Appear in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2003

Motion Planning for Humanoid Walking in a Layered Environment

Tsai-Yen Li
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.

Pei-Feng Chen
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.

Pei-Zhi Huang
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.

***Abstract* - Motion planning is one of the key capabilities for autonomous humanoid robots. Previous researches have focused on weight balancing, collision detection, and gait generation. Most planners either assume that the environment can be simplified to a 2D workspace or assume that the path is given. In this paper, we propose a motion planning system capable of generating both global and local motions for a humanoid robot in a layered or two and half dimensional environment. The planner can generate a gross motion that moves the humanoid vertically as well as horizontally to avoid obstacles in the environments. The gross motion is further realized by a local planner that determines the most efficient footsteps and locomotion over uneven terrain. If the local planner fails, the failure is feedback to the global planner to consider other alternative paths. The implemented humanoid planning system is an interactive tool that can compute collision-free motions for a humanoid robot in an on-line manner.**

I.INTRODUCTION

The potential market of service and entertainment humanoid robots has attracted great research interests in the recent years. Several models of humanoid robots have been designed in research projects. Among the active research topics, enabling a humanoid robot to move autonomously with motion planning capability is one of the challenging problems that need to be addressed. An autonomous robot should be able to accept high-level human commands and walk in a real-life environment consisting of floors and stairs 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.

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. The first level only considers *global motion planning*, which is the motion planning of the whole body treated as a simple projected geometry. Given the gross motion from the first level, the second level only considers *local motion planning* that moves the legs of a humanoid robot to realize the corresponding gross motion in an efficient way.

In this paper, we propose a motion-planning system capable of generating efficient walking motions for a humanoid to reach a goal on a layered environment. We assume that the system is given an elevation and height description of the objects in the workspace and accept a goal-oriented command from a user. The system will generate a feasible global path and the associated locomotion that bring the humanoid to reach the goal as efficient as possible. At a first glace, the problem is similar to the general path-planning problem. However, since the definition of obstacles for this problem depends on the leg length of the humanoid and the local relative height, the problem definition deserves further clarification. A user may have personal preference on the paths if the goal can be reached via various paths of different heights. In addition, the motion plan proposed by a global motion planner may not always be feasible for locomotion arrangement. In this case, the interaction between the two levels of planning becomes an interesting problem.

The rest of the paper is organized as follows. After reviewing related work in motion planning and humanoid in the next section, we will describe in details the problem we consider in this paper. Then, we will then present our global and local motion planners in Section IV and V, respectively. In Section VI, we will present several examples from the simulation in our experiments. Finally, we will conclude our work with future directions in the last section.

II. RELATED WORK

The gross motion-planning problem was originally brought up in the context of robotics to generate collision-free path for mobile robots or manipulators. A survey of approaches to the problem can be found in [9]. Generally speaking, early research focuses on developing theoretical foundation and complete solutions for the problem [11]. Due to the curse of dimensionality, several researches in the last decade proposed practical solutions that can be applied to wider arrange of applications despite they usually lack completeness[1][2].

Many efficient planners have been proposed to solve the problem for objects with low degrees of freedom (DOF's) (typically less than or equal to four). Most of these planners are complete planners because they can always give a correct answer (success or failure) to the given problem. Among these planners, the potential-field based approach is the most popular one and is also the one used in our gross motion planner. An artificial potential field is typically used in the workspace as a heuristic to search the configuration space for a feasible path.[2]

The research of generating humanoid motions can be found in robotics and computer animation [3][10][14]. Although various aspects of motion generation have been studied, we will only concern the lower-body motion and the resulting body displacement. Early researches focus on generating a dynamically stable motion for a given path on a flat or uneven ground [4][5][7]. Although the locomotion for

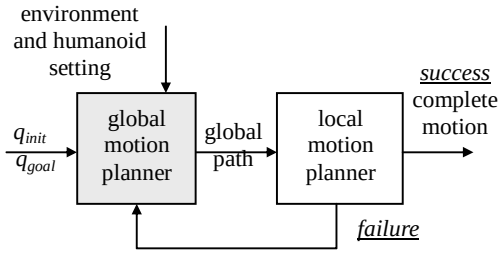


Fig. 1. Planning loop for a typical query of humanoid motion

regular walking can be computed kinematically, many approaches choose to use or modify motion-captured data due to the complexity of a human figure. Techniques such as motion warping [16] or dynamic filtering [15][17] are often used to ensure that the captured motions can be transformed into a dynamically feasible one. However, these techniques are not as flexible as kinematics-based methods in handling obstacles in an uneven terrain.

Not until recent years, the problem of gross motion planning for humanoid robots becomes one of the active research topics in robotics and computer animation [6][8][13]. 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 to move the humanoid along the generated global path. In [8], a biped robot can plan its footsteps amongst obstacles but cannot step onto them. In [13], a multi-layer grid is used to represent the configuration space for a humanoid with different locomotion such as walking and crawling. The humanoid may change its posture along a global path. In short, most gross motion planners for humanoid robots assume a flat ground and adopt canned motions for simplicity. However, the assumption is often over-restricted since a humanoid robot is more likely to work in a layered environment filled with objects of various shapes and heights as in the real life.

III. PROBLEM DESCRIPTION

According to motion granularity, the motion-planning problem usually can be classified into global (gross) motion planning and local (fine) motion planning. For the problem of walking on a layered environment for a humanoid, both types of planning needs to be considered in order to ensure that the desired task can be accomplished. Although the gross and fine motion planners can be designed separately and solved sequentially, we think they should be connected in a loop with feedbacks as shown in Fig. 1. The global path from the global motion planner is fed into the local motion planner to create corresponding footsteps and locomotion. However, the local motion planner may fail to generate locomotion for the given path. In this case, the planner should feedback the failure with reasons to the global planner to compute another global path. Taking this decoupled view can greatly reduce the complexity of such a planning problem. In the following subsections, we will describe the problems of global planning and local planning separately in more details.

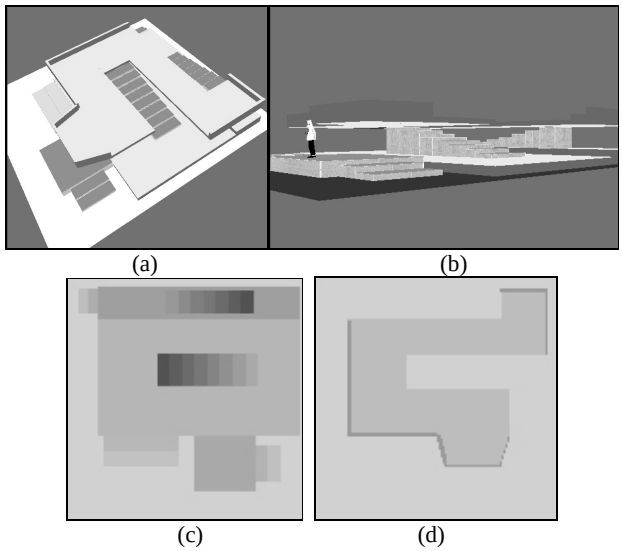


Fig. 2. (a) Top view of the workspace, (b) side view of the workspace, (c) and (d) are the height maps of first and second layers

A. Global Planning Problem

The global planner assumes that we are given a geometric description of the objects in the workspace as well as the geometric and kinematic description of a humanoid. The workspace contains multiple layers, and each layer is comprised of objects of various heights. Unlike the basic path-planning problem where the definition of obstacles is rather straightforward, the obstacles in our global planning problem are not explicitly given. Instead, 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 due to the humanoid's height. In addition, a humanoid must stand on a large enough area in order to maintain a stable stance. If the ground of the workspace is described as a smooth surface, the slope of the surface cannot be too large to cause foot slippery. In summary, the planning problem is rather complex in real life, and we need to make reasonably assumptions to simplify the problem.

First, we assume a discrete workspace. The input to our planner could be a continuous function for the elevation of the ground and a polygonal description of the objects. However, we assume that we can convert these descriptions into several layers of elevation grids of some resolution. Each cell in a grid contains an elevation value for the whole cell in that layer. An example of workspace with the layered environment is shown in Fig. 2. The elevation for each cell in a layered grid (128x128) is represented by a gray-scale value in Fig. 2(c) and 2(d). We denote the height of a cell i at layer l by c_i^l, and the offset of layer l from some reference ground by d^l. Second, we assume that the resolution of the elevation grid is coarse enough for a humanoid's foot to step

onto a cell. We also assume that the maximal height that a humanoid can step onto is denoted by *h*, which is a property of the given humanoid. Third, the height of the humanoid is *H*, and we assume that the humanoid does not bend its body to pass an obstacle for now. Fourth, we assume that a humanoid will not stay in the object boarder region for more than some designated units of time, *m*. This situation happens when the geometry of a humanoid intersects the boarder. This assumption is to make sure that the humanoid does not stay in the border region except for trespassing purpose. Fifth, we assume that the geometry of the humanoid can be simplified to an enclosing circle of radius *r* such that the orientation dimension can be ignored at planning time. We assume that a humanoid will always face forward and we can recover its orientation in a postprocessing step.

In summary, the objective of the global motion planner is to find a collision-free path for the body trunk of the humanoid to move from the initial configuration to the goal configuration in a two-and-half-dimensional space. The output of the planner is a global path that will be sent to the local planner for further processing.

B. Local Planning Problem

The local planner aims to find a feasible locomotion for the lower body of a humanoid with a given global path. We assume that the output path from the global planner is a 3D stepwise curve. This curve is a polyline comprised of a set of vertical or horizontal connected line segments. In other words, we temporarily ignore the orientation change of the path and stretch the path into a one-dimensional stair-like profile. According to the kinematic parameters of the humanoid, the local planner will generate a *feasible* and *efficient* plan for footstep placement and the corresponding locomotion for lower-body joints. A feasible motion plan must satisfy geometric and kinematic constraints. For example, the humanoid should be collision-free and all joints are within their joint limits. However, we do not use any explicit dynamics model for simplicity reasons, and we assume that this simplification does not cause dynamics feasibility problems in normal walking motions. By efficient plans, we mean that the path should be the most efficient in terms of energy consumption. An efficient motion usually also means a natural motion that a human normally takes.

The local planning problem described above is challenging because the number of possible arrangements (each arrangement consists of a set of footsteps) grows exponentially in the length of the global path (or number of footsteps) even if we restrict the possible footstep sizes to a limited number. However, according to our daily walking experience, we typically plan foot placement only for the next two or three steps instead of for the whole path. Therefore, it is reasonable to take an incremental approach where we call the local planner in every step to plan only for a few steps (two or three, typically) ahead. Another advantage of this approach is that we can allow the configuration of obstacles to change at run time without calling for global replanning immediately as long as the change does not prevent the local planner from generating feasible locomotion. Thus, we will redefine our local planning problem as finding a feasible locomotion for the next *n* steps with a given path profile. The planner should return failure and indicate the failure location along the path if it cannot find a feasible locomotion plan for the next *n* steps.

IV. GLOBAL MOTION PLANNING

We will now present our approach to solving the global motion-planning problem. In addition to being collision-free, all configurations along the path must be reachable according to the kinematic constraints, such as joint limits, of the humanoid robot. The path must also satisfy the stability constraint requiring that any continuous portion of the path cannot stay in the border region for longer than some period of time. In the following subsubsections, we will first compute a reachability map and then a collision map to represent the properties of the grids in the configuration space.

A. Reachability Map and Collision Map

Suppose that we are given the heights and offsets for a set of objects in the workspace. Offset is the base elevation where the object is placed. Different offsets mean different layers. For each layer, we compute a height map containing the elevation value of each cell above the layer in the workspace grid. As indicated in the previous section, 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. A cell could be unreachable because its height difference with its neighbors is too large for a humanoid to step onto or the clearance between this layer and the layer above is too small for a humanoid to fit in.

Given an initial configuration of the humanoid, we can compute a map, called *reachability map*, where obstacle regions are comprised of the unreachable cells. A reachability map consists of several slices with one slice for each layer. For example, Fig. 3(a) and 3(b) show the reachability map for the two layers of the example scene in Fig. 2. 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 [9]. Suppose that the current cell under propagation is *i*. The algorithm advances to a neighboring cell i' at 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 cell i' at 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). We can convert this map, built in

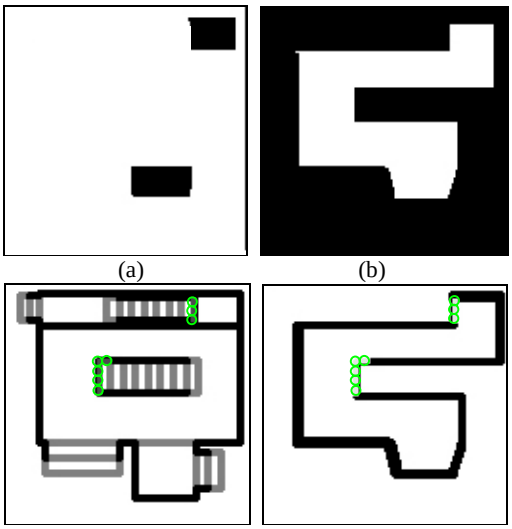


Fig. 3: The reachability map (a)(b) and collision map (c)(d) for layer one and two of the environment in Fig. 2.

the workspace, into its corresponding C-space by growing the obstacle regions with the humanoid's radius *r*. The resulting C-space map can then be used to build a potential field to guide the search in the planning process.

As mentioned in the previous subsection, we need to identify the regions where unstable situation might occur if a humanoid stay there for too long. A map describing the regions is called *collision map*. A cell in the collision map is defined as *unstable* if and only if the region covered by the enclosing circle of a humanoid contains cells with height difference less than *h*. The region comprised of unstable cells is called the *unstable region*. A cell is defined as *forbidden* if there exists height difference larger than *h*. The remaining cells in the collision map are the *free* cells. One can compute such a map by first identifying the cells with different heights in their neighborhood horizontally and vertically. We can then grow these cells by the radius *r* to form the final collision map. These two maps are used in the planning algorithm presented in the next subsection.

B. Path Planning Algorithm

The planning algorithm for computing the humanoid motion is shown Fig. 4. The STABLE_BFP algorithm is similar to the classical Best-First Planning (BFP) algorithm [9] for low-DOF problems. In an 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* and check their validity (via the LEGAL operation) for further consideration. A configuration is legal if it is collision-free (in the freespace of reachability map), marked unvisited, and temporarily stable. It is temporarily stable if and only if the humanoid has not entered the unstable region for a period longer than a given threshold determined by the user. This

```
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 LEGAL(q') then
8         if q' is unstable then
9           q'.cnt=q.cnt+1;
10        else
11          mark q' visited;
12        install q' in T with a pointer to q;
13        INSERT(q', OPEN);
14        if q' = q_g then SUCCESS ← true;
15  if SUCCESS then
16    return the backtracked feasible path
17  else return failure;
```
Fig. 4: The STABLE_BFP algorithm

duration is kept as an instability counter for each cell in the unstable region when we propagate nodes in it. Note that the validity of a configuration in the unstable region depends on the instability counter of the 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 region. A configuration in this region can be visited multiple times as long as its instability counter does not exceed the maximal value.

In the STABLE_BFP algorithm, we use FIRST operation to select the most promising configuration for further exploration. In the BFP planners, an artificial potential field is usually the only index for goodness. 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. Therefore, in the FIRST operation, we use a linear combination of both criteria (potential for horizontal measures and height difference from q_i for vertical measures) with weights specified by the user.

C. Postprocessing

If our planner succeeds in finding a feasible path for a humanoid, we need to perform two tasks in the postprocessing step. First, we convert the found path into a smooth one via a smoothing routine. The smoothing algorithm is very similar to the ones for typical path planners. One usually replaces a subpath in the original path continuously with a straight-line path segment. A major difference for smoothing a path in our new planner is on the metric for measuring distance. This metric is defined with the same criteria as in the FIRST procedure such that the user preference can be preserved. Finally, we have to recover the orientation parameter of the humanoid for each step in the path. If the path contains a sharp turn that is hard for a humanoid to follow,

we can add additional steps into the path to slow down the orientation change.

V. LOCAL MOTION PLANNING

From the problem definition described in the previous section, we assume that the local planner is asked to plan an efficient motion only for the next few steps (two or three, in our settings) instead of for the whole global path. We will first describe how we compute a collision-free walk locomotion for a given footstep length and a path profile. Then we will describe how to plan the foot placement for future steps.

A. Kinematics-Based Locomotion Generation

In this subsection, we will describe how we generate a collision-free locomotion for humanoid walking with a given foot location for the next step. We use an inverse-kinematics approach to compute the locomotion. According to [3], walking motion for a human figure can be divided into two phases: *single support* and *double support*. In the double support phase, both legs touch the ground and mass center of the body shift gradually from the back leg to the front leg. In the single support phase, a stretched leg supports the body while the other leg swings forward. Keyframes are defined at the conjunction of the two phases or in the interior of the double support phase.

We use the following principles to compute the joint angles between two keyframes. In either phase, the stretched leg on the ground determines the pelvis location. In the case of double support, the joint angle of the other leg is determined by inverse kinematics between the foot and pelvis. In the single support phase, the foot trajectory for the swinging leg in the Cartesian space is determined first and the joint angles are then computed according to inverse kinematics.

Now, the remaining problem is how to compute a collision-free trajectory for the swinging foot. We use a Bezier curve with two fixed endpoints on the ground to represent the trajectory. The other two control points of a Bezier curve become the parameters that we can adjust to avoid collision with obstacles. We use a numerical approach that move the control points iteratively until a legal curve is found. However, the four-dimensional space spanned by the two control points is too large to search in an on-line manner. Therefore, we use the following rule to find the collision-free curve as quickly as possible and avoid an exhaustive search.

The locations of the control points start at some location slightly above the midpoint between two endpoints. When a collision between the curve and the environment is detected, the control points are moved upward and outward to lift the curve as shown in Fig. 5. The curve is divided into two parts (left and right) from the summit point. Each part computes the intersected points of the curve and moves the corresponding control point for some distance along the direction from the midpoint of two endpoints toward the intersection points. If one part does not cause intersection while the other does, its control point also moves upward for the same amount as the other control point moves. The search process stops when a legal curve is found or the locations of the control points have left a given legal region. In the second case, the planner returns failure.

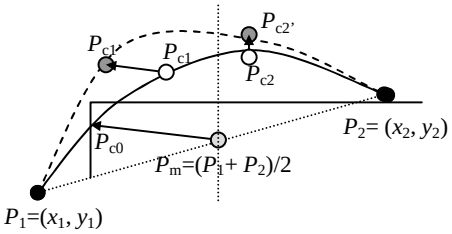


Fig. 5: Adjusting the Bezier curve for the swinging leg to avoid collisions

B. Footstep Planning

We now consider the problem of finding foot placement. Assume that the local planner is called in every step of execution, and we will search for the most energy efficient locomotion for the next n steps, where n is set to, say, 3. We first discretize the range of all possible footstep sizes into m unit lengths for each step. Then we try to find a set of desirable footstep locations, called *footstep configuration*, denoted by $q_f=(s_1, s_2, s_3)$, where s_1, s_2, and s_3 are step lengths. The energy efficiency for q_f, denoted by $Eff(q_f)$, is defined by $(s_1+s_2+s_3)/E$, where E is the total energy consumption that will be defined later.

Depending on the available time for planning, one may choose to search for the global maximal or a local maximal configuration. In our on-line application, since the planner is called in every step of execution at run time, time efficiency is crucial. Therefore, we use a Best-First Search algorithm to find a collision-free locomotion plan for the next n steps with the maximal energy efficiency only locally. The algorithm starts from some neutral position initially and explores its neighbors for better configurations. We maintain a list of available configurations and choose the most promising configuration in each step for further exploration until a local maximal is reached. The program returns failure if no such a footstep configuration exists. In an iteration of the execution loop, we plan for the next few steps by taking the goal/current footstep configuration as the starting point for the next iteration. In most occasions, the previous footstep configuration usually provides a good initial guess that can quickly bring the search to a local maximum.

The energy E consumed in one step depends on several factors. For example, it depends on the walking speed, vertical and horizontal movements, and joint movements. Although the energy consumption for a given humanoid robot can be computed with a dynamic model, our real-time planning requirement discourages such an approach. Instead, we take a simpler approach of computing this energy from statistic data obtained from real human walking experiments. We assume that the energy consumption is composed of three parts: horizontal movement, vertical movement, and

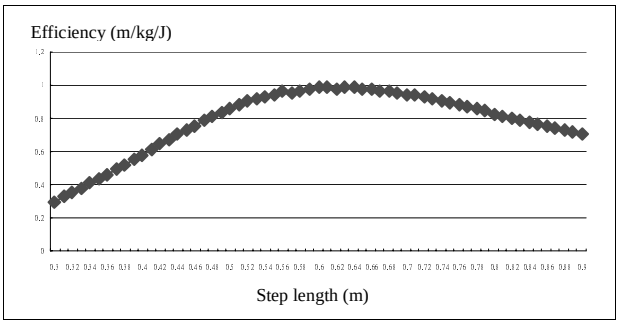


Fig. 6. Energy efficiency for different step lengths on a flat ground

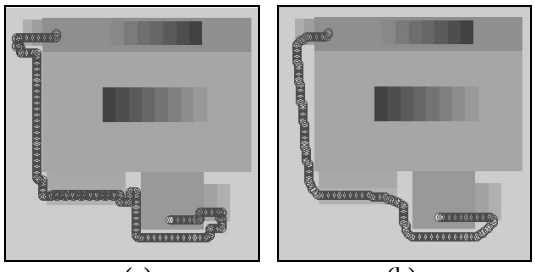
joint movement. That is, $E=\alpha*E_n+\beta*E_h+\gamma*E_r$, where E_n, E_h, and E_r are the energy for horizontal movement, vertical movement, and joint movement, respectively. We assume a normal walking speed so that we have a common ground for comparison. According to [12], the energy for normal walking on a flat ground can be computed with the following formulas: $E_n = E_w / n = (32+102*s^4) / n$, where s is the step size and n is the number of steps per minute. Experiments show that s/n=0.007 for adult male. Therefore, E_n can be simplified to be a function of s only. According to [12], the energy consumption for stepping upward is proportional to the stepping height. Third, for the same footstep size, a humanoid could raise legs for different amounts and therefore consume different amounts of energy. We take a weighted sum of changes on the joint angles to compute the extra energy consumption, E_r. The weights are mainly determined by the masses of the raised parts. With these formulas, we can compute the energy efficiency for different footstep lengths, as shown in Fig. 6. Since the statistic data is taken from experiments with real humans, the most efficient step length is around 60cm, which agrees with our daily experiences.

VI. 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 650MHz PC. The resolutions for the grid workspace and configuration space are all 128x128 in the global planner. The resolution for the locomotion is on the discretization of footstep length, which is divided into 13 units in the range of 30cm to 90cm. In the following subsections, we will use several examples to demonstrate the effectiveness of the planners.

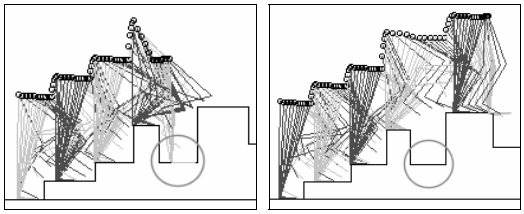
A. An Example for Global Motion Planning

Fig. 7 shows an example of global motion generated for the scene of Fig. 2. Fig. 7(a) and 7(b) show the unsmoothed and smoothed global paths that take the humanoid from a stage to the other side of the platform.. Note that there are two ways to get to the goal location, and the planner generates the shorter path to reach the goal. The planning time for



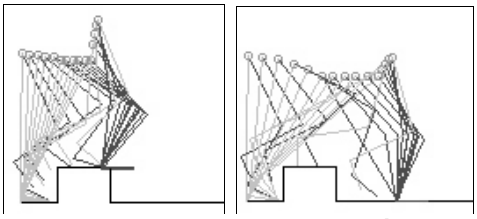
(a) (b)

Fig. 7. An example of (a) unsmoothed and (b) smoothed global path in a layered scene



(a) (b)

Fig. 8. Locomotion planning: (a) fails with one-step planning and (b) succeeds with two-step planning



(a) (b)

Fig. 9. Efficient locomotion generation: (a) stepping on is less energy efficient than (b) stride over

such an example is 220ms for preprocessing computation (such as building the reachability map, collision map, and potential field) and 20ms for path searching and smoothing.

B. An Example for Local Motion Planning

In Fig. 8, we show an example of local motion plan that takes the lower body of a humanoid to climb up a stair with a given ground profile. In the example of Fig. 8(a), the humanoid fails to make further moves after stepping down the stair because it only plans one step ahead. In Fig. 8(b), we plan two steps ahead and therefore avoid the problem by taking a larger step length. In Fig. 9, we compare the planning results with and without considering energy efficiency. For example, according to the energy model described in Section V, the motion in Fig. 9(b) is more efficient than the motion in Fig. 9(a) because it does not need to step onto the obstacle.

C. Integrated Example

We use the integrated example in Fig. 10 to demonstrate the interaction between the global and the local planners. The global motion generated initially upon a user request

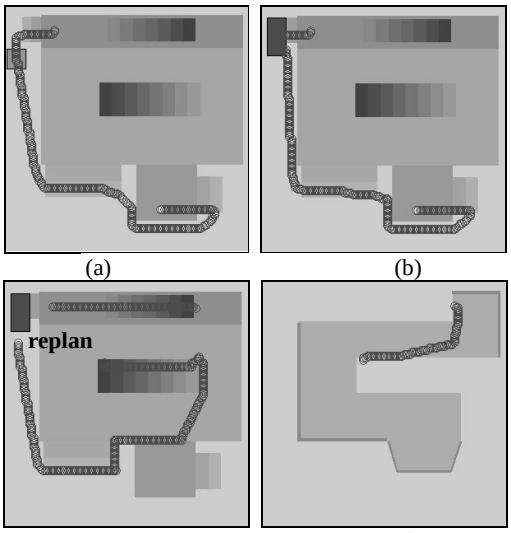


Fig. 10. Placing (a) a short object and (b) a tall object on the path at execution time. (c)(d) An alternative path is generated.

may not succeed at execution time when it is sent to the local motion planner for further processing. For example, an object could be placed on the way of the global path at execution time as shown in Fig. 10(a) and 10(b). Although the new green square object in Fig. 10(a) is on the path, the local planner generates a locomotion plan that steps over the object, as shown in Fig. 11(b) (compared to the original plan in Fig. 11(a)). If the placed object is too tall (Fig. 10(b)), the local planner may fail at execution time. In this case, the local planner returns the failure point to the global planner in order to generate an alternative feasible path via the second floor, as shown in Fig. 10(c) and 10(d). The re-planning takes about 761ms including rebuilding all the maps.

VII.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 paper, we have proposed a planning system composed of global and local motion planners that can generate feasible and efficient motion plans for a humanoid in a layered environment. Several simulation examples have been presented to demonstrate the capabilities of the planner.

In the current system, we assume that a humanoid robot can only perform normal walking motions. However, a real human usually can avoid obstacles with various kinds of locomotion and body motions. For example, a human can crawl or stoop to avoid upper-layer obstacles. We will consider this kind of situations as an extension of our work in the future.

VIII. ACKNOWLEDGMENT

This work was partially supported by National Science Council under contract NSC 91-2213-E-004-005.

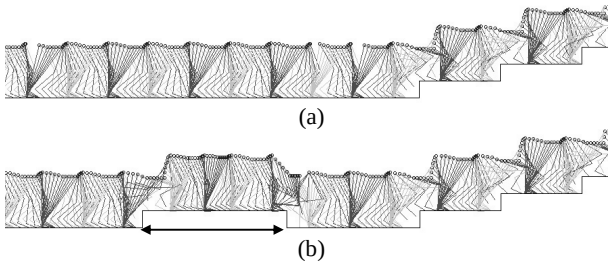


Fig. 11. Locomotion adapting to a dynamically placed object

IX. REFERENCES

[1] J. Barraquand, L. Kavraki, J.C. Latombe, T.Y. Li, and P. Raghavan, "A Random Sampling Scheme for Path Planning," *Intl. J. of Robotics Research*, 16(6), Dec. 1997.

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

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

[4] Q. Huang, K. Kaneko, et al., "Balance Control of a Biped Robot Combining Off-line Pattern with Real-time Modification," *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pp.3346-3352, April, 2000.

[5] H. Ko and N.I. Badler, "Animating Human Locomotion with Inverse Dynamics," *IEEE Transaction on Computer Graphics*, 16(2), pp.50-59. 1996.

[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., "Motion Planning for Humanoid Robots under Obstacle and Dynamic Balance Constraints," *Proc. of IEEE Intl. Conf. on Robotics and Automation*, May 2001.

[8] 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.

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

[10] N. Pollard, et. al., "Adapting Human Motion for the Control of a Humanoid Robot," *Proc. of 2002 IEEE Intl. Conf. on Robotics and Automation*, pp2265-2270, May 2002.

[11] J.H. Reif, "Complexity of the Mover's Problem and Generalizations," *Proc. of the 20th IEEE Symp. on Foundations of Computer Science*, pp. 421-427, 1979.

[12] J. Rose and J.G. Gamble, *Human Walking*, Williams and Wilkins, 1994.

[13] 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.

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

[15] S. Tak, O. Song, and H.-S. Ko, "Motion Balance Filtering," *Proc. of the Eurographics Conf.*, 2000.

[16] A. Witkin and Z. Popovic, "Motion Warping," *Computer Graphics Proc.*, SIGGRAPH95, pp.105-108, 1995.

[17] 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.