreachable regions marked as obstacles. These slices could be "connected" if there exists an object whose height is large enough to bring the humanoid to step onto some neighboring cells of the above or below layer. We compute this map by a wave propagation algorithm similar to the one used to construct NF1 potential fields (Latombe, 1991). Suppose that the current cell under propagation is i. The algorithm advances to a neighboring cell i' at the layer l or a neighboring layer l' only if the height difference between them is less than h ($\left|c_i^l - c_{i'}^l\right| < h$ or $\left|(d^l + c_i^l) - (d^{l'} + c_{i'}^{l'})\right| < h$) and the humanoid height H can fit into the clearance above the cell i' at the layer l or l' ($\left|(d^{l+} - d^l - c_{i'}^l)\right| > H$ or $\left|(d^{l'+} - d^{l'} - c_{i'}^{l'})\right| > H$, where l+ denotes the layer above l).

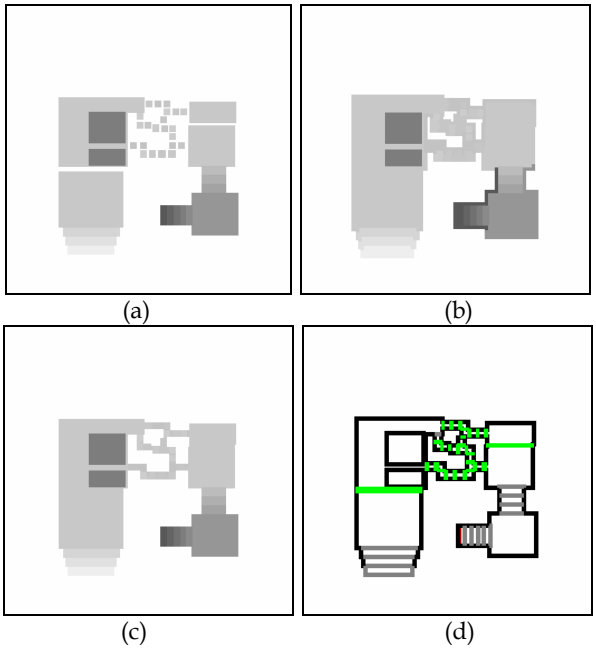


Fig.5. (a) height map for the environment in Fig. 1., (b) map in (a) after applying dilation, (c) map in (b) after applying erosion. After the closing operator, the columns in the discontinuous region are connected. (d) instability map

4.3 Reachable Region for Striding Ability

For a humanoid with versatile motion abilities, the reachable region in the workspace is affected by several factors. For the walking motion, we need to take the maximal gait size and leg length of the humanoid as well as the height of the bounding cylinder into account. In the previous subsection, we have described a method to compute the reachability map for an environment with several layers. However, for an environment with deep narrow gaps such as the one shown in Fig. 1, large height differences between neighbouring cells prevent a humanoid to move across the region occupied by the columns. The main difficulty comes from the fact that we only check the neighbours that are one cell away although the maximal gait size of the humanoid could occupy several cells.

A straightforward approach to overcome this problem is extending the search algorithm to visit all the neighbours that are a few cells away from the current configuration and take the legal ones for further processing. An obvious drawback of this approach is that the number of neighbours increases rapidly as the gait size increases. The planning time will suffer as the number of visited cells soars. In addition, if we do not have a good idea about the reachable region, it would be difficult to compute an effective potential field to guild the search.

In this work, we propose a method for reachability computation that can account for the striding ability of a humanoid to handle the discontinuous regions mentioned above. The idea is that we can connect together the separated regions within the gait range by using the "Closing" operator, an important morphological operator in image processing (Gonzale & Woods, 2002). Closing tends to smooth sections of the image. It generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the image. Closing of set A by a structuring element B, denoted $A \bullet B$, is defined as:

$$A \bullet B = (A \oplus B) \Theta B \tag{1}$$

The equation, in words, says that the closing of A by B is simply the dilation of A by B, followed by the erosion of the result by B. The height map is A in our case, and we select the circle with a radius of half of gait size as the structuring element B. The dilation operation ($A \oplus B$) will extend the edge of objects in bitmap by $1/2$ gait size, which fills small gaps or holes between two disconnect regions if these regions are near. After dilation, the resulting bitmap shows that two regions are connected if their distance is within one foot step (See Fig. 5(b)). But for the extended edges which do not connect to other regions, we apply erosion ($A \Theta B$) to eliminate these useless edges. The erosion operation will corrode the edges in bitmap by $1/2$ gait size, which dose not break the connected regions. Fig. 5(c) shows the result after erosion. Note that the unnecessary edges are removed after the operation and the map gets back to its original shape as in Fig. 5(a) except for the small discontinuous regions that are connected now. This modified height map after the closing operation is then used to determine the reachability of each cell for the walking motion with the method described in the previous subsection.

Since we have more than one type of locomotion available for the humanoid to use, we will need to compute a reachability map for each type of locomotion according to the parameters of the humanoid such as the step size and the size of the bounding cylinder for that type of locomotion. We can then take the union of these reachability maps to form the final reachability map. In other words, a cell is reachable if there exists at least one type of locomotion that can bring the humanoid to the cell. We record the available locomotion types for each cell such that we know how to choose an appropriate motion later in the search process.

4.4 Instability Map

In order to know which part of the region is collision-free, we can convert the height map, built in the workspace, into the corresponding C-space by growing the obstacle regions with the radius of the bounding cylinder of the humanoid for a given locomotion. However, not all collision-free regions allow the humanoid to stay for long. For example, we may allow the humanoid to walk on a stair for trespassing purposes but do not want it to stay at the edge of a stair for too long. We need to identify these regions where unstable situation might occur. A map describing the regions is called *instability map* (see Fig. 5(d) for an example). A cell in the instability map is defined as unstable if and only if the region covered by the enclosing circle of a humanoid contains cells with different heights and this height difference is smaller than the maximal step height of the humanoid. Otherwise, if this region contains an object whose height difference is larger than the maximal step height of the humanoid, we will consider the cell a *forbidden* cell. Similarly, a cell is defined as *gap* if and only if the region in height map is reachable after the closing operation. A humanoid can enter the unstable or gap regions for trespassing purpose but the transition time usually needs to be short. Therefore, we have set an upper bound for each type of region in the search process to prevent the humanoid from staying in these regions for too long.

```
STABLE_BFP()
1   install q_i in T;
2   INSERT(q_i, OPEN); mark q_i visited;
3   SUCCESS ← false;
4   while ¬ EMPTY(OPEN) and ¬ SUCCESS do
5       q←FIRST(OPEN);
6       for every neighbor q′ of q in the grid do
7           if q′ is stable then
8               mark q′ visited;
9           if LEGAL(q′, q) then
10            install q′ in T with a pointer toward q;
11            INSERT(q′, OPEN);
12            if q′ = q_g then SUCCESS ← true;
13  if SUCCESS then
14      return the backtracked feasible path
15  else return failure;
```

Fig.6. The STABLE_BFP algorithm

4.5 Path Planning Algorithm

The planning algorithm that we use to compute the humanoid motion is shown Fig. 6. The STABLE_BFP algorithm is similar to the classical Best-First Planning algorithm (Barraquand & Latombe, 1991) that was used to solve path-planning problems with low DOF's. In each iteration of the search loop, we use the FIRST operation to select the most promising configuration q from the list of candidates (*OPEN*) for further exploration. We visit each neighbor q' of q, check their validity (via the LEGAL operation) and maintain their stability (through STABLE operation) for further consideration. The number of neighbors is related to the number of major locomotion, n. That is, if we visited m neighbors for each locomotion, we have to visited $m*n$ neighbors in each iteration. The procedures for checking stability and legality are shown in Fig. 7. A configuration is legal if it is collision-free, marked unvisited,

and temporarily stable. It is temporarily stable if and only if the humanoid has not entered the unstable regions or the gap regions for longer than some threshold. This duration is kept as an instability counter (*q′.cnt*) in each cell in the unstable region and step counter (*q′.step*) in the gap region when we propagate nodes into it. Note that the validity of a configuration in the unstable or gap regions depends on the counter of its parent configuration. If there are more than one possible parent configurations, we cannot exclude any of them. Therefore, in the STABLE_BFP algorithm, we do not mark a configuration visited if it is in the unstable or gap region. A configuration in these regions can be visited multiple times as long as the counter does not exceed the maximal bound. In a gap region, we need to check the gap width with the gait size. We keep the start point of a gap (*gap_begin*) when we first enter the gap region and the end point of a gap (*gap_end*) when a stable region is reached. These two points are used to count the gap width, and the procedure is to ensure that the gap does not exceed the gait size and the humanoid can stride from *gap_begin* to *gap_end*.

```
STABLE (q′ ,q)
    if q′ is unstable then
       q′.cnt= q.cnt+1;
    else if q′ is gap then
       if q.ingap then
           q′.step= q.step+1;
       else
           q′.step=1;
           q′.ingap=true;
           gap_begin= q;
    else if q′ is stable and q.ingap then
       gap_end= q′;
```

```
LEGAL (q′ ,q)
    STABLE(q′ ,q)
    if q′ is visited or forbidden then
       return false;
    if q′ is gap and q′.step>N then
       return false;
    if q′ is unstable and q′.cnt>M then
       return false;
    if q′ is stable and q.ingap then
      check distance between gap_begin and gap_end;
      if the distance is large than gait size, then
       return false;
    return true;
```

Fig. 7. The STABLE and LEGAL procedures.

In the STABLE_BFP algorithm, we use the FIRST operation to select the most promising configuration for further exploration. In general BFP planners, the artificial potential field is usually the only index for the goodness criterium. Planners with this approach can usually yield short paths. In our case, the height difference could be an important index as well since one may prefer climbing up or stepping down stairs to taking a longer path. Preference on each available locomotion is also an important factor. For example, generally speaking, we prefer walking to crawling. Therefore, in the FIRST operation, we use a linear combination

of these criteria, whose weights are specified by the user. In general, unstable regions and gap regions have lower preference, and staying in these regions is not preferred. Therefore, we use instability (*q.cnt*) or step counter (*q.step*) as a penalty measurement to avoid motions over difficult areas whenever possible.

5. Implementation and Experiments

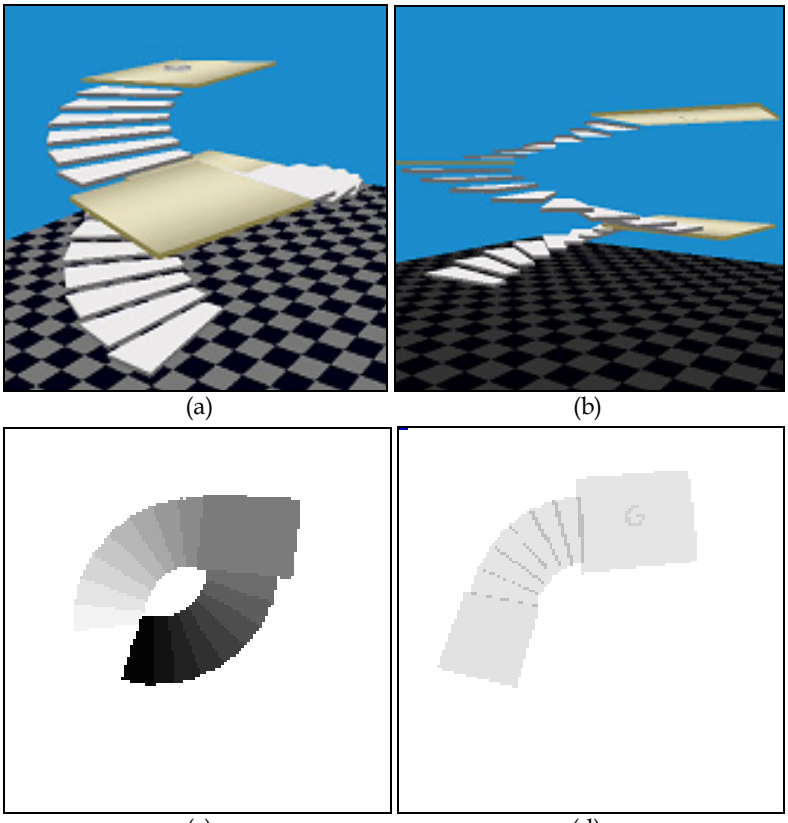
We have implemented the global and local motion planners in Java and connected the planners to a VRML browser to display the final simulation results. All the planning times reported below are taken from experiments run on a regular 1.6GHz PC. The size of the workspace is 25.6m x 25.6m, and the height and gait size of the humanoid is 180cm and 60cm, respectively. The width of the shoulder (for enclosing cylinder) and the foot length is 60cm and 37cm, respectively. The resolutions for the grid workspace and configuration space are all 256x256 in the global planner. In the following subsection, we will describe how to optimize storage space by merging several layers with sparse objects in them to fewer layers. We will also use two examples to demonstrate the ability and efficiency of the planner in later subsections.

5.1 Merging Layers with Sparse Objects

In the proposed system, we use the offset value of an object to separate the workspace into several 2D grid layers. However, for workspace like gyratory stairs, such as the one shown in Fig. 8, each stair may have a unique offset value, and we need a layer for each offset. The result is that each of the layered maps only contains sparse objects. As the number of layers increases, not only the storage space will increase, but the search performance will degrade as well. In our implementation, we first sort the objects in workspace by their offsets in ascending order. Then we add objects into the first layer one by one until an object overlapped with other objects in this layer. A new layer is then created on demand. After this process, each object is assigned to a specified layer, and the resulting number of merged layers is usually much smaller. Fig. 8 is the example of gyratory stairs, which need 25 layers if we use offsets to separate the workspace directly. If we use the merge approach mentioned above, only two layers are needed to represent the workspace.

5.2 Example

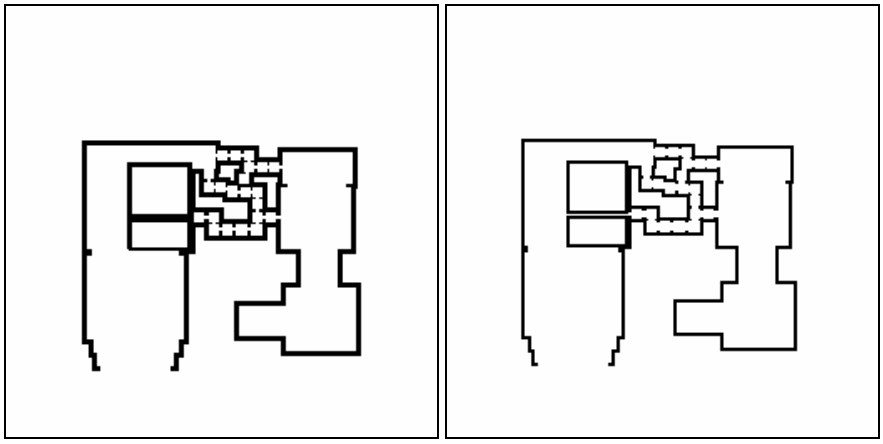
In Fig. 9, we use an example to illustrate the features of this planning system. The environment, which is the same as the one in Fig. 1, consists of two layers of objects with various sizes and heights scattered on the two main platforms, connected by several columns of various heights. The C-obstacles (configuration space obstacles) with different bounding cylinders for forward walking and side walking are shown in Fig. 9 (a) and (b), respectively. Note that the narrow passage exists only when side-walk locomotion (Fig. 9. (b)) is used. The global path found by the planner is shown in Fig. 9 (c) and (d). In this example, the initial configuration is on the ground, and the goal is at some location on the second layer that is reachable only through the following passages: climbing upstairs to the left platform, changing locomotion to side walk to pass the narrow passage, crossing the columns to the right platform, and climbing upstairs again to the second layer on the right platform. The planning time for a typical run consists of two parts: preprocessing and search. In this example, the total preprocessing time is 251ms (constructing layer map takes 10ms, computing the reachable region takes 35ms, computing the potential field takes 74ms, and computing the instability map takes 132ms) and the search time is 15ms.



(a) (b)

(c) (d)

Fig. 8. Merging sparse layers. (a) and (b) are the 3D workspace with spiral stairs from different views (c) and (d) are the merged results for layer 1 and layer 2, respectively.



(a) (b)

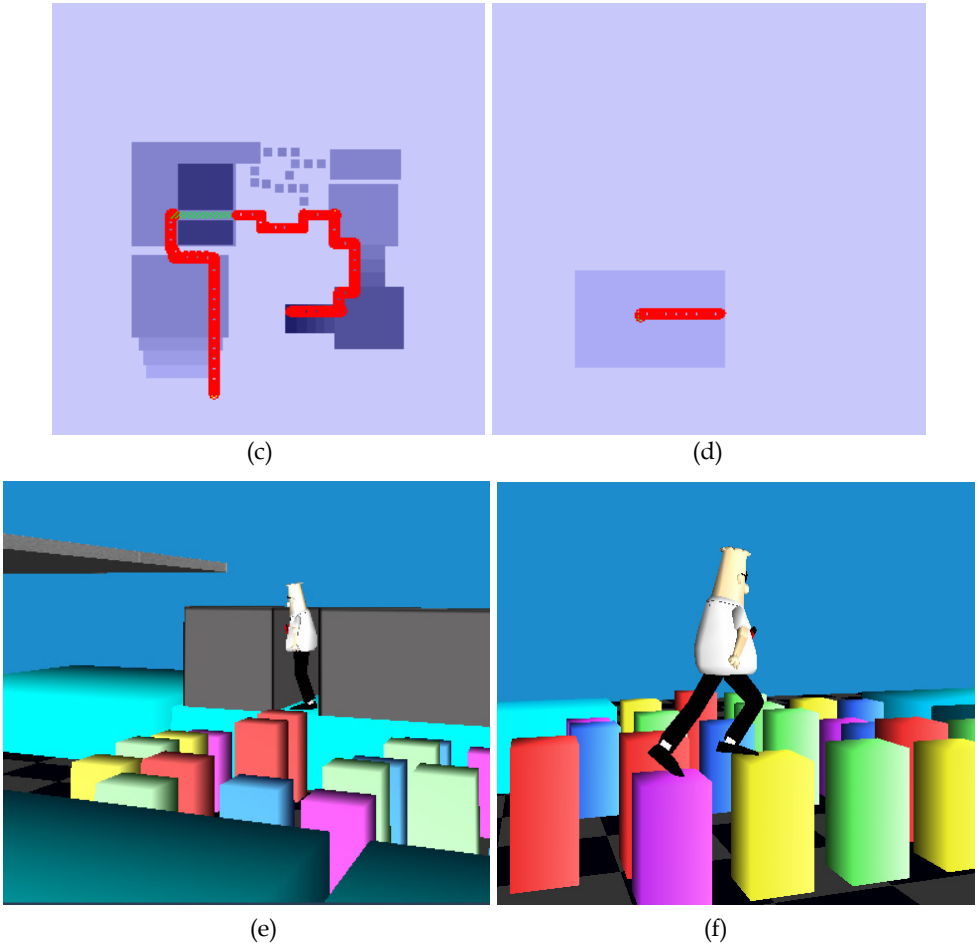


Fig. 9. (a) the C-obstacles for forward walking and (b) side walking. (c) the search result of the global path planner on layer 1 and (d) layer 2, (e) passing the narrow passage with side-walk motion, (c) feasible local motions generated to walk across the columns.

In Fig. 10, we use an outdoor environment to illustrate the use of several types of locomotion. The environment, as shown in Fig. 10(a), is partitioned into three blocks by the river. Upper and lower blocks are connected through the lily pads floating on the river. The broken bridge between the left and the right block is the shortest path between them. However, the distance is too large for the humanoid to stride over. In this case, the humanoid can only use jumping to move over the broken bridge. In addition, the passage to the house is too narrow for the humanoid to walk through without switching to the side-walk locomotion because the narrow passage exists only when side-walk locomotion is used. The global path found by the planner is shown in Fig. 10 (b), the red portion of the path is frontal walking, yellow portion is jumping, and green portion is lateral walking. In this example, the initial configuration is on the left

bottom of the lower block, and the goal is in the yard of the house at the center of the right block that is reachable only through the following passages: climbing upstairs to the platform, detouring round the woods in the forest (Fig. 10(c)), striding over the lily pads (Fig. 10(d)), jumping over the broken bridge between the left and the right block (Fig. 10(e)), changing locomotion to side walk to pass the narrow fence (Fig. 10(f)), and finally reach the goal. The total planning time is 312ms (281ms for preprocessing and 31ms for path searching).

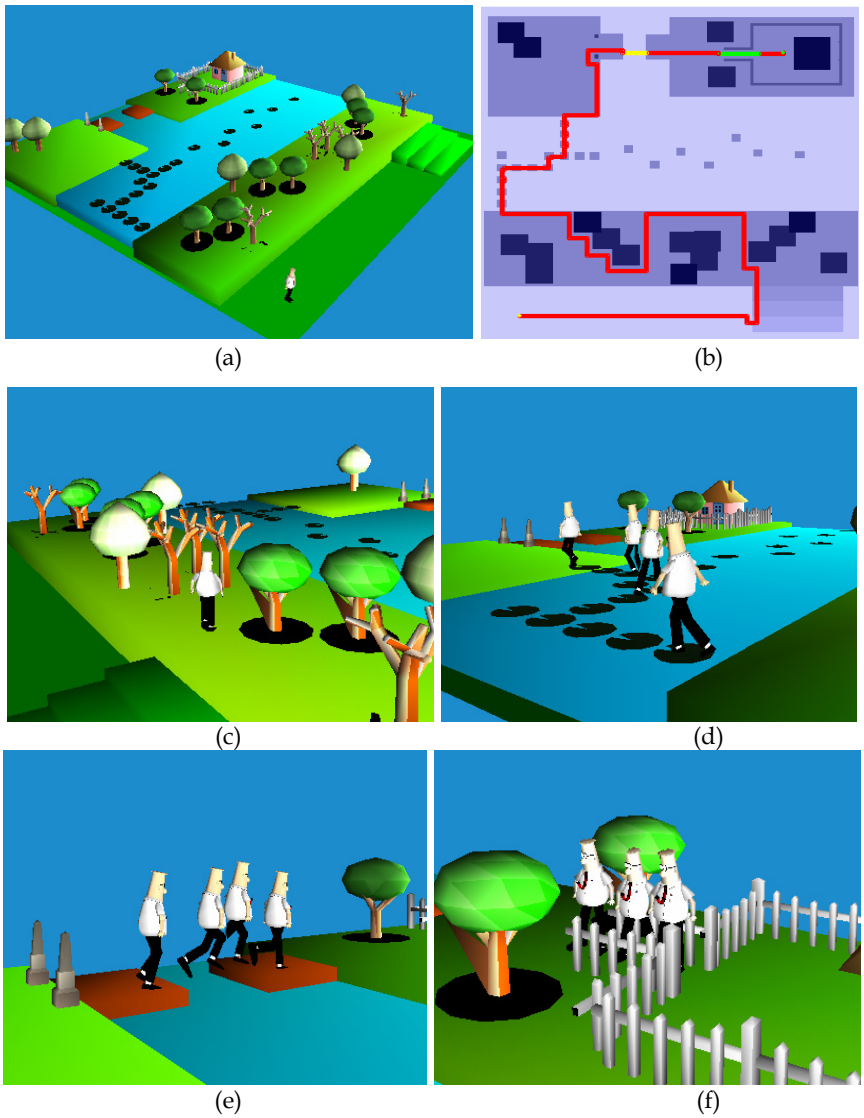


Fig. 10. An outdoor environment requiring the humanoid to use several motion skills to reach the goal.

5.3 Planning Performance Compared to the Traditional Approach

As mentioned in previous section, traditional straightforward search algorithms may also be able to find paths with the striding ability by extending the range of the neighbor search. Compared to this traditional approach, our approach is superior in performance and extensibility. Assume that a normal walking gait size is 60cm, which is equal to 6 cells in search space with the current resolution. In the traditional approach, the number of neighbors that needs to be covered is 64 for each configuration. This number could be even larger if we allow the humanoid to have a larger gait size or if we use finer resolution for the search space. The search time for an example similar to the one shown in Fig. 9 is 1246ms with the traditional approach where 64 neighbors are visited at a time and totally 47952 nodes are visited. With our approach, the preprocessing takes 263ms, and the search takes only 34ms. The total number of visited nodes is 2612. The proposed planning approach is faster than the traditional approach by about an order of magnitude. In addition to the reduced number of visited neighbors, the new approach is fast also because the potential field computed with a correct reachability map is more effective. The traditional approach can only use the distance to the goal as a heuristic since the reachable regions in the reachability map are discontinuous.

6. Conclusions and Future Work

Building autonomous humanoid robots has been the goal of many applications in robotics and computer animation. Spatial reasoning is a key capability to enable a robot to accept high-level commands and move autonomously. In this chapter, we have described a motion-planning system that can generate feasible humanoid animations with versatile locomotion in a complex virtual environment. The focus has been put on designing a global path planner that can generate a gross motion plan respecting the environmental and the humanoid constraints. The reachability can be defined by the properties of humanoid and can be extended through locomotion abilities. With the new problem definition, we modify the Best-First planning algorithm to account for user preference on horizontal or vertical distances that a humanoid travels. We have extended the planning system to consider such abilities and demonstrated the efficiency and effectiveness of the planner by simulation examples. We believe that this work will inspire further studies on the interesting problem of computing humanoid motion automatically for movie generation or autonomous humanoid robots.

In the global planner, collision detections for transition between different types of locomotion are done by checking the potential collisions with both types of motions. In fact, this assumption has over-simplified the transition problem since the bounding cylinder of a transition motion could be larger than either motion at both ends. Besides, not every transition between different types of locomotion is possible. In (Choi et al., 2003), a predefined transition graph is used to check whether the transition between different locomotion is valid or not. In the future, we would like to construct a transition graph; every node is a type of locomotion and the edge between two nodes exists only if the transition is possible. Corresponding models for collision checks should be attached to the edges to ensure that the global planner can properly check the collisions.

A real human usually can avoid obstacles with various kinds of locomotion and body motions. In the future, we would also like to enable the humanoid robot with more locomotion abilities when moving to the goal. For example, a human can crawl or stoop to avoid upper-layer obstacles. In addition, in a complex environment, there could exist movable objects that can be moved away by pulling or pushing by the humanoid robot to make ways for itself to reach the goal. We will also consider this kind of manipulation and reasoning capability in the future.

7. References

Bruderlin, A. & Calvert, T.W. (1989). Goal-Directed, Dynamic Animation of Human Walking, *Proceedings of ACM SIGGRAPH 1989*

Barraquand, J. & Latombe, J.C. (1991). Robot Motion Planning: A Distributed Representation Approach, Intl J. of Robotics Research, 10:628-649

Chestnutt, J.; Kuffner, J.; Nishiwaki, K. & Kagami, S. (2003) Planning Biped Navigation Strategies in Complex Environments, *Proceedings of IEEE International Conference on Humanoid Robotics*

Choi, M.G.; Lee, J. & Shin, S.Y. (2003). Planning Biped Locomotion Using Motion Capture Data and Probabilistic Roadmaps, *ACM Transactions on Graphics*, Vol. 22, No. 2, pp. 182–203

Gonzale, R.C. & Woods, R.E. (2002). *Digital Image Processing*, Second Edition, Prentice Hall

Kalisiak, M. & Panne, M. (2001). A Grasp-Based Motion Planning Algorithm for Character Animation, *Journal of Visual Computer and Animation*, pp. 117-129

Kuffner, J. (1998). Goal-Directed Navigation for Animated Characters Using Real-time Path Planning and Control, *Proceedings of CAPTECH'98 Workshop on Modeling and Motion capture Techniques for Virtual Environments*, Springer-Verlag

Kuffner, J.J.; Nishiwaki, K.; Kagami, S. Inaba, M. & Inoue, H. (2001). Footstep Planning Among Obstacles for Biped Robots, *Proceedings of 2001 IEEE International Conference on Intelligent Robots and Systems (IROS 2001)*

Kuffner, J.J.; Nishiwaki, K.; Kagami, S.; Inaba, M. & Inoue, H. (2001). Motion Planning for Humanoid Robots under Obstacle and Dynamic Balance Constraints, *Proceedings of 2001 IEEE International Conference on Robotics and Automation*

Latombe, J.C. (1991). *Robot Motion Planning*, Kluwer, Boston, MA

Li, T.Y.; Chen, P.F. & Huang, P.Z. (2003). Motion Planning for Humanoid Walking in a Layered Environment, *Proceedings of the 2003 International Conference on Robotics and Automation*

Pettre, J.; Laumond, J.P. & Simeon, T. (2003). A 2-Stage Locomotion Planner for Digital Actors, *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation*

Pollard, N.S.; Hodgins, J.K.; Riley, M.J. & Atkeson, C.G. (2002). Adapting Human Motion for the Control of a Humanoid Robot, *Proceedings of 2002 IEEE International Conference on Robotics and Automation*, pp. 2265-2270

Shiller, Z.; Yamane, K. & Nakamura, Y. Planning Motion Patterns of Human Figures Using a Multi-Layered Grid and the Dynamics Filter, *Proceedings of IEEE International Conference on Robotics and Automation*, pp.1-8

Sun, H.C. & Dimitris, N.M. (2001). Automating Gait Generation, *Proceedings of ACM SIGGRAPH 2001*

Witkin, A. & Popovic, Z. (1995). Motion Warping, *Computer Graphics Proceedings, SIGGRAPH95*, pp. 105-108

Yamane, K. & Nakamura, Y. (2000). Dynamics Filter – Concept and Implementation of On-Line Motion Generator for Human Figures, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 688-695