Paper: jc12-3-3332:2008/4/25

A Unified Approach to Planning Versatile Motions for an Autonomous Digital Actor

Yueh-Tse Li and Tsai-Yen Li

Computer Science Department, National Chengchi University
64, Sec. 2, Zhi-Nan Rd. Taipei, Taiwan 116
E-mail: li@nccu.edu.tw
[Received ; accepted]

Enabling a digital actor to move autonomously in a virtual environment is a challenging problem that has attracted much attention in recent years. The systems proposed in several researches have been able to plan the walking motions of a humanoid on an uneven terrain. In this paper, we aim to design a planning system that can generate various types of motions for a humanoid with a unified planning approach. Based on our previous work, we add two additional motion abilities: leaping and moving obstacles into the system. In previous work, the order of moving obstacles is determined first, and then the corresponding paths for the pushing/pulling motions are generated. In this work, we take a unified approach that accounts for all types of motions at the same time. We have implemented a planning system with this unified approach for a humanoid moving in a layered virtual environment. Several simulation examples are demonstrated in this paper to illustrate the effectiveness of the system.

Keywords: unified planning, motion planning, autonomous digital actor, leaping and pushing motions

1. Introduction

In a 3D virtual environment, allowing human-like digital actors to plan their motions automatically is a challenge similar to the motion-planning problem for autonomous robots. Direct manipulation is commonly used to walkthrough a virtual environment with control devices such as keyboard and mouse. However, we are more interested in designing a system with a higher level of control that can generate appropriate motions along a collision-free path for the actor to reach the goal. In recent years, Kuffner [1] , Shiller [2], and Li [3] have proposed various methods to generate humanoid motions in complex environments. In this paper, we extend our previous work [3] to design a unified approach to incorporate more motion abilities for digital actors in a layered virtual environment.

The motion abilities of a digital actor can be roughly classified into *movement* and *manipulation*. All types of locomotion that allow a digital actor to move its body belong to the type of movement while grasping and pushing objects belong to the type of manipulation. In this work, we hope that a digital actor can generate motions to reach its goal by moving some obstacles. In previous work, this problem has been shown to be a complex motion planning problem [4]. In [5], the authors used a graph structure to represent the state space of the environment for determining the moving order of the obstacles and then the path for realizing the corresponding moving and passing motions. Several complex examples have been created to illustrate the need of manipulating movable obstacles for reaching a goal that was not reachable originally.

In most previous work, the ability of moving obstacles is usually treated as a special planning case triggered only when no feasible paths exist without moving the obstacles. However, in our daily experience, we may choose to move an obstacle (such as a chair) if the alternative path to the goal is a long detour. In other words, instead of pre-determining the order of these possibilities, we take all possible solutions into account at the same time according to their costs such as energy consumption or path length. For example, we may push a chair aside to pass a narrow passage or we may change our locomotion to lateral walking if the clearance allows this kind of passing. In this paper, we propose a unified approach to this planning problem that put all motion abilities on the same scale in order to create a sequence of various motions that better match our daily experience.

We will organize the rest of the paper as follows. In the next section, we will review the work related to our research. In Section 3, we will formally describe our problem definition. In Section 4, we will describe how to plan the motion of an actor to leap over obstacles. In Section 5, we will describe a new motion ability about moving obstacles with a unified approach. In Section 6, we will present the experimental results generated by our planning system.

2. Related Work

The problem of motion planning (or known as the piano-mover's problem) [6] has been well studied in the field of robotics in the past three decades. A variety of problems and algorithms can be found in Latombe's book [7]. Although the problem was motivated by robot

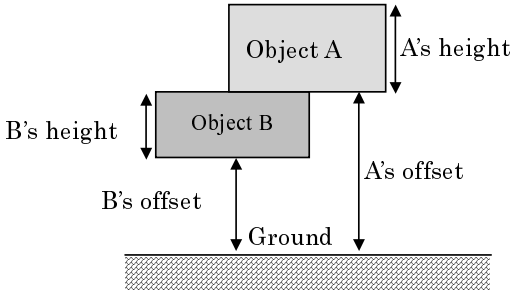

Fig. 1. Illustration of object heights and offsets.

automation, the techniques developed for this problem has been widely used in many other fields, including computer animation, drug design, and many others. According to [8], most motion planners divide the planning process into two phases: the *preprocessing* phase for converting the original geometric reasoning problem into some abstract data structure (ADT) and the *query* phase where we search the ADT for a feasible path.

Due to the complexity of a humanoid model, the planning problem for a humanoid digital actor is usually decoupled into a two-level problem: *global motion planning* for a feasible path connecting the initial and goal configurations and *local motion planning* to realize the global path with humanoid locomotion. In [9], the authors represented the virtual environment with a layered structure consisting of objects of various heights such that the global planner can take the terrain information and object heights into account. In [3], the digital actor was further equipped with several motion abilities such as jumping, side-walking, and striding over deep gaps.

The problem of manipulating movable objects was studied in some early work in robotics [4, 10, 11]. For example, in [12], the controllability and planning of object pushing has been studies for mobile robots. In [5], moving obstacles was considered as a special ability of the digital actor that can be used to find a path to reach the goal that was originally unreachable. Given a goal configuration of the actor, the planner first needs to know whether a feasible path without moving obstacles exists or not. If not, by analyzing the relations of the connected components of the freespace, the planner can decide which obstacles to move and attempt to move them to some unobstructive positions. Although good examples have been demonstrated in this work, the workspace is limited to a 2D plane and the ability of moving obstacles is considered only when no feasible path exists. In this paper, we would like to account for various motion abilities in a more unified way for a more general scene consisting of a layered 3D environment.

3. Problem Description

In our system, we assume that the workspace consists of several objects (may be treated as obstacles) of various heights and offsets from the reference ground as shown in **Fig. 1**. We use the formulation of [9] to organize the obstacles in the environment: objects of the same offset are grouped into the same layer. The objects that are movable by the actor are assumed to be given.

We assume that we are given the geometric and kinematics description of the humanoid actor. We also assume that the actor is equipped with the motion abilities of (1) frontal walking, (2) side walking, (3) jumping, (4) striding over a gap, (5) leaping over an obstacle, and (6) moving an object, where the last two abilities are added in this paper based on the work of [3]. We know the maximal step length that the actor can stride over and a maximal height that the actor can step onto. Therefore, an object with a height less than this maximal stepping height may not be treated as an obstacle. If the actor needs to pass under some obstacles in an upper layer, the clearance between the two layers must be larger than the height of the actor. For the newly introduced leaping motion, we assume that there exists a maximal height and a maximal depth limiting the geometry of the obstacles that the actor can leap over.

4. Leaping over Obstacles

Compared to the work of [3], we have added a new motion ability: leaping over obstacles in this paper. An actor with this ability will use his hands as a hold to leap over an obstacle. However, not all obstacles can be leaped over successfully. The height of the obstacle must be within a range of $[H_{\min}, H_{\max}]$ defined according to the kinematics property of the humanoid. In addition, the depth of an obstacle cannot be too large for the actor to leap over.

In order to find out the obstacles that can be leaped over, we have used the Opening operator [13] commonly used in Computer Vision to process the bitmap representing the workspace. The operation (denoted by $\circ$) working on a set A and a structure element B is defined as

$$A \circ B = (A \Theta B) \oplus B \quad \ldots\ldots\ldots\ldots \quad (1)$$

where Θ and $\oplus$ means erosion and dilation operations, respectively. Each layer of the workspace is represented as a map with each grid cell containing a height value. The set of cells in this map with heights in $[H_{\min}, H_{\max}]$ are treated as A in Eq. (1). The structure element B in Eq. (1) is a circle enclosing the geometry of the humanoid in performing a maximal leaping motion. After the erosion operation, the obstacle regions that can be leaped over will disappear, and the regions that are too wide for leaping will grow back to their original shape after the dilation operation. For example, in **Fig. 2**, a height map (**Fig. 2(b)**) is first generated for the scene in **Fig. 2(a)**, and the maps after erosion and dilation are shown in **Fig. 2(c)** and **2(d)**. Comparing to the original map, the region occupied by a thin obstacle in the lower portion has disappeared and becomes reachable by the use of the leaping motion. The resulting height map will then be used to build a potential field to guide the search for a feasible path.

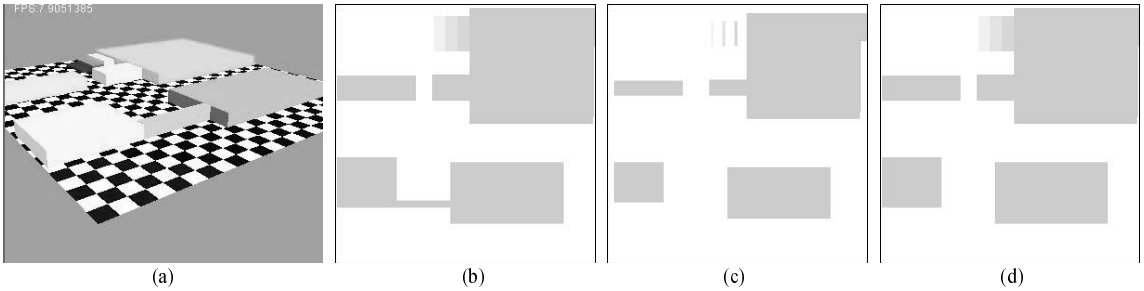

Fig. 2. Opening operation (a) inclined view of the scene (b) original height map (c) after erosion (d) after dilation.

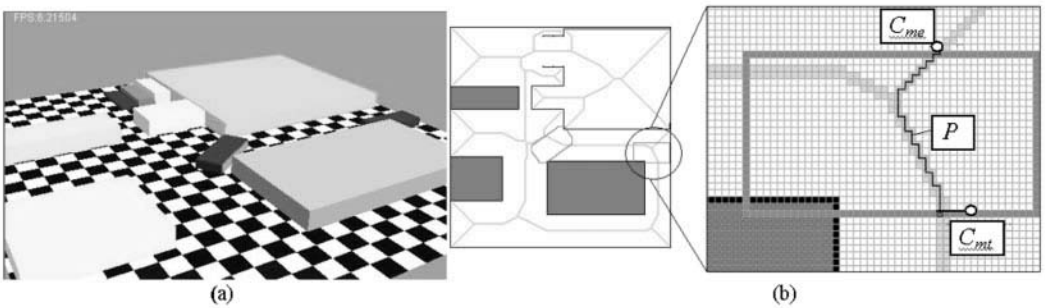

Fig. 3. (a) An inclined view of the scene with two movable obstacles (b) Skeleton, triggering point, and estimated path P.

5. Motion Planning with Movable Obstacles

Moving obstacles around to make space for passing is a common strategy that a human uses to reach a destination. For example, we may move chairs around to pass a cluttered environment even though there could be other detoured doorways that can be used to reach the same destination. The decision of moving obstacles depends not only on the feasibility of creating a new path but also on the efforts of making such a move. In this section, we will describe our unified planning approach to take this special motion ability into consideration.

5.1. Problem Definition

We assume that the movability of the obstacles with respect to a given actor is specified by the user initially. However, in order to avoid deadlocks, we allow an obstacle to be moved only once. Thus, the movability of an obstacle will be disabled after it has been moved. We also assume that the available grasp or push points are given. (In our current implementation, a grasp point could be any point along the boundary of an obstacle.) An obstacle can be moved on its own layer (plane) and the final location needs to allow the obstacle to be placed stably. In addition, during the movement, it needs to avoid collisions with other obstacles on the same layer or possibly on other layers due to its height. We assume that the geometry of the humanoid actor can be represented by an enclosing circle. Thus, by growing the workspace by the radius (r) of this circle, we can compute the corresponding configuration space, where the robot (the actor) can be treated as a point.

5.2. Estimating Actor's Trespassing Path

In [5], the planning for moving obstacles is triggered when the obstacles decompose the freespace into several separated components. However, in our work, the attempts to move obstacles are made when the obstacles are blocking the paths that can potentially take the actor to the goal. We first build a reachable region and a potential field [3] by ignoring all movable obstacles. Then, a voronoi-diagram-like skeleton [14], as shown in **Fig. 3**, is computed for the freespace by the use of the NF2 procedure described in [7]. The obstacles are then placed back to the workspace when the search for the actor's path starts. The planning for moving an obstacle is evoked when the search has reached a boundary cell of the movable obstacle, denoted by O_m. This boundary cell is called *triggering point* (C_{mt}). We first compute an estimated path for the actor to trespass the obstacle by temporarily ignoring the obstacle. Then we try to move the obstacle out of this path by defining another path planning problem for it. The estimated path starts from C_{mt} and is connected to the nearest point in the skeleton. We then continue to search on the skeleton for a point (C_{me}) in the freespace. This resulting estimated path, denoted by P and shown in **Fig. 3(b)**, will be used as the path of the actor that the movable obstacle should avoid colliding with.

5.3. Planning for the Movable Obstacle

Once we have the estimated path for the actor, we can define a path planning problem for the movable obstacle as follows. The subject under our planning is the movable obstacle (O_m) that is treated as a free-flying object on a plane with three degrees of freedom. Other obstacles in the environment are treated as the obstacles for O_m to

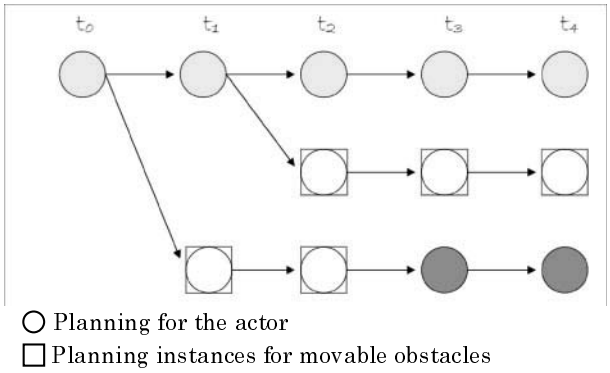

Fig. 4. Creating planning instances for unified search.

avoid colliding during the planning. We use them to build the C-obstacles in the 3-dimensional C-space of O_m. The initial configuration of O_m interferes with the path P, and our goal is to move it to a configuration without interfering with the actor along the estimated path. We use a breadth-first search algorithm to search for such a configuration. During the search, O_m needs to remain collision-free from all other obstacles at all layers and stay inside the region of its current layer such that it can remain stable after the move. In addition, in order to prevent O_m from being moved to a configuration that may block the return path of the actor, we further require that at the found final configuration, O_m needs to keep a minimal distance r from other obstacles such that the configuration of O_m does not prevent the actor from returning to its original configuration before moving O_m.

5.4. The Unified Approach

Our approach differs from the previous work in that we do not pre-determine the necessity of moving an obstacle. Instead, we hope that the search of moving an obstacle can be considered along with the regular search for a feasible motion of the actor. When the search for moving an obstacle is triggered, we create a planning instance for it by duplicating the current system configuration, including the configurations of the actor and other obstacles, at the moment. Then, a planning thread is spawn for this instance and run in parallel with the original thread, as illustrated in **Fig. 4**. However, instead of creating these planning instances in different threads provided by the operating system, we simulate the parallelism by allocating a specific period of time Δt for a planning instance when it is selected for planning. After this period of time, if no feasible solution for the movable obstacle is found, the control will be temporarily returned to the system, and the planning instance will be inserted into a candidate list. This planning instances may be evoked again at a later time, and the search will continue from the leftover point in the last run. If a planning instance succeeds, the planning for the actor will continue by assuming that the movable obstacle has been updated to its new configuration, as illustrated by circles in **Fig. 4**. We call this type of approach *a unified approach* because different types of planning are considered simultaneously.

STABLE_BFP(q_i,q_g)

```
1.  install q_i in T;
2.  INSERT(q_i,OPEN); mark q_i as visited;
3.  SUCCESS <- FALSE;
4.  while not EMPTY(OPEN) and not SUCCESS do
5.    q <- FIRST(OPEN);
6.    if q.manip is TRUE then
7.      Manip_Success <- q.Instance(Δt)
8.      if Manip_Success then
9.        q.searchspace=
          Create_Searchspace(q.instance);
10.       q.manip <- FALSE ;
11.     INSERT(q ,OPEN);
12.   else
13.     for every neighbor q' of q in the grid do
14.       if q' is trigger point
15.         q'.instance <- Establish_Instance();
16.         q'.manip <- TRUE ;
17.         Manip_Success <- FALSE;
18.         mark q' as visited;
19.       if q' is stable
20.         mark q' is visited;
21.       if LEGAL(q',q) then
22.         install q' in T with a pointer toward q
23.         INSERT(q' ,OPEN);
24.       if q' = q_g then SUCCESS <- true;
25. if SUCCESS then
26.     return the backtracked feasible path
27. else return failure;
```

Fig. 5. The STABLE_BFP algorithm for global path planning.

6. Global Planning with Multiple Motion Abilities

Given the initial and goal configurations (q_i and q_g) and motion abilities of a digital actor, our planner is supposed to generate a feasible motion plan including the global path and the chosen locomotion along the path that can bring the actor to reach the goal. The planning algorithm, called STABLE_BFP, used in our planner is shown in **Fig. 5**. The algorithm is a variation of the best-first planning algorithm commonly used for path planning in a low-dimensional space [7]. All candidate configurations are stored in a list called OPEN, and the best configuration defined according to some user-specified metric is retrieved for further exploration by the FIRST function (line 5) in each search loop. Similar to the A* algorithm, the metric used in FIRST is defined as the accumulated cost of getting to the current configuration and the estimated cost of getting to the goal. The cost to the goal is estimated by the potential field built from the goal in the reachability map. The cost of getting to the current configuration is defined as the cost of energy consumption for the current locomotion. For the motion ability of moving

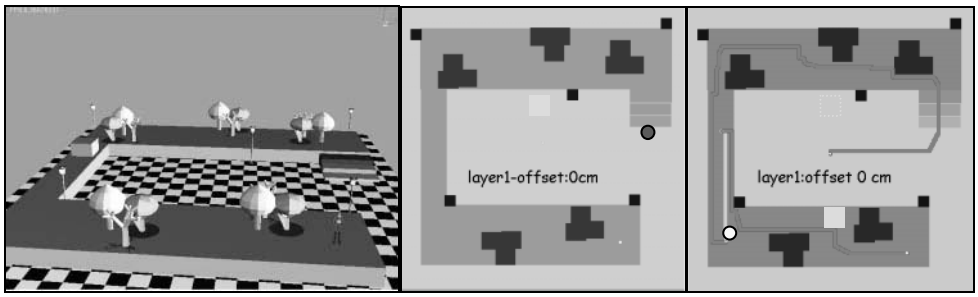

Fig. 6. An illustrative scenario requiring an obstacle to be move along a long hallway (a) 3D view (b) height map and obstacle location (c) path for moving the obstacle along the hallway.

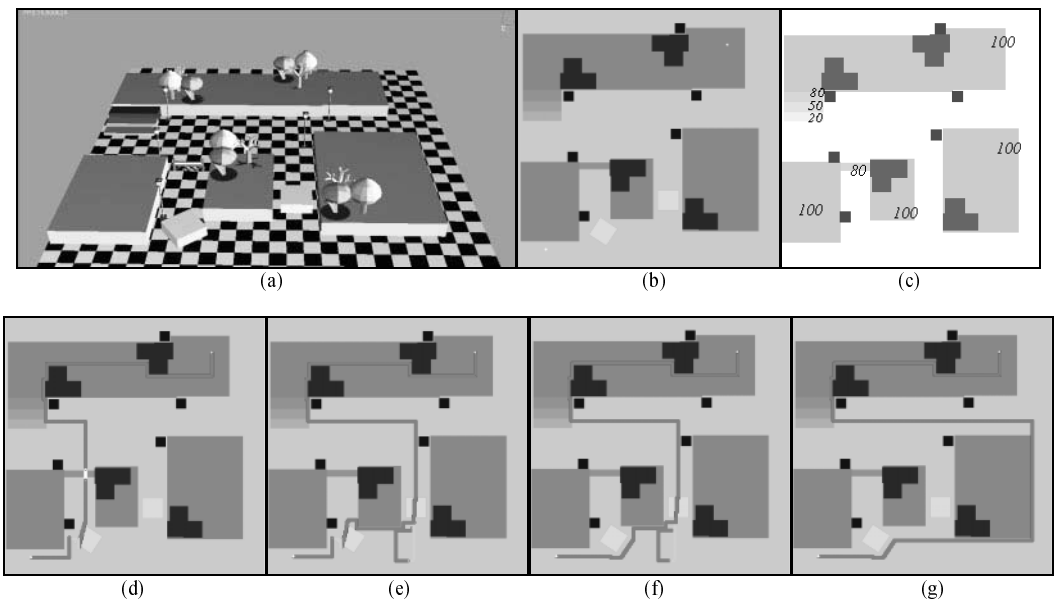

Fig. 7. (a) Inclined view of the experimental scene, (b) and (c) height maps, (d)-(g) resulting paths under different user preference conditions.

obstacles, a constant additional cost is added after each trial (of Δt time) in moving the obstacle in a planning instance. Thus, a planning instance can continue after some time if no other easier paths are found.

A few subroutines in the STABLE_BFP procedure need further explanations. When a configuration is regarded as a triggering point, a planning instance will be created via the Establish_Instance() routine (line 15) and the status of the triggering point will be set to "manipulation." When a configuration with the manipulation status is retrieved through the FIRST operation, its associated planning instance will continue to run for some given period of time. If the result of this run is successful, a new search space with updated obstacle configurations will be created via the Create_Searchspace() routine (line 9) and searched in parallel with the original search space.

For a configuration whose status is not manipulation, we will visit its neighbors and check their legality via the LEGAL() subroutine. Similar to [3], a configuration is defined as legal if it is unvisited, collision-free, and temporarily stable. The stability of a given workspace is computed according to the heights of neighboring cells. Three types of unstable cells are currently defined: *border*, *gap*, and *barrier*, and a stability counter for the allowable number of continuous cells of the same type is kept for each configuration. A configuration is considered illegal if the stability counter exceeds some given maximal value.

7. Experimental Results

The planning system described above has been implemented in Java and tested on a regular PC with Intel 1.66 GHz CPU and 1024 MB RAM. The resolution of the grid in the workspace is 256×256. The height of the digital actor is 160 cm, and the maximal step height is 40 cm. The range of heights that the actor can leap over is between 70 and 95 cm, and the maximal depth for leaping is 100 cm.

We use the scene in **Fig. 6** to test the ability of the actor in moving an obstacle for a long distance in a layered environment. The scene consists of two layers and the movable obstacle and the initial configuration of the actor (small dot at the lower-right corner) are on the second

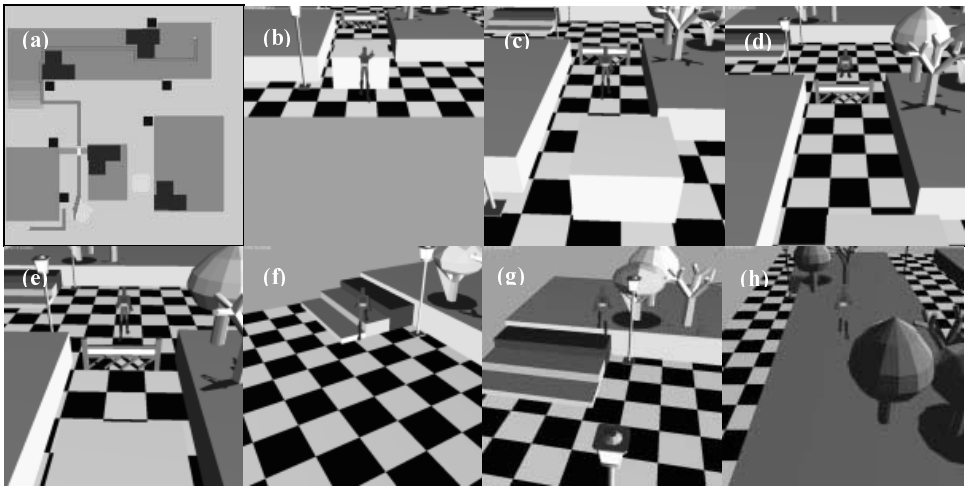

Fig. 8. Snapshots of the motions used by the actor to reach the goal.

layer while the goal configuration is on the first layer. The only way that the actor can get down to the first layer is by walking through the narrow hallway on the left and then stepping down the stairs. However, a movable obstacle is blocking the hallway. The planner succeeds in generating a motion for the actor to pull the obstacle back along the hallway until it can pass. The resulting path is shown in **Fig. 6(c)**.

Another example scene is shown in **Fig. 7(a)** to illustrate that the generated path may vary according to user preference. Two square movable obstacles are shown in light grey in **Fig. 7(b)** and the height of each object in the scene is shown in **Fig. 7(c)**. Four sets of preference parameters are used to test this scene. In **Fig. 7(d)** (case A), the user prefers shorter path and thus a path requiring moving an obstacle and leaping over a fence is generated. In **Fig. 7(e)** (case B), the preference parameters are the same as case A expect for that the weight for the leaping motion is reduced. In **Fig. 7(f)** (case C) and **7(g)** (case D), the weights for moving obstacles are slightly and greatly reduced, respectively. The actor may end up making a long detour as in the last case. This is also the path that can be generated by the approach in [5] because there is no separated connected component in freespace. In contrast, our unified planning approach is more flexible in taking advantage of various motion abilities. The overall computation times for these four cases are 484, 1672, 1750, and 985 ms, respectively. Snapshots of the actor executing various motions to realize the generated path for case A are shown in **Fig. 8**. The actor first pulls the obstacle out to enter the hallway and leaps over a fence afterward. Then, it turns to step onto a stairway to the second floor where the goal is located.

8. Conclusion

Based on our previous work on enabling humanoid digital actor to move with various motion abilities on a layered environment, we have extended a digital actor to have two more motion abilities: leaping over obstacles and moving obstacles. The planning for leaping motion can be treated with an OPENING operation, which is very similar to the CLOSING operation for the ability of striding over deep gap. Unlike previous work in planning movable objects, we have proposed a unified approach to generate motion plans that can take all motion abilities, including moving obstacles, into account at the same time. The experimental results reveal that versatile motions can be generated interactively for a humanoid actor according to user's preference on these abilities.

References:

[1] 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.
[2] Z. Shiller, K. Yamane, and Y. Nakamura, "Planning Motion Patterns of Human Figures Using a Multi-Layered Grid and the Dynamics Filter," Proc. of 2001 IEEE Int. Conf. on Robotics and Automation, pp. 1-8, 2001.
[3] T. Y. Li and P. Z. Huang, "Planning Humanoid Motions with Striding Ability in a Virtual Environment," Proc. of the 2004 IEEE Int. Conf. on Robotics & Automation, pp. 3195-3200, 2004.
[4] E. D. Demaine, M. L. Demaine, and J. O'Rourke, "PushPush and Push-1 are NP-hard in 2D," Proc. of the 12th Annual Canadian Conf. on Computational Geometry, pp. 211-219, 2000.
[5] M. Stilman and J. Kuffner, "Navigation Among Movable Obstacles: Real-time Reasoning in Complex Environments," Proc. IEEE Int. Conf. on Humanoid Robotics (Humanoids'04), 2004.
[6] J. Schwartz and M. Sharir, "On the 'piano movers' problem II, General techniques for computing topological properties of real algebraic manifolds," Advances in Applied Mathematics, 4, pp. 298-351, 1983.
[7] J. Latombe, "Robot Motion Planning," Kluwer Academic Publishers, 1991.
[8] J. Barraquand, L. Kavraki, J. C. Latombe T. Y. Li, and P. Raghavan, "A Random Sampling Scheme for Path Planning," Int. J. of Robotics Research, 16-6, pp. 759-774, 1997.
[9] T. Y. Li, P. F. Chen, and P. Z. Huang, "Motion Planning for Humanoid Walking in a Layered Environment," Proc. of the 2003 Int. Conf. on Robotics and Automation, 2003.
[10] P. C. Chen and Y. K. Hwang, "Practical path planning among movable obstacles," Proc. of IEEE Int. Conf. on Robotic Automation, pp. 444-449, 1991.
[11] G. Wilfong, "Motion Planning in the Presence of Movable Obstacles," Proc. of ACM Symp. Computational Geometry, pp. 279-288, 1988.
[12] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," Int. Journal of Robotics Research, 15-6, pp. 533-556, 1996.

[13] R. C. Gonzale and R. E. Woods, "Digital Image Processing," Second Edition, Prentice Hall, pp. 224-244, 1985.
[14] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, "Computational Geometry: Algorithms and Applications," Second Edition, Chapter 7: Voronoi Diagrams, 2000.

Name:
Yueh-Tse Li

Affiliation:
Graduate Student, Computer Science Department, National Chengchi University

Address:
64, Sec. 2, Zhi-Nan Rd. Taipei, Taiwan

Brief Biographical History:
2007- M.S. Computer Science Department, National Chengchi University

Name:
Tsai-Yen Li

Affiliation:
Professor, Computer Science Department, National Chengchi University

Address:
64, Sec. 2, Zhi-Nan Rd. Taipei, Taiwan

Brief Biographical History:
1995- PhD. Mechanical Engineering Department, Stanford University
2007- Visiting Professor, Stanford University

Main Works:
- "An Incremental Learning Approach to Motion Planning with Roadmap Management," Journal of Information Science and Engineering, 23(2), pp. 525-538, March, 2007.

Membership in Academic Societies:
- Association for Computing Machinery (ACM)
- The institute of Electrical and Electronics Engineers (IEEE)
- Taiwanese Association for Artificial Intelligence (TAAI)