Appear in Proceedings of the 8th International Symposium on Smart Graphics, 2008

Real-Time Camera Planning for Navigation in Virtual Environments

Tsai-Yen Li and Chung-Chiang Cheng

Computer Science Department, National Chengchi University,
64, Sec. 2, Zhi-nan Rd., Wenshan, Taipei 116, Taiwan, R.O.C.
{li, g9401}@cs.nccu.edu.tw

Abstract. In this work, we have developed a real-time camera control module for navigation in virtual environments. With this module, the tracking motion of a third-person camera can be generated automatically to allow a user to focus on the control of an avatar. The core of this module consists of a motion planner that uses the probabilistic roadmap method and a lazy update strategy to generate the motion of the camera, possibly with necessary intercuts. A dynamic roadmap specified relative to the avatar is updated in real time within a time budget to account for occlusions in every frame of the control loop. In addition, the planner also allows a user to specify preferences on how the tracking motion is generated. We will use several examples to demonstrate the effectiveness of this real-time camera planning system.

Keywords: Real-Time Camera Planning, Probabilistic Roadmap, Intelligent Camera Control, Budget-based Planning

1. INTRODUCTION

Real-time 3D games are becoming a popular application of virtual environments in our daily life. Through graphics rendering and interactions, a user can immerse into realistic virtual scenes that are difficult to experience in the real life. In a typical virtual environment such as World of Warcraft and Second Life, a user uses control devices such as keyboard and mouse to navigate and interact with objects or other users in a virtual world. In these virtual environments, a user usually expects to have the control of how to navigate through the environment but may not want to be overwhelmed by the complexity of cumbersome controls provided by the limited devices available on a desktop computer. The quality of a navigation experience usually is highly affected by how the camera is controlled. Thus, how to design an intelligent mechanism for camera control that can ease the burden of the user has been a research topic that has attracted much attention in the past years [10].

Typically, there are two types of camera control that are commonly adopted by 3D games: *first-person view* and *third-person view*. In a first-person control mode, the camera is attached to the eye of the avatar or some short distance behind the avatar. Therefore, moving the avatar would automatically transport the attached camera. However, the view is always limited to the front of the avatar, which may not be desirable in some applications. On the other hand, the control from the third person view

detaches the camera from the avatar and provides a more flexible way to view the avatar as well as the surrounding environment. The camera in this control mode can track the avatar from a distance or perform intercuts when appropriate. Although this kind of control is more flexible, it also presents an extra burden for the user to take care of the control of the camera in addition to the avatar. Therefore, it is highly desirable for the computer to take care of the control of the camera such that the user can concentrate on the control of avatar. There has been much research on designing camera control module in a game [11][15] but few can plan camera motions in real time based on the predicted motion of an avatar.

In this work, we focus on the problem of providing an effective third-person control of the camera. We aim to design a real-time motion planner for camera to track the predicted motion of the avatar. The generated camera motion should avoid occlusions from obstacles and follow cinematographic idioms [1] as much as possible. With this automatic camera control module, we hope that the user can concentrate on controlling the motion of the avatar while maintaining a high-quality view. After reviewing related work in the next section, we will describe how we model the planning problem and why we choose a probabilistic roadmap approach in Section 3. In Section 4, we will describe how we modify the planner to take intercuts into account. Then, we will demonstrate the effectiveness of our planner by the several examples from our experiments. We will then conclude the paper with future extensions.

2. RELATED WORK

There has been a significant amount of research into virtual camera control in recent years. According to how the problem is modeled and solved, we can roughly classify them into three categories: *algebraic system*, *constraint satisfaction*, and *planning*.

Algebraic System: In this type of systems, camera control is formulated as an algebraic problem whose solution is the desired camera position [5]. With this approach, the solution can be computed quickly but may lack flexibility. In [9], the authors have developed a high-level language called Declarative Camera Control Language (DCCL) to describe various cinematographic idioms. A compiler has also been developed to convert the camera specifications into parameters for an algebraic system to solve for the camera position.

Constraint Satisfaction System: This type of systems allows the users to specify their preferences on how to position the camera as constraints [2]. Then the system models it as a constraint satisfactory problem and tries to solve for a solution or a set of possible solutions [4][8]. Some systems further model it as an optimization problem according to some criteria to search for a good solution from the set of feasible ones [3][11]. Due to the computational complexity, most of these systems are not suitable for real-time interactive systems such as games. In [11], the authors use an efficient occlusion computation technique to maintain frame coherence in real time. Our work differs from theirs in that we have used a roadmap-based approach to search for a coherent trajectory and adopted the concepts of time budget to ensure real-time performance.

Motion Planning System: This type of approaches was originated from Robotics for the generation of collision-free motions for robots. A good survey of algorithms

for this problem can be found in Latombe's book [13]. Due to the computational complexity of this problem, most of the existing planning algorithms were not designed for real-time applications. Nevertheless, some attempts, especially for graphics applications, have been made to generate the motions for digital actors or cameras on the fly [16][17][18]. Like our approach, [16][18] also used the Probabilistic Roadmap Method (PRM) to compute camera or character motions. A probabilistic roadmap usually is built in a precomputation step to make the search in the query step efficient. However, due to the long precomputation time, this approach usually only works for static scenes.

3. REAL-TIME CAMERA PLANNING

3.1. Problem Description

Controlling a camera from a third person view is a challenging task because it is too complex for the user to control the avatar and the camera at the same time. The easiest way to position the camera is to fix the camera behind the avatar in the avatar's local coordinate system. However, there is no guarantee that the generated view will not be occluded by obstacles. The kind of occlusions usually creates undesirable interruptions on game playing and should be avoided whenever possible. Some games try to detect the occluded obstacle and change it to semi-transparent while others attempt to increase the inclined angle of the camera but without guarantee of success. Another common way is by positioning several cameras at known locations in a virtual environment. When an avatar enters a specific region, the camera is switched to a most appropriate predefined location. Although this method is computational efficient, it has the drawbacks of limited views and being scene-dependent.

We think that a third-person camera should follow the avatar and adjust its relative position and angle with respect to the avatar to achieve a good and coherent view. The camera should be controlled by the computer to satisfy hard constraints as well as soft constraints. The hard constraints include that the camera should not collide with obstacles and the view to the avatar should not be occluded by the objects in the environment. The soft constraints include that the view angle and view distance should satisfy the preference of a user as much as possible. Although an intercut is allowed to avoid occlusion when necessary, it should follow the idioms in cinematography [12] whenever possible. In addition, the available planning time for each frame should be limited by some desired frame rate (such as at least 15 frames per second) for interactive applications. Therefore, it is crucial to design an efficient planning algorithm to generate the motion of the camera within a given time budget in order to interleave motion planning and graphics rendering without causing jerky control.

3.2. Definition of Configuration Space

In this paper, we use motion planning technique to generate camera motion in real time.. When treated as a free-flying object, a camera have six degrees of freedom: position (x, y, z), and orientation (roll, pitch, yaw). However, due to the curse of di-

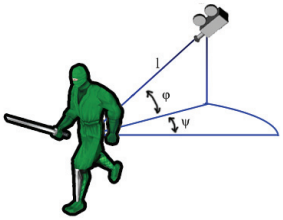

Fig. 1. Representation of camera configuration (l, ϕ, φ)

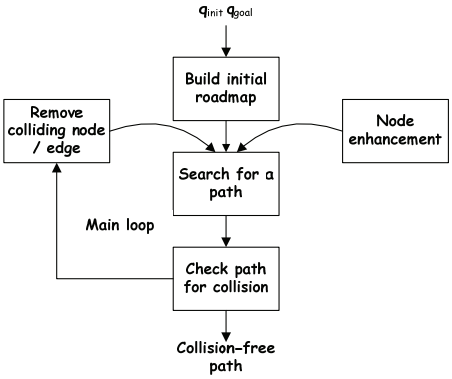

Fig. 2. Flowchart of the Lazy PRM algorithm

mensionality for the motion planning problem, searching the underlying six-dimensional space directly could be too complex to do in real time. Nevertheless, a regular camera rarely uses the roll DOF because of the disorder that it may cause. In [14], the authors only use three positional DOF to define a camera configuration because they assume that the camera always pointed to the avatar and position it at the center of the view. Similarly, we consider the position of the camera only but we used the relative coordinate frame with respect to the avatar in a polar coordinate system defined by distance l, horizontal angle ψ, and vertical angle φ, as shown in Fig. 1. Although both representations are equivalent, the relative coordinate system provides a more intuitive way for defining constraints or preferences.

3.3. Construction of Probabilistic Roadmap

The planning algorithm that we have used in our system is the Probabilistic Roadmap Method (PRM), one of the most popular methods in recent years for motion planning problems in high-dimensional configuration space. It has the advantage of being probabilistic complete and can be constructed incrementally, which is what we need in our real-time application. In a traditional PRM planner, an initial number of nodes are randomly generated in the free space of the configuration space, and nearby nodes are further connected by local paths, typically straight-line segments. For a given initial and a goal configuration, we first add them into the roadmap and then search for a continuous path in the roadmap connecting the two configurations. If one

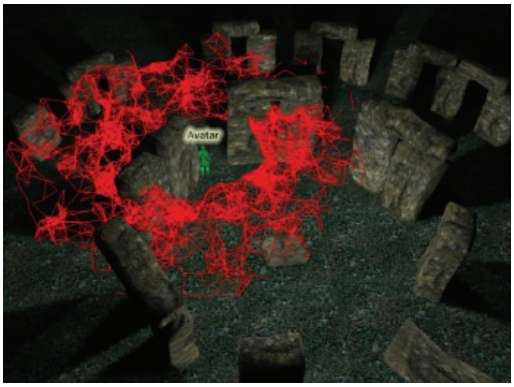

Fig. 3. An example of probabilistic roadmap computed with respect to the avatar

exists, the path is found; otherwise, more nodes can be added into the roadmap to enhance its connectivity.

In our real-time camera planning problem, we have used a roadmap update strategy similar to the one called Lazy PRM, proposed in [6][7]. The flowchart of the algorithm is shown in Fig. 2. A main feature of this approach is that it does not check collisions for a static roadmap in a preprocessing step like traditional PRM planners. Instead, the validity (legality) of a node or a link is checked only when the node or the link is under consideration during the search. Therefore, only the nodes that are related to finding a specific path at the moment need to be checked. Another difference of our planner is that the roadmap is described relative to the avatar and updated as the avatar moves, as shown in Fig. 3. Therefore, the roadmap is dynamic in nature even though there are no dynamic obstacles in the virtual scene. Nevertheless, since the motion of the avatar is continuous, the change of the roadmap is also incremental, which makes the real-time update of the roadmap feasible.

3.4. Camera Planning Algorithm

In a typical path planning problem, the inputs are an initial and a goal configuration in addition to the geometric description of the obstacles, and the output is a continuous path in the free space connecting the initial and the goal configurations. However, the camera planning problem differs from the classical motion planning problem in a few aspects. First, the goal is not clearly defined. In addition, there is no guarantee that a feasible path can be found in the given time budget. We will first describe the planning algorithm first and then describe the criteria that have been used to search for a feasible configuration.

In Fig. 4, we list the pseudo code for the time-budgeted camera planning algorithm. In this algorithm, we assume that a probabilistic roadmap with an appropriate number of nodes has been built in the configuration space. Instead of searching for the goal, the planner searches for a configuration that has the lowest cost for some criteria within a given time budget (TIME_LIMIT). The search starts from the current configuration and explore its neighbors in a best-first fashion. New legal neighbors will

Algorithm: Time_Budget_Based_Path_Search

```
1  OpenList ← { N_init }
2  Timer ← 0; VisitedNode ← null
3  while OpenList ≠ {} and Timer < TIME_LIMIT
4     N_c = extractMin(OpenList)
5     if isVisited(N_c)
6     then continue
7     else VisitedNode ← VisitedNode ∪ N_c
8     foreach neighbor N_s of N_c
9     if isConnected(N_s) = false
10       then continue
11    new_cost ← Cost_Function(N_s)
12    if cost[N_s] not nil and new_cost < cost[N_s]
13    then cost[N_s] ← new_cost
14       OpenList ← OpenList ∪ N_s
15       if cost[N_s] < cost(CUR_BEST_NODE)
16       then CUR_BEST_NODE ← N_s
17 return CUR_BEST_NODE
```

Fig. 4. Time budget based camera planning algorithm

then be kept in the priority queue called OpenList, where nodes are sorted by the cost function (line 11) to be described in the next subsection. When the time is allowed, the node with the lowest cost in OpenList is chosen for further expansion. When the time is up or the OpenList becomes empty, the configuration CUR_BEST_NODE is returned and then the next configuration for the camera to take can be determined by back-tracking the path to the initial configuration.

Since the avatar is not static, the validity of a node or a link in a roadmap is also dynamic at run time. We use a linear dead-reckoning algorithm to predict the motion of the avatar in the next few steps. We augment the roadmap by an additional time dimension such that, at any given time, we can check the validity of a node according to this prediction but in a lazy manner, which means that we will validate a node or a link only when it is necessary. Although linear extrapolation may not be the best way to predict the motion of an avatar when the time is far in the future, the potential errors may not be critical since they can be recovered quickly when the avatar's motion is updated in the future.

3.5. Design of Cost Functions

The quality of a camera view is a subjective matter although there exist cinematographic principles that can be used for evaluation. In this work, we have defined a cost function composed of three components to allow the user to specify their preferences. The first component, f_1, is composed of more subjective parameters that the user prefers when viewing the avatar, as shown below.

$$f_1(q) = w_h \cdot d_h(q) + w_a \cdot d_a(q) + w_l \cdot d_l(q), \quad (1)$$

where the three difference functions are:

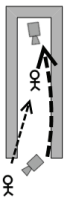

Fig. 5. Illustration of entering a deadend when intercut is needed

- **Height difference** (d_h): the difference between the current height and the ideal height of the camera.
- **View angle difference** (d_a): the angle of the current camera from the line of interest (LOI), which is the direction right behind the avatar in the tracking mode.
- **Distance difference** (d_l): the difference between the current distance and the ideal distance from the avatar.

For a given configuration q, f_1 is computed according to these three functions and their corresponding weights specified by the user.

The second component f_2 is computed based on visibility as shown below.

$$f_2 = \sum_{i=1}^{n} \alpha^i O_{occ}(q, t+i), \tag{2}$$

where $C_{occ}(q, t)$ is a constant occlusion cost for q at a give time t, and α is a decaying factor ($0 < \alpha < 1$), and n is the number of time steps under consideration for q. In other words, a configuration will be checked for occlusion for a period of time with decreasing weights in the future.

The third component f_3 is the distance cost of camera configurations in the roadmap:

$$f_3(q) = \text{distance}(q, q_s), \tag{3}$$

where q_s is the start camera configuration in the current search. This component allows the user to specify his/her preference of remaining unchanged in the current configuration.

The overall cost function $f(q)$ for a given camera configuration q is computed as a linear combination of these three components, where the weights (w_1, w_2, w_3) can also be chosen by the user to specify their emphasis on these criteria.

$$f(q) = w_1 \cdot f_1(q) + w_2 \cdot f_2(q) + w_3 \cdot f_3(q) \tag{4}$$

4. Consideration of Camera Intercuts

Due to the time constraints and unpredictability of avatar motions, the real-time camera planner presented in the previous section does not guarantee to find an non-occluded view and may take the camera to a difficult situation (such as the example shown in Fig. 5) where a legal camera configuration with continuous movement is not possible in the future. In this case, an intercut that allows the camera to jump to an-

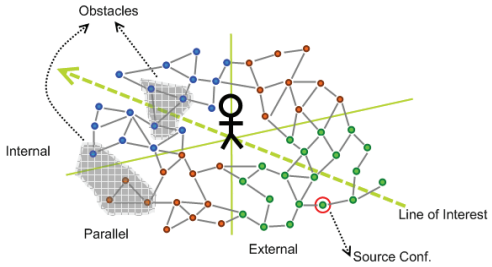

Fig. 6. Classification of nodes in the roadmap according to LOI

other appropriate location might be more desirable. In this section, we will describe how we account for this type of camera operation in the current planning algorithm.

In order to account for intercuts, we need to consider more than one camera. We can plan two cameras with two different sets of cost functions at the same time and then switch between them when the current camera is trapped in a difficult situation. However, the planning time will double in this case. Instead, we propose to use the concept of virtual links in the roadmap to consider both cameras in the same data structure. A virtual link is a temporary link from the start configuration to another disconnected configuration. The virtual links are constructed every time before the search starts. A virtual link is considered legal only if it satisfies the following conditions:

- The other end of a virtual link must be a legal node (non-occlusive and collision-free).
- Both ends of a virtual link must lie in the same side of the Line of Interest, which is defined as the heading direction of the avatar in this case.
- The other end of a virtual link must lie in a region that is far enough from the start configuration; otherwise, the user may experience more a jump cut instead of an intercut.

In order to facilitate the selection of the other end of a virtual link, we divide the roadmap into six different regions with three regions at one side of LOI as shown in Fig. 6. These regions (internal, parallel, and external) are defined according to the view angles relative to the heading orientation of the avatar. A valid virtual link must have their end points from two different regions, and the distance between the two ends must be above some threshold.

After a given number of virtual links are computed for the start configuration, the search can proceed as before except for that the cost of performing an intercut is made higher than regular movements. We modify the cost function f_3 to take this case into account. The modified f_3 is as follow.

$$\begin{aligned} f_3(q) &= C_{cut}, && \text{if a virtual edge} \\ &= \text{distance}(q, q'), && \text{otherwise} \end{aligned} \tag{6}$$

The movement cost C_{cut} for an intercut should be higher than the regular movement cost, and its value should be a time-dependent variable. In our design, we have used the following formula to define C_{cut}.

$$\begin{aligned} C_{cut} &= C_{cut}^{max}, && \text{right after the intercut} \\ &= \max(C_{cut}^{min}, C_{cut} - 1) && \text{otherwise,} \end{aligned} \tag{7}$$

Fig. 7. Inclined view of the test scene and a sample path taken by the avatar

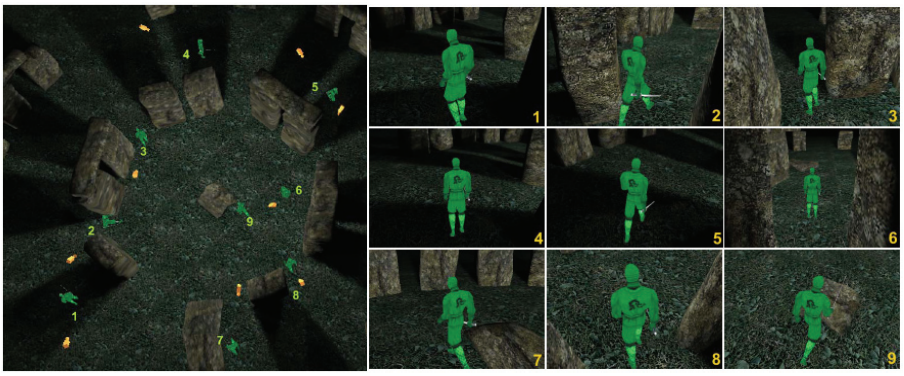

Fig. 8. Snapshots in part (b) are taken from locations at part (a)

where C_{cut}^{max} is the cost right after the intercut is performed. The cost will decrease gradually as the time passes by.

5. Experimental Results

We have developed the camera planning module in C++ in an experimental system based on the Object-Oriented Graphics Rendering Engine (OGRE) 3D graphics engine [1]. A collision detection module with simple hierarchical bounding boxes has also been implemented to detect the possible collisions of the avatar and the camera with the obstacles in the environment. In addition, to simplify computation, we have used four rays shooting from the camera to the avatar to detect occlusions. All experiments were conducted on a regular PC with a Pentium 4 3.0GHz CPU and 1GB of RAM. The allocated maximal time budget for planning computation in each frame is 65ms but actually the frame rate is maintained at around 60fps in all experiments. A constant number of 900 nodes are maintained for the dynamic roadmap occupying a region with a radius of around one fourth of the workspace of the virtual scene.

5.1. Basic Experiments

We have used a test scene in Fig. 7 to see if the camera can move to avoid occlusions with the obstacles in the environment. In this experiment, the main focus is

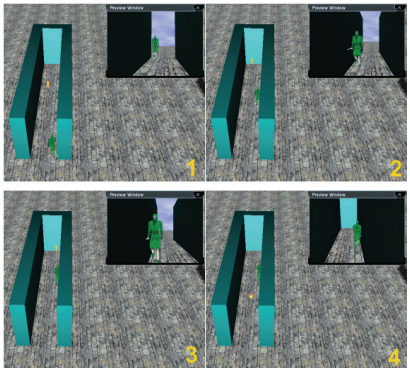

Fig. 9. Snapshots of the camera undergoing an intercut

on occlusion, and thus no special preference has been specified. As shown in Fig. 7, the scene consists of a circular array of huge stones that are likely to block the camera view when the avatar moves around the stones. In order to repeat the experiments for different parameters, we record the path of the avatar ahead of time (as shown in the right of Fig. 7) and replay the motion as a simulation in the experiments for different camera parameters. In Fig. 8(b), we show the snapshots of the views acquired from the simulation experiment at various locations defined in Fig. 8(a). In general, the camera can successfully track the avatar except for some short period of time when the avatar turned very sharply. In this case, the view was slightly occluded.

We also have designed the scene in Fig. 5 to intentionally trap the camera inside the dead end. In this case, the only way for the camera to avoid being occluded or getting into the obstacles is by an intercut. The experimental result is shown in Fig. 9. In the first snapshot, the camera entered the corridor first with an external view. As the avatar moved closer to the dead end of the corridor in the second and the third snapshots, the camera had fewer and fewer legal configurations that the camera can move to without jumps. Therefore, an intercut view was selected eventually by the planner, as shown in the fourth snapshot, to avoid bumping into the walls.

5.2. Experiments on Prediction Depths

The depth of the prediction for avatar motion that can be reached within a given time budget affects the final result. We have done an experiment to study this factor. The test scene is a maze-like environment where the camera needs to track the avatar closely to avoid occlusions shown in Fig. 10. The prediction of the avatar's future motion is also crucial for avoiding occlusions. We have used five different sets of preference parameters (exp1-exp5) to run the experiments with four different values of prediction depths (0, 1, 2, 3 seconds). These five different sets of parameters were chosen based on the following combinations: 1) no special preferences; 2) a middle shot from a fixed distance; 3) a close shot from an external view; 4) a middle shot from an external view; and 5) a long shot. At the right of Fig. 10, we show the plot of percentage of time that the view is occluded during the navigation for four prediction depths with five different preference settings. We found that prediction of the avatar's

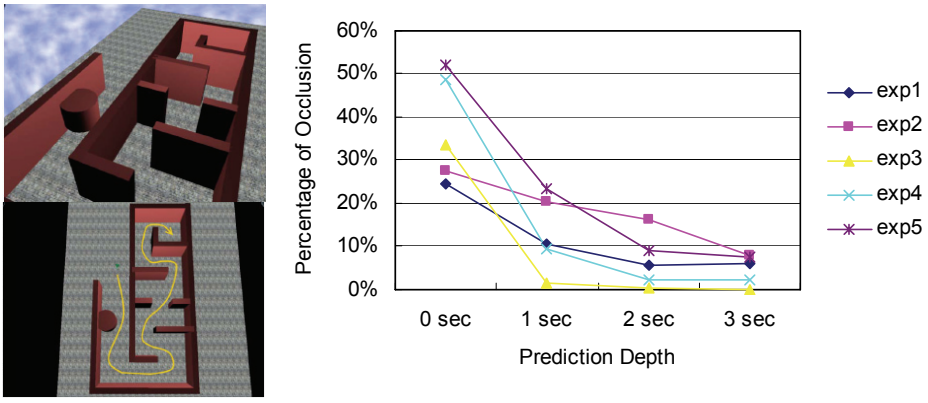

Fig. 10. Test scene and the experimental results of how prediction depth affects the percentage of occlusion time during several runs of navigation with different preference settings

motion indeed decreased the percentage of occlusion time for all preference settings but the effect degrades quickly when the prediction depth is longer than one second. This result suggests that using the concept of time budget and motion prediction to plan camera motions is an effective way to automate the generation of camera motion, and the prediction depth does not need to be long.

6. CONCLUSIONS

It is highly desired in real-time applications such as 3D games to have the camera motions generated automatically in a third-person control mode. In this work, we have successfully used a lazy PRM method developed originally for robotics to maintain a dynamic probabilistic roadmap specified relative to the avatar. As the avatar moves, the roadmap is maintained in a lazy fashion so that the validity of the nodes and links are checked only when they are visited. We also have proposed to use virtual links to account for intercuts in the search for a legal and high-quality move. The user is allowed to specify their viewing preferences or styles through the parameters that we have designed. We have tested the planner in a real-time 3D virtual environment with different scenes, and satisfactory results have been obtained and reported.

In the future, we are planning to investigate the possibility of using more sophisticated motion prediction methods for the avatar to understand its effect on the resulting camera motions. We would also like to study how to extend the work to consider more sophisticated viewing objectives in addition to tracking the avatar during navigation. These viewing objectives may be implicitly or explicitly specified in the environment under the concept of semantic virtual environment.

7. ACKNOWLEDGMENTS

This research was funded in part by the National Science Council (NSC) of Taiwan under contract no. NSC 95-2221-E-004-015.

References

1. Arijon, D. Grammar of the Film Language. Hastings House Publishers (1976).
2. Bares, W., Gregoire, H. J. P., and Lester, J. C., Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds. In Proc. of the Tenth Conf. on Innovative Applications of Artificial Intelligence (1998).
3. Bares, W., McDermott, S., Boudreaux, C., Thainimit, S., Virtual 3D camera composition from frame constraints. In Proc. of the ACM Intl. Conf. on Multimedia, ACM Press, (2000) 177–186.
4. Bares, W., Thainimit, S., and McDermott, S., A Model for Constraint-Based Camera Planning. In Proc. of the 2000 AAAI Spring Symposium (2000).
5. Blinn, J. F. Jim blinn's corner: Where am I? what am I looking at? IEEE Computer Graphics and Applications, 8(4):76–81 (1988).
6. Bohlin, R. and Kavraki, L. E., Path planning using lazy PRM. In IEEE Int. Conf. on Robotics & Automation (2000), 521-528.
7. Bohlin, R. and Kavraki, L. E., A Lazy Probabilistic Roadmap Planner for Single Query Path Planning. In Proc. of IEEE Int. Conf. on Robotics and Automation (2000).
8. Bourne, O. and Sattar, A., Applying Constraint Satisfaction Techniques to 3D Camera Control. In 17th Australian Joint Conf. on Artificial Intelligence (2004).
9. Christianson, D. B., Anderson, S. E., He, L. W., Salesin, D. H., Weld, D. S., and Cohen, M. F., Declarative Camera Control for Automatic Cinematography. In Proc. of the Thirteenth National Conf. on Artificial Intelligence (AAAI' 96) (1996).
10. Christie, M., Machap, R., Normand, J. M., Olivier P., and Pickering J., Virtual Camera Planning: A Survey. In Proc. of the 5th International Symposium on Smart Graphics (2005).
11. Halper, N., Helbing, R., and Strothotte, T., A Camera Engine for Computer Games: Managing the Trade-Off between Constraint Satisfaction and Frame Coherence. In Computer Graphics Forum: Proceedings Eurographics 2001, 20(3):174-183 (2001).
12. He, L. W. and Cohen, M. F., and Salesin, D. H., The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing. In Proc. of the 23rd Annual Conf. on Computer Graphics and Interactive Techniques (1996).
13. Latombe, J., Robot Motion Planning, Klumer, Boston, MA (1991).
14. Li, T. Y. and Yu, T. H., Planning Tracking Motions for an Intelligent Virtual Camera. In Proc. of IEEE Int. Conf. on Robotics and Automation (1999).
15. Lin, T. C., Shih, Z. C., Tsai, Y. T., Cinematic Camera Control in 3D Computer Games. In The 12th Int. Conf. Central Europe on Computer Graphics, Visualization and Computer Vision (2004).
16. Nieuwenhuisen, D. and Overmars, M. H., Motion Planning for Camera Movements in Virtual Environment. In Proc. IEEE Int. Conf. on Robotics and Automation (2004).
17. Oliveros, D. A. M., Intelligent Cinematic Camera for 3D Games. Thesis, Univ. of Technology, Sydney Australia (2004).
18. Salomon, B., Garber, M., Lin, M. C., and Manocha, D., Interactive Navigation in Complex Environment Using Path Planning. In Proc. of the 2003 symposium on Interactive 3D graphics (2003).
19. OGRE 3D, http://www.ogre3d.org/