Appear in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2004

Planning Humanoid Motions with Striding Ability in a Virtual Environment

Tsai-Yen Li
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.
li@nccu.edu.tw

Pei-Zhi Huang
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.
g9102@cs.nccu.edu.tw

***Abstract*—Enabling a humanoid robot to move autonomously in a real-life environment presents a challenging problem. Unlike traditional wheeled robots, legged robots such as humanoid robots have advanced abilities of stepping over an object or striding over a deep gap with versatile locomotions. However, only few humanoid researches today have addressed this problem. In this paper, we consider the problem of planning a humanoid robot's motion in a layered environment cluttered with obstacles and deep narrow gaps. We extend the motion planning system for humanoids in our previous work to account for multiple locomotions and striding ability for a humanoid robot. Each locomotion corresponds to a tier in the search space, and the tiers are connected at the locations where motion transitions are possible. According to a humanoid's geometric properties, such as maximal gait size and step height, we propose to apply the Closing morphological operator to the workspace bitmap to compute the reachability region for a humanoid with the striding ability. Our experiments show that our system is efficient in generating versatile motions for a humanoid to reach its goal in a complex environment.**

***Keywords*- Motion Planning, Humanoid Robot, Versatile Locomotions, Computer Animation.**

I. Introduction

Humanoid research has attracted much attention in recent years. Appealing humanoid robots have been designed in several 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. It is an even more interesting problem for humanoid robots since locomotion capability possessed by a humanoid robot is usually much better than other mobile robots. 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 addition, a humanoid robot should be able to change its locomotion according to the environmental constraints in order to avoid obstacles or pass through narrow passages. Similar needs for the planning capability also arise in the domain of computer animation in generating motions for autonomous characters.

In our previous work, we have been able to plan efficient humanoid walk motions in a layered environment. The approach used in the planner decomposes the planning problem into global and local planning such that each subproblem 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 passed to the local planner to realize the path with appropriate walk motions. However, in our previous work, a humanoid's locomotion is limited to forward walking only, and a humanoid cannot pass a deep gap even if it can stride over it. In this paper, we will extend the work to overcome the above two limitations and 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 paper 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 brief description of the problem we consider in this paper. In the fourth section, we will present the planning algorithm that we propose in the paper. In the fifth section, some implementation details and experimental results will be presented. In the last section, we will conclude the paper with some future work.

II. Related Work

Motion planning problems have been studied for more than three decades. An overview of motion planning algorithms can be founding in Latombe's book [9]. The researches pertaining to humanoid motion planning or simulation can be found in fields of robotics and computer animation [8][11][13]. These researches differ mainly on the way they perform the global path planning and how the locomotion is taken into account. For example, early work in computer animation focused on generating human walking motion to achieve a high-level goal [1]. However, no planning is done to generate the global path automatically. Since the application is mainly for computer graphics, how to simulate human walking with realistic looking is the main concern [13]. A dynamic filtering algorithm is proposed in [14] to ensure that the generated motions can be transformed into a dynamically feasible one.

In [6], a gross motion planner utilizing graphics hardware has been proposed to generate humanoid body motion on a flat ground in real time. Captured locomotion is used in this case to move the humanoid along the generated global path. In [5], a stochastic search approach with versatile locomotion has been proposed to generate humanoid motions in an unstructured environment where a set of predefined grasp points serve as contact constraints. In [3], sequences of valid footprints are searched through augmented probabilistic roadmap

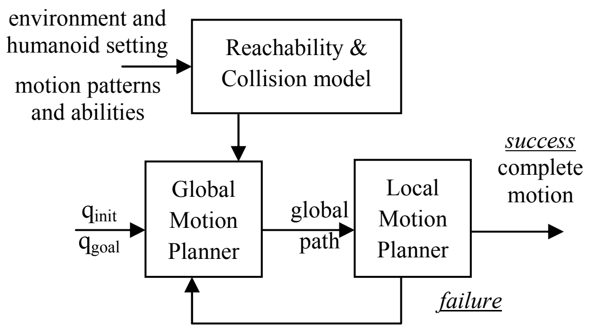


Fig.1. Planning loop for a typical query of humanoid motion.

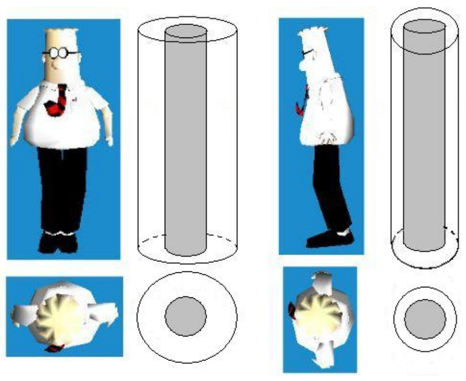


Fig.3. Frontal and lateral walking model for the humanoid.

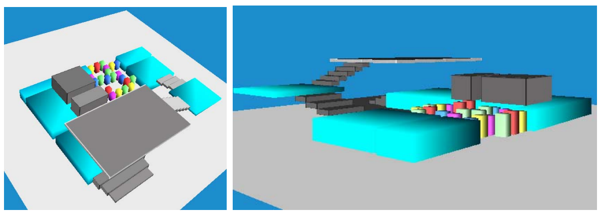


Fig.2. Example of a layered virtual environment with deep narrow gaps (colored columns between two large platforms)

with a posture transition graph, and then each pairs of footprints are substituted by corresponding motion clips. Since the motion clips are captured in advance, the locomotion may not be flexibly adapted to uneven terrain to avoid collisions. In [12], a multi-layer grid is used to represent the configuration space for a humanoid with different locomotion such as walking and crawling and the humanoid may change its posture along a global path. In [7] and [8], a humanoid robot with real-time vision and collision detection abilities is presented. The robot can plan its footsteps amongst obstacles but cannot 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. A recent work that can plan efficient humanoid motions in real time with a given general description of the objects in the workspace has been proposed in [10]. A more detailed description of this approach will be given in the next two sections.

III. Problem Description

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 can be 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 appropriate 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. 1. The global motion planning will be the main concern of this paper.

A. *Modeling the Environment*

We assume that we are given geometric description of the objects in the workspace such as the one shown in Fig. 2. Each object is given with 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 an object's bottom to some referenced ground. Each cell is also assigned a height value above the offset. According to the objects' offset values, we separate the 3D workspace into multiple connected 2D layers. Note that in a layered environment a point in the horizontal plane may have different offset and height values on different layers.

B. *Modeling the Humanoid*

The kinematic description of a humanoid robot includes the maximal gait size, step height, 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 humanoid's kinematic properties as well as the objects' geometry. For example, an object is an obstacle to a humanoid only if there is no way for the humanoid to step onto or pass under the object for a given locomotion due to the humanoid's height and leg length.

In global motion planning for a humanoid, we assume that the humanoid can be modeled as a bounding cylinder to simplifying collision detection. In our system, we uses a large and a small cylinders to model a humanoid in 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 locomotion. We use the largest lateral width orthogonal to the humanoid's moving direction to determine the radius. For example, the cylinder used to plan the side-walk motion is smaller than the one for regular walking motion.

The heights of both cylinders are also related to the humanoid's height and locomotions. If we allow the humanoid to bend its upper body during the walking cycle, the actual height may be lower than the humanoid's height. When using cylinders to model the geometry of a humanoid, we are actually ignoring its orientation at the planning time. We can recover the orientation of a humanoid in a postprocessig step according to the locomotion used.

We assume that a humanoid robot can perform several locomotions and choose the most appropriate one according to the environment. Although we can consider several major locomotions, such as walking, crawling, jumping and climbing in the local motion planner in order to generate a collision-free motion for the humanoid figures, only forward and side walking motions will be demonstrated in this work. In addition, we assume that the local planner can generate the motion transition from one type of locomotion to another at a given configuration.

IV. Motion Planning for Humanoids

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

A. Reachable Regions

The reachable region for a humanoid in the workspace is affected by several factors. For walking motion, we need to take the humanoid's maximal gait size and leg length into consideration as well as the bounding cylinder's height. In our previous work, we use neighboring cells' height difference and the humanoid's leg length to decide the reachable regions in each layer. The height difference between two neighboring layers is also checked to ensure that a humanoid can pass through under the constraint of the layers above the current one. However, for an environment with deep narrow gaps such as the one shown in Fig. 2, large height differences between neighboring 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 neighbors that are one cell away although the humanoid's maximal gait size could occupy several cells.

A straightforward approach will be extending the search algorithm to visit all the neighbors that are *d* 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 neighbors 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 paper, we propose a method to compute the reachability region from a given configuration. This method can accounts for the striding ability of a humanoid to handle discontinuous regions. The idea is that we can connect together the separated regions within gait range by using the "Closing" operator, an important morphological operator in image processing [4]. 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•B*, is defined as:

$$A \bullet B = (A \oplus B)\Theta B\,.$$

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. 4(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. 4(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. 2(a) except for connected discontinuous small regions.

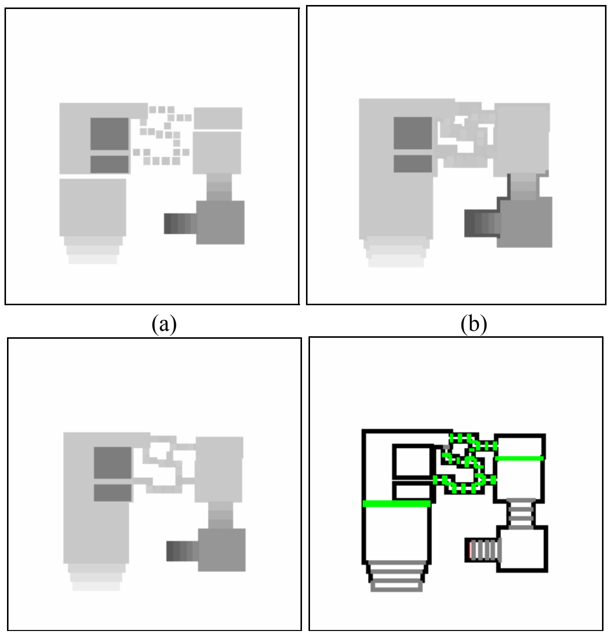


Fig.4. (a) height map for the environment in Fig. 2., (b) map in (a) after applying dilation, (c) map in (b) after applying erosion. After the closing operator, the columns in the discontinuous region are connected. (d) instability map

```
STABLE_BFP()
install $q_i$ in $T$;
INSERT($q_i$, OPEN); mark $q_i$ visited;
SUCCESS ← false;
while ┐EMPTY(OPEN) and ┐SUCCESS do
    $q$←FIRST(OPEN);
    for every neighbor $q'$ of $q$ in the grid do
        if $q'$ is stable then
            mark $q'$ visited;
        if LEGAL($q'$, $q$) then
            install $q'$ in $T$ with a pointer toward $q$;
            INSERT($q'$, OPEN);
            if $q' = q_g$ then SUCCESS ← true;
if SUCCESS then
    return the backtracked feasible path
else return failure;
```

Fig.5. The STABLE_BFP algorithm

We use the height map after the closing operation to determine each cell's reachability. A cell is considered an obstacle cell if and only if there are no ways to reach the cell from its neighbors under the height constraint. Given an initial configuration of the humanoid, we can compute a map, called *reachability map*, where obstacle regions are composed of the cells that cannot be reached. This map can be computed by a wave propagation algorithm such as the one used to construct NF1 potential field [9]. The only difference is that we advance to a cell only if the height difference is less than the humanoid's leg length.

B. Collision Map

We can convert the height map, built in the workspace, into its corresponding C-space by growing the obstacle regions with the bounding cylinder's radius of the humanoid for a given locomotion. We need to identify the regions where unstable situation might occur if a humanoid stays there for too long. A map describing the regions is called *instability map* (see Fig.4(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 humanoid's step height. Otherwise, if this region contains an object whose height difference is larger than humanoid's step height, we call the cell a *forbidden* cell. 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 to prevent the humanoid from staying in these regions for too long.

C. Path Planning Algorithm

The planning algorithm that we used to compute the humanoid motion is shown Fig. 5. The STABLE_BFP algorithm is similar to the classical Best-First Planning algorithm that is used to solve path-planning problems with low DOF's[2]. In each iteration of the search loop, we use the FIRST operation to select the most promising configuration q from the list of

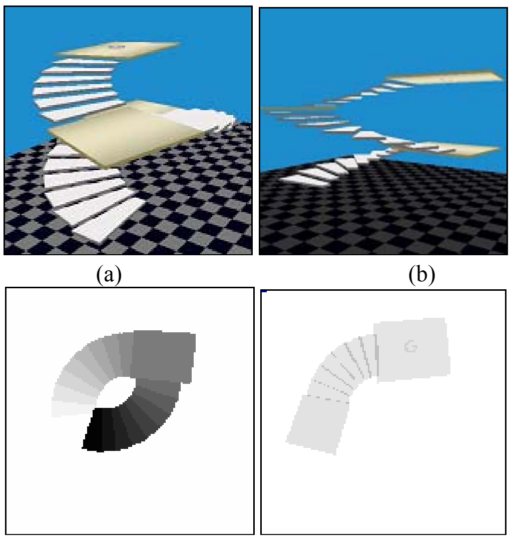
```
STABLE ($q'$,$q$)
    if $q'$ is unstable then
        $q'$.cnt= $q$.cnt+1;
    else if $q'$ is gap then
        if $q$.ingap then
            $q'$.step= $q$.step+1;
        else
            $q'$.step=1;
            $q'$.ingap=true;
            gap_begin= $q$;
    else if $q'$ is stable and $q$.ingap 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. 6. The STABLE and LEGAL procedures

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 locomotions, 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. 6. 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 region or gap region longer than some specified value. 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.*

In the STABLE_BFP algorithm, we use the FIRST operation to select the most promising configuration for further ex-



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

Fig.7. 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.

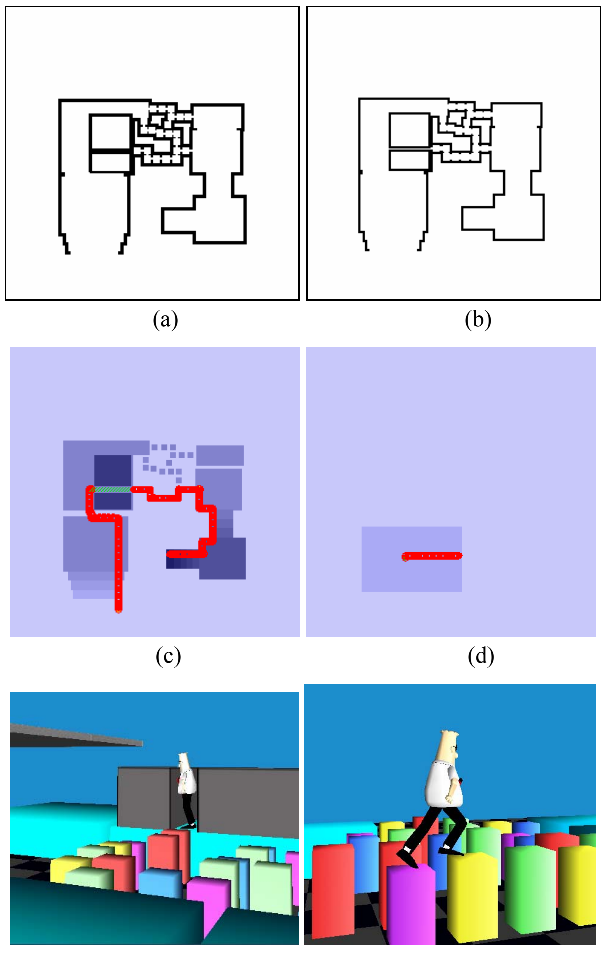
ploration. In the case of the BFP planners, the artificial potential field is usually the only index for the goodness criterium. Planners with this approach can usually yield short paths. In our case, the height difference could be an important index as well since one may prefer climbing up or stepping 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 area whenever possible.

V. IMPLEMENTION 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 an example to demonstrate the ability and efficiency of the planner in later subsections.

A. Merging Layers with Sparse Objects

In our system we use an object's offset value to separate the workspace into several 2D grid layers. However, for workspace like gyratory stairs, such as the one shown in Figure 7, 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.7. 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.



(a) (b) (c) (d) (e) (f)

Fig.8. (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.

B. *Example*

In Fig.8, we use an example to illustrate the feature of this planning system. The environment, which is the same as the one in Fig. 2, consists of two layers of objects with various sizes and heights scattered into two main platforms, connected by several marble columns of various heights. Our previous planner was not able to generate a path across these columns because of the deep gaps amongst them. In addition, note that the passage on the left platform is too narrow for the humanoid to walk through without switching to the side-walk locomotion. The C-obstacles (configuration space obstacles) with different bounding cylinders for forward walking and side walking are shown in Fig. 8 (a) and (b), respectively. Note that the narrow passage exists only when side-walk locomotion (Fig. 8. (b)) is used. The global path found by the planner is shown in Fig. 8 (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.

C. *Planning Performance Compare to Traditional Approach*

As mentioned in previous section, traditional straightforward search algorithms may also be able to find paths with the striding ability by extending neighbor search range. Compare 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.8 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. In other words, 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 reachability region is discontinuous.

VI. Conclusions and Future Work

Motion planning is one of the key abilities that an autonomous agent should have in the context of humanoid robot and animation character. Although previous work has made great process in generating humanoid motions automatically in layered environments and uneven terrain, the motions are still limited by the capabilities of the humanoid such as the available locomotions and striding. In this paper, we have successfully extended the planning system to consider such abilities and demonstrated the efficiency and effectiveness of the planner by simulation examples. In the future, we would like to enable the humanoid robot with more locomotion abilities such that it can have more motion choices in moving the robot to the goal. In addition, although the decoupled approach in our system is efficient, there must exist counterexamples where the local planner cannot realize the path found by the global planner. Therefore, we will also work on effective replanning algorithms by taking the feedback of the local planner into account.

Acknowledgement

This work was partially supported by National Science Council under contract NSC 92-2213-E-004-001.

References

[1] A. Bruderlin and T. W. Calvert, “Goal-Directed, Dynamic Animation of Human Walking,” *Proc. of ACM SIGGRAPH*, 1989.

[2] J. Barraquand and J. Latombe, “Robot Motion Planning: A Distributed Representation Approach,” *Intl J. of Robotics Research*, 10:628-649, 1991.

[3] M.G. Choi, J. Lee, S.Y. Shin, “Planning Biped Locomotion using Motion Capture Data and Probabilistic Roadmaps,” *ACM Transactions on Graphics*, Vol. V, No. N, pp.1–25, October 2002.

[4] R.C. Gonzale and R.E. Woods, *Digital Image Processing Second Edition*, Prentice Hall, 2002.

[5] M. Kalisiak and M. Panne, “A grasp-based motion planning algorithm for character animation,” J. Visual. Comput. Animat, pp.117-129, 2001.

[6] J. Kuffner, “Goal-Directed Navigation for Animated Characters Using Real-time Path Planning and Control” *Proc. of CAPTECH’98 Workshop on Modeling and Motion capture Techniques for Virtual Environments*, Springer-Verlag, 1998.

[7] J. Kuffner, et al., “Footstep Planning Among Obstacles for Biped Robots,” *Proc. of 2001 IEEE Intl. Conf. on Intelligent Robots and Systems (IROS 2001)*, 2001.

[8] J. Kuffner, et al., “Motion Planning for Humanoid Robots under Obstacle and Dynamic Balance Constraints,” *Proc. of IEEE Intl. Conf. on Robotics and Automation*, May 2001.

[9] J. Latombe, *Robot Motion Planning*, Kluwer, Boston, MA, 1991.

[10] T.Y. Li, P.F. Chen, P.Z. Huang, 2003.9, “Motion Planning for Humanoid Walking in a Layered Environment,” in *Proceedings of the 2003 International Conference on Robotics and Automation.*

[11] N. Pollard, et al., “Adapting Human Motion for the Control of a Humanoid Robot,” *Proc. of 2002 IEEE Intl. Conf. on Robotics and Automation*, pp.2265-2270, May 2002.

[12] Z. Shiller, K. Yamane, Y. Nakamura, “Planning Motion Patterns of Human Figures Using a Multi-Layered Grid and the Dynamics Filter” *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pp.1-8, May 2001.

[13] H. C. Sun and N. M. Dimitris, “Automating gait generation,” *Proc. of ACM SIGGRAPH*, 2001.

[14] K. Yamane and Y. Nakamura, “Dynamics Filter – Concept and Implementation of On-Line Motion Generator for Human Figures,” *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pp.688-695, April 2000