

Extracting Camera-Control Requirements and Camera Movement Generation in a 3D Virtual Environment

Hirofumi Hamazaki¹, Shinya Kitaoka¹, Maya Ozaki¹,
Yoshifumi Kitamura¹, Robert W. Lindeman², Fumio Kishino¹

¹ Graduate School of Information Science and Technology, Osaka University
2-1 Yamada-oka, Suita, Osaka, 565-0871, JAPAN
{hamazaki.hirofumi | kitaoka.shinya | ozaki.maya | kitamura | kishino}@ist.osaka-u.ac.jp

² Department of Computer Science, Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609, USA
gogo@wpi.edu

ABSTRACT

This paper proposes a new method to generate smooth camera movement that is collision-free in a three-dimensional virtual environment. It generates a set of cells based on cell decomposition using a loose octree in order not to intersect with polygons of the environment. The method defines a camera movement space (also known as Configuration Space) which is a set of cells in the virtual environment. In order to generate collision-free camera movement, the method holds a path as a graph structure which is based on the adjacency relationship of the cells, and makes the camera move on the graph. Furthermore, by using a potential function for finding out the force that aims the camera at the subject and a penalty function for finding out the force that restrains the camera on the graph when the camera moves on the graph, we generate smooth camera movement that captures the subject while avoiding obstacles. Several results in static and dynamic environments are presented and discussed.

Categories and Subject Descriptors

H.5.4 [Information Interfaces And Presentation]:
Hypertext/Hypermedia

General Terms

Algorithms, Design, Human Factors

Keywords

video game, virtual reality, camera control, path planning, octree, interactive system

1. INTRODUCTION

In recent years, applications like games and Second Life which allow a user to explore a three-dimensional (3D) virtual environment have emerged. These applications create camera movement such as translation, rotation, and zoom according to the character's movement. Also as in traditional camera movement, the camera is manipulated to keep following the character in a 3D virtual environment. This gives the character manipulator a deeper realistic sensation. But if the camera just keeps following the character, it sometimes may penetrate a building or go under the ground in the virtual environment, or it may not capture the character due to obstacles which exist between it and the character. In order to avoid these cases, we have to set control requirements relating to camera movement, depending on the goal of the virtual environment.

On the other hand, the study of avoiding collisions with obstacles has been carried out in robotics, for example, PRM (Probabilistic Roadmap Method) [1, 2] and RRT (Rapidly exploring Random Trees) [3, 4]. Also, The cell decomposition methods which create a roadmap graph to avoid collisions with obstacles are famous ones. The cell decomposition methods divide an environment into equal-sized cells [5, 6] or varying-sized cells based on an octree [7, 8], and pick a set of cells which is collision-free with the obstacles. Then, these cells are connected with nearby cells, and the roadmap graph is created. By moving on the roadmap graph, robot's movements become collision-free.

Therefore, we propose methods for efficiently automating the process of determining legal camera movement based on camera-control requirements by extending the cell decomposition methods. By automatically creating a roadmap graph that is a collision-free space between the camera and the obstacles from the virtual environment based on hierarchical cell decomposition using a loose octree [9] not just an octree, then making the camera follow the subject smoothly in this space, we can automatically generate the camera movement that fulfills the control requirements.

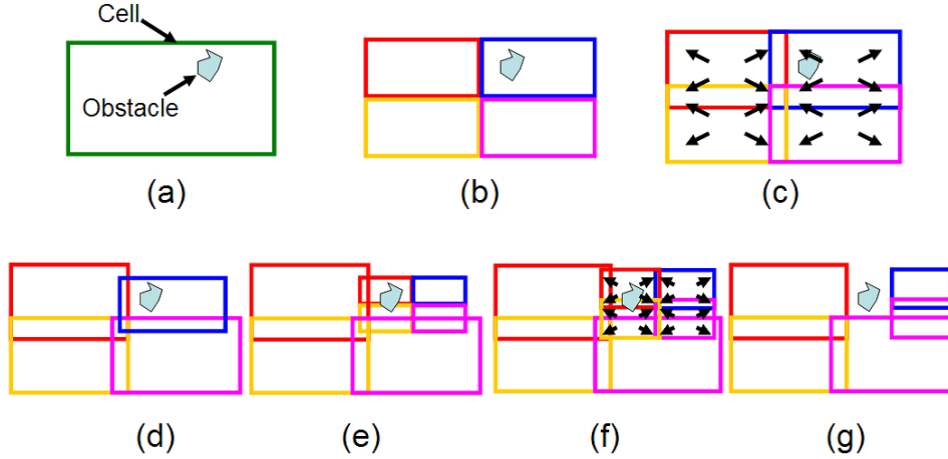


Figure 1: Procedure of creating a camera movement space in the case of a maximum octree depth of two. (a) Make a cell that contains the obstacle; (b) Create four more new cells by evenly dividing the cell (actually create eight new cells). New cell's colors are different (the top left is red, the top right is blue, the bottom left is yellow, the bottom right is pink); (c) Enlarge the new cells respectively; (d) Detect the intersection between each cell and the obstacle, add the cell to the roadmap graph when the cell does not intersect with the obstacle, and restore the size of the cell when the cell intersect with the obstacle; (e) Create four new cells again by evenly dividing the cell that intersect with the obstacle; (f) Enlarge new cells again; (g) Remove the cells that intersect with the obstacle because depth of division reaches the maximum octree depth.

2. A CAMERA CONTROL METHOD

In the camera movement generation process, it is important to figure out the camera movement space. However, because this space changes depending on the user's intention, we define a camera movement space which is a collision-free space between the camera and obstacles in the environment. This prevents the situation in which the camera penetrates obstacles and captures neither the subject nor the positional relationship between the subject and the environment. We define camera-control requirements as :

- The camera does not penetrate obstacles.
- The camera keeps following the subject.

To fulfill this requirements, we extract a camera movement space from the environment. Here, we describe a method of extracting camera-control requirements and a method of generating camera movement that meets the control requirements.

Also, we support 3D virtual environments as follows :

- The number of cameras is one.
- The number of subjects is one.
- The subject's movement is unknown.
- The subject's orientation is not considered.
- The environment is constructed from polygon data only.

This enables us to focus on the movement of a chase camera.

2.1 Extracting Camera-Control Requirements

We represent a camera movement space as a set of axis-aligned cells based on cell decomposition, and then create a camera roadmap graph using the adjacency relationship of the cells. In this way, when a subject is contained in or

intersecting with a cell, we are assured of seeing the subject if the camera reaches the cell because a cell is a convex hull. The method's procedure (Figure 1) is described as follows :

1. Define a maximum octree depth.
2. Make an axis-aligned cell that contains all obstacles. This cell is the octree's root.
3. Create eight new cells by evenly dividing the cell in the x-axis, y-axis, and z-axis (eight octants). Because the cell contains obstacles, we need to find out the cells that do not contain any obstacles.
4. Enlarge the new cells respectively. As a result, overlapped regions of cells arise.
5. In each cell, detect the intersection between the cell and the obstacles.
 - Intersection case
 - If depth of division has not reached the maximum octree depth, restore the size of the cell, create eight new cells by dividing the cell again, and go to step 4.
 - If depth of division reaches the maximum octree depth, stop.
 - Non-intersection case
 - The cell is added to the roadmap graph.

In this way, a camera movement space based on hierarchical cell decomposition using a loose octree is created. A loose octree does not only divide the region into eight cells but also enlarges each cell and creates overlapped regions of each cell. With overlapped regions, the cells are mutually adjacent and we can create a camera movement space as a roadmap graph.

In addition, to speed up the intersection detection between the cells and obstacles, we preliminarily manage the obstacles using an AABB (Axis Aligned Bounding Box) tree [10].

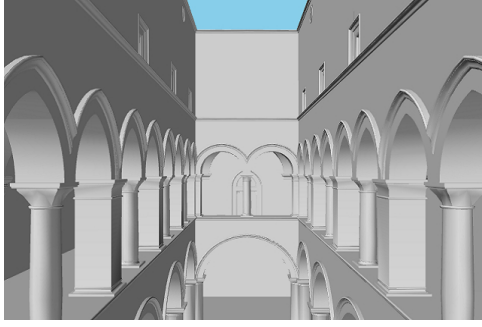


Figure 2: Shrine architecture (model by Marko Dabrovic)

2.2 Generating Camera Movement

When the cell that overlays the camera is different from the cell that overlays the subject, we define these cells as the starting point and the ending point. By making the camera move from the starting point to ending point on the roadmap graph, we generate camera movement that fulfills the camera-control requirements. For finding the path on the roadmap graph, we use the A^* algorithm [11] because we can arbitrarily configure a cost function used for path finding, and also can definitely find a path which has minimum cost. In this paper, we configure the cost function as the Euclidean distance between the subject and the cells in the camera movement space. Also, the A^* algorithm is only used when the cell that overlays the subject changes due to movement of the subject for reducing the cost of the path finding.

The camera moves along the path that was found by the A^* algorithm. However, the path is not composed of line segments but a cell range, so we have to create a trajectory for the camera. In this paper, we obtain a trajectory which is pursuant to Hooke's law by using a potential function for finding out the force that aims the camera at the subject and a penalty function for finding out the force that restrains the camera on the graph. This way, we generate camera movement that is smooth and collision-free.

2.3 Resulting Camera Movement

We applied the proposed method to an environment with a shrine architecture (Figure 2). In this environment, we created a camera movement space, and then generated camera movement that captures a moving subject in the environment.

In Figure 3, we show the camera movement space that was created in the environment. The set of cells represented by the green line segments is the camera movement space. Figure 4 shows a camera trajectory created in the architecture. The purple cell is the ending cell of path finding. The green cell range represents a path in the camera movement space, which was found by the A^* algorithm. The red line that extends to the side of the subject is the camera trajectory. In this example, using the Euclidean distance between the subject and the cells of the camera movement space as a cost function, we sought a path which has the shortest movement distance of the camera. Also, the camera trajectory shows that we did find the trajectory within the cells of the camera movement space.

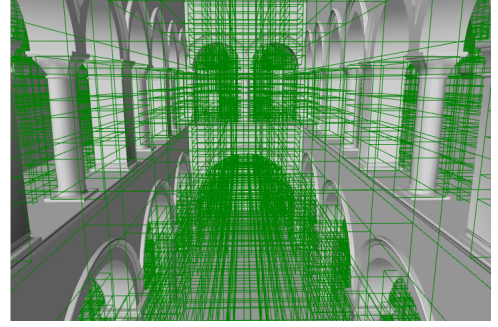


Figure 3: A created camera movement space in the environment

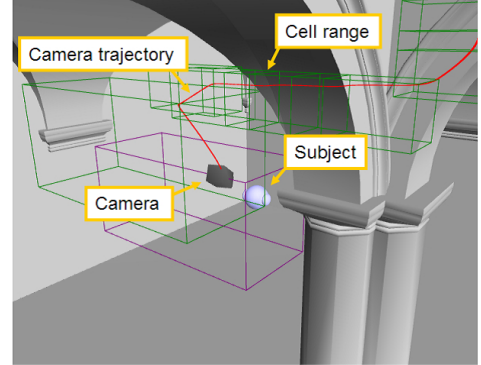


Figure 4: Camera trajectory when the subject goes behind a pillar

3. DYNAMIC ENVIRONMENTS

Thus far, we have considered a static virtual environment without movable obstacles. Next we describe how to adapt our proposed method to a virtual environment with movable obstacles (a dynamic virtual environment). We add the constraints below for a dynamic environment.

- The number of obstacles ranges from dozens to hundreds.
- Obstacle movement is unknown.
- Obstacle shape is a rectangular solid.

In Figure 5, we show a dynamic virtual environment. The orange objects are moving obstacles. To adapt our method,

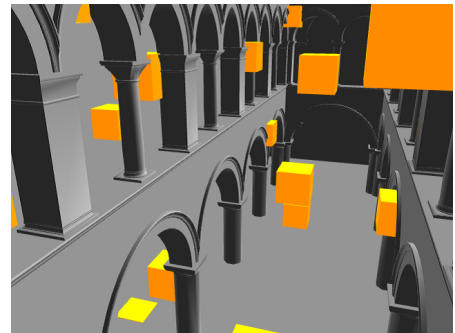


Figure 5: Dynamic virtual environment

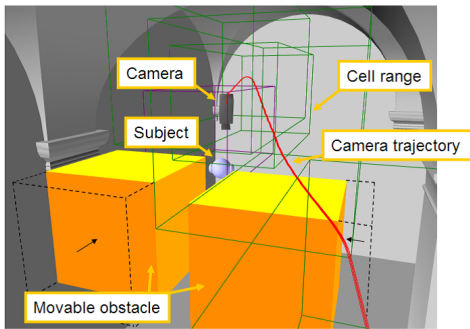


Figure 6: Camera trajectory in the dynamic environment

we slightly modify the method for extracting camera-control requirements and the method for generating camera movement.

First, to change the method for extracting camera-control requirements, we add the following processes.

- By detecting the intersections between the camera movement space created in a static environment and movable obstacles, cells intersected with the obstacles are excluded from the camera movement space.
- If excluded cells no longer intersect with movable obstacles, the cells are restored to their previous camera movement space.

Also, we perform the intersection detections in real time. By changing the camera movement space as a result of the intersection detections, a new camera movement space is created. However, if there exist many movable obstacles in the environment, the cost for intersection detection for all obstacles is high. Therefore, we surround the camera with a bounding box. Intersection detection between the obstacle and the camera movement space is performed when the obstacle intersects with this bounding box.

Next, we modify the method for generating camera movement. Because the camera movement space is changed in real time to adapt to a dynamic environment, we have to reflect this change in camera movement generation process. In a static environment, we use the A^* algorithm only when the cell which overlays the subject changes. However, in a dynamic environment, we use the A^* algorithm at regular intervals. Also, because the camera movement space is represented as a set of cells, there are margins for moving the camera in the path. We therefore center a sphere with a certain size around a movable obstacle, and generate a repulsive force along the normal of the sphere's surface. In this way, every time the obstacle gets close to the camera, a camera movement is reflected and avoiding obstacles is more smoothly performed. We show a camera trajectory created in the dynamic environment in Figure 6.

4. CONCLUSIONS AND FUTURE WORK

In this paper we have described the method to generate camera movement based on camera-control requirements. First, we defined a camera movement space as a collision-free space between a camera and obstacles in an environment, and camera-control requirements as the camera movement

space. Next we extracted the camera movement space as a roadmap graph that is created by hierarchical cell decomposition using a loose octree. We then found the path between the camera and the subject within the camera movement space. By controlling the camera on the path by a potential function and a penalty function, we generated camera movement that captures the subject while avoiding obstacles. We also adapted the methods to an environment with movable obstacles.

In this paper, we have considered chase camera movement. In the future we will try to make a camera movement space by using a K-D tree and compare a K-D tree with an Octree to examine which one is more suitable for environments. In addition, we plan to generate cinemagraphic camera movements which include panning that rotates the camera and cutaways that change the camera position. Our future work also includes other camera-control requirement investigation.

5. REFERENCES

- [1] L. E. Kavraki, P. Švestka, J. C. Latombe and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE*, Vol. 12, Num. 4, pp. 566-580, 1996.
- [2] L. K. Dale and N. M. Amato. Probabilistic roadmaps-putting it all together. *Robotics and Automation, IEEE*, Vol. 2, pp. 1940-1947 2001.
- [3] J. J. Kuffner, Jr and S. M. LaValle. RRT-connect : an efficient approach to single-query path planning. *Robotics and Automation, IEEE*, Vol. 2, pp. 995-1001, 2000.
- [4] S. M. LaValle, M. S. Branicky and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *Journal of Robotics Research*, Vol. 23, Num. 7-8, pp. 673-692, 2004.
- [5] C. W. Warren. Fast path planning using modified A^* method. *Robotics and Automation, IEEE*, Vol. 2, pp. 662-667, 1993.
- [6] K. Azarm and G. Schmidt. Integrated mobile robot motion planning and execution in changing indoor environments. *Intelligent Robots and Systems, IEEE*, Vol. 1, pp. 298-305, 1994.
- [7] M. Herman. Fast, three-dimensional, collision-free motion planning. *Robotics and Automation, IEEE*, Vol. 3, pp. 1056-1063, 1986.
- [8] J. Rosell, C. Vázquez and A. Pérez. C-space decomposition using deterministic sampling and distances. *Intelligent Robots and Systems, IEEE*, pp. 15-20, 2007.
- [9] M. Deloura, ed. Section 4.11: Game programming gems I. Charles River Media, 2000.
- [10] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, Vol. 2, Num. 4, 1997.
- [11] P. E. Hart, N. J. Nilsson and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE*, Vol. 4, Num. 2, pp. 100-108, 1968.