Appear in *Proceedings of 2002 15th IPPR Conference on Computer Vision, Graphics, and Image Processing*, Taiwan, 2002

An Assisted User Interface for 3D EXAMINE Mode

Yu-Te Lin (林祐德) and Tsai-Yen Li (李蔡彥)
Computer Science Department, National Chengchi University
64, Sec.2, Chih-Nan Road, Taipei, Taiwan 11605, ROC
{s8712, li}@cs.nccu.edu.tw

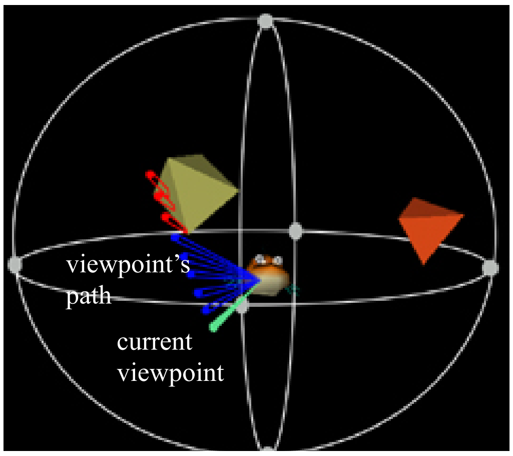
Abstract

Despite the rapid development of 3D technologies, controlling a viewpoint in a 3D environment remains a non-trivial or even difficult task for a novice user. Previous researches have shown that motion planning techniques can be use to assist user navigation in the so-called WALK mode for VRML browsers. In this paper we propose to use a motion-planning approach for assisting viewpoint control in the EXAMINE mode for 3D browsers. This approach uses a Reconfigurable Random Forest (RRF) structure to learn the 3D environment progressively as the viewpoint moves along. Unlike previous work that simplifies the workspace by projecting the environmental obstacles into 2D polygons, we have to deal with 3D motions and 3D obstacles. Efficiency of the planner becomes crucial in designing such an intelligent user interface with an acceptable frame rate. We have implemented and incorporated the planner into a Java3D-based VRML browser and tested it with several example scenarios with satisfactory results.

1. Introduction

3D applications are becoming prevalent on personal computers with the rapid development of graphics hardware. In addition to the traditional applications such as Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM), more appealing applications in entertainment such as games and virtual shopping are emerging. Although the frame rate for 3D displays is increasing, it remains a great challenge for a novice user to control a viewpoint in a 3D scene with a 2D device such as a mouse[1].

Controlling 3D viewpoint is difficult because the constraints that a 3D browser might have imposed on the interface to increase reality. An interface without any constraints might appear easy to control but the viewpoint might get into the inside of an object. This situation might confuse the user because the display could turn into a blank screen suddenly because of no back light inside an object. To overcome this problem, a typical browser will support the function of collision detection that can be turned on to avoid penetrating an object. However, due to the limited range of view frustum, it is very often that a user gets stuck at a corner of the scene without seeing the object obstructing the viewpoint. It usually takes several maneuvers before the user can escape this kind of situation.



(a)

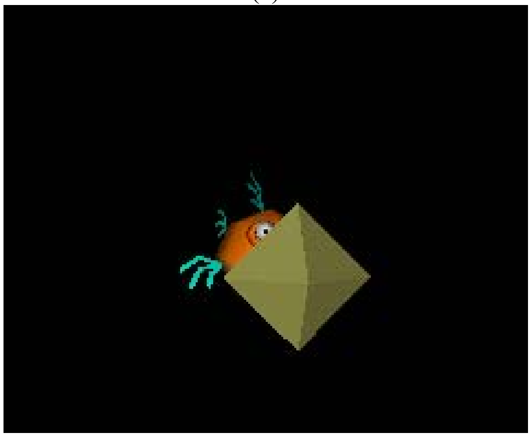

(b)

Figure 1. (a) Current viewpoint, target object, and trace of the line of sight occluded by an obstacle (b) an undesirable obstruction by an obstacle.

3D viewpoint controlling problems appear in many 3D applications. A typical 3D application, such as VRML[15] browser, provides several modes of viewpoint control. These modes include WALK, EXAMINE, FLY, etc. The WALK mode assumes that the viewpoint moves on a horizontal plane of fixed height while the FLY mode does not confine where the plane should be. The EXAMINE mode allows a user to focus its sight on an object and rotate the viewpoint around it as illustrated in Figure 1. Most previous researches on

3D intelligent user interface focus on the first two modes because they are most frequently used. Nevertheless, the EXAMINE mode plays a complementary role that is also important when the user would like to examine an object in details. For example, the work in [3] uses the FLY mode for navigation control, but whenever the virtual laparoscope finds the cancer cells, the doctor needs to switch to the EXAMINE mode to examine the cells. 3D virtual mart is another application where customers might need to examine products they have found when their avatars navigate to an interesting site.

Several previous researches have addressed the viewpoint control problem. Most of them focus on WALK and FLY modes. For example, in our previous work, we have used the approaches of motion planning [8][9] and artificial force fields[10] to assist a user in controlling the viewpoint in the WALK mode. The experiments in the work show that one can improve the navigation efficiency by as much as seventy-three percents for a given scene. However, we have not seen similar work on the EXAMINE mode yet. In the WALK mode, because the viewpoint is constrained on a horizontal plane, one can usually assume that the objects in the environment can be simplified by projecting them into 2D polygons in the workspace. This step greatly simplifies the computational complexity of the problem, especially for the planning approaches. However, for the EXAMINE mode, this assumption is no longer valid. Therefore, maintaining great efficiency becomes a crucial factor for practicality of such an approach.

In this paper, we propose a novel approach to address the viewpoint control problem in the EXAMINE mode. The approach adopts a motion planner to generate collision-free motions for the viewpoint automatically when it is going to be obstructed by an obstacle. The motion planner takes an incremental approach to build the roadmap required for planning a collision-free path. It updates the roadmap progressively as the viewpoint moves along.

For the rest of the paper, we will first review previous research related to our work in the next section. In Section 3, we will give a more detail description of the viewpoint control problem we are going to address. In Section 4, we will propose a progressive version of the RRF algorithm to solve our problem. Then we will present our implementation of the intelligent interface and some experimental results. Finally, we will conclude the paper with some future work.

2. Related Work

The work related to our research can be found in the field of intelligent user interface design and artificial intelligence. We will review 3D user interface design first and then narrow down to the issues of delegation-based and direct-manipulation-based intelligent user interface design.

2.1. 3D computer-human interface design

Many researches have been undertaken to invent efficient ways to communicate with a computer and to evaluate the effectiveness of these interfaces. Among these interfaces, being capable of interacting with virtual 3D environments has been considered as an important trend for future computer-human interfaces. The VR-types of interfaces such as Head Mounted Display HMD), 3D tracking devices, data gloves, force feedback joysticks, haptic devices, etc, are all good examples of such interfaces. New metaphors such as eyeball in hand, and flying vehicle in hand have also been proposed and tested[1]. It is reported that most users like the idea of eyeball-in-hand metaphor in the context of virtual space exploration. However, the great challenge comes when we are asked to manipulate a 3D virtual scene only with a regular 2D mouse on a desktop computer. Some work has been carried out to design intuitive interfaces for controlling 3D rotations with 2D devices[4][13].

2.2. Delegation-based intelligent user interface

Although many intelligent user interfaces have been proposed in the literature, most of them are not for 3D manipulation[12]. Exceptions include using motion-planning techniques to provide task-level controls. For example, Drucker and Zeltzer[2] argue that a task-level viewpoint control is crucial for exploring virtual scenes such as virtual museums since the users should be allowed to concentrate on scene viewing instead of be distracted by low-level navigation controls. Kuffner [5] also utilizes fast path-planning techniques to assist real-time animations. However, most of these approaches use geometric reasoning techniques as a tool to delegate control. They use a third-person view to specify the desired tasks, which is very different from the first-person view commonly used in the direct manipulation metaphor.

2.3. Direct-manipulation-based intelligent user interface

In our previous work, we have proposed to incorporate motion-planning techniques into the control loops of user interface design so that assisting motions can be generated automatically. In [9], we used an incremental motion planner, called RRT, to extend the method to consider navigation in large virtual environments. However, this work only applies to the WALK mode for architectural walkthrough applications. Xiao and Hubbold[16] utilize force fields to guide navigation in 3D environment, and Hong et al.[3] apply a potential field for interactive navigation in the application of

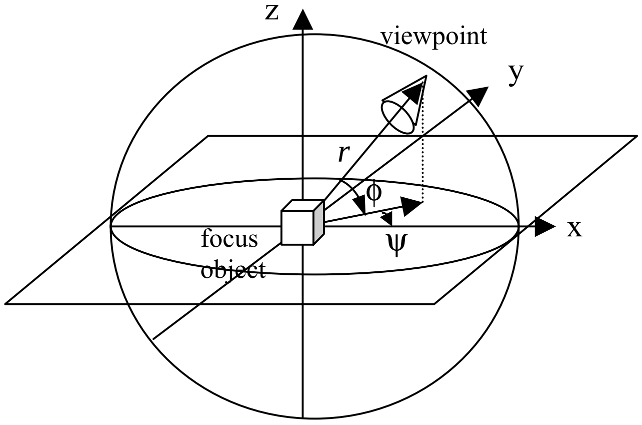


Figure 2. Control parameters in the EXAMINE mode

medical operation. In [10], we also proposed an adaptive force field that can be used to help the user reduce the chance of colliding with obstacles. Experimental results show that the motion planning method and the force field method are complementary and can be used together to further improve navigation efficiency.

3. Problem Description

3.1. Obstructions in the EXAMINE mode

We assume that in the EXAMINE mode, the focus of the view (used as a rotation center) is determined by the browser as the user clicks on the display. Then the motion of the viewpoint will be constrained on the spherical surface of some radius *r*. The configuration of the viewpoint can then be defined by two parameters (ϕ, ψ) in the spherical coordinate system as shown in Figure 2. The line segment connecting the viewpoint and focus object is called *line of sight*. Although the viewpoint could have rotation around the line of sight, the principles of Cinemagraphy suggest a camera to remain leveled at all time. Therefore, we will assume the motion of the viewpoint will be controlled with these two parameters via mouse input. We will also assume that the browser will provide an interface for the user to zoom in or zoom out while remaining in the EXAMINE mode.

We also assume that we are given a geometric description and configurations of the objects in the virtual environment. These objects could be one of the focus objects as well as obstructing obstacles in the EXAMINE mode. We say that a viewpoint is in an undesirable or illegal configuration if the viewpoint or its line of sight collides with any other objects in the environment. When the collision detection function is turned on, a browser should detect this kind of collision and prevent the viewpoint from penetrating into or obstructed by other objects.

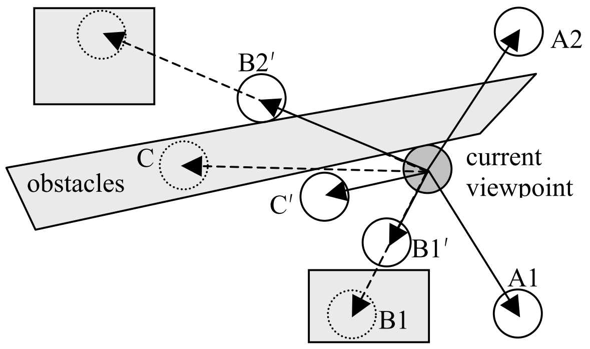


Figure 3. Possible situations for goal configurations and their modifications

3.2. The goal configuration prediction problem

As a user control the viewpoint's motion via mouse input, the new configuration of the viewpoint is updated according to the horizontal and vertical components of the input vector. These two components are used as moving velocities for updating the two parameters (ϕ, ψ) mentioned in the previous subsection. If the light of sight collides with any other objects at the new configuration, the viewpoint will get stuck at the configuration unless we advise it to move away.

In our previous work[8][9], we have proposed to use a motion planner to generate a collision-free path that will guide the user through these difficult areas. In order to form a valid path-planning problem, we have to define a legal goal configuration first. In our previous work, we have classified the possible goal configurations into three categories: A, B, and C, as shown in Figure 3. In the A category, no modification is needed for forming a legal goal configuration while the viewpoint can be directly modified along the line of sight in category B. In the third category, no direct modification is possible and therefore a legal goal configuration can be generated only if we change the line of sight along object boundary.

4. Progressive RRF_CONNECT Planner

4.1. The planning problem

In an EXAMINE operation, when the light of sight collides with an obstacle in the environment, we will call a path planner to generate a path connecting the current configuration to the predicted goal configuration defined as in the previous subsection. Since the planner is incorporated into the control loop of the user interface, the efficiency of the planner is crucial. Many efficient path-planning algorithms have been proposed in the literature. However, we have not seen efficient 3D planners that can be used in real-time applications. Fortunately, the problem we have at hand has several characteristics that we take advantage of. First, as a user enters the EXAMINE mode, he/she will remain in

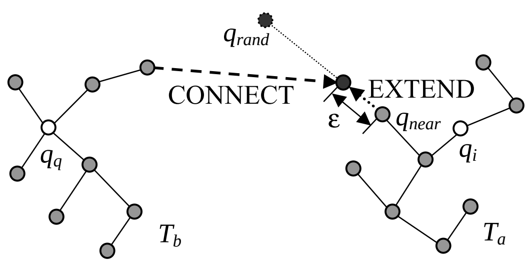


Figure 4: Two RRT's use EXTEND and CONNECT to merge into one tree

the mode for some time before it performs a zooming operation or enters other modes. Therefore, the configuration space of our problem can be restricted to a spherical surface for some time. Second, when a user zooms in or out, the restricted configuration subspace could be similar since the obstacles in these subspaces should change smoothly. In the following subsections, we will propose to use an algorithm called Reconfigurable Random Forest (RRF) planner to generate the viewpoint path. However, before introducing the algorithm, we have to first introduce its most basic data structure called Rapidly-Exploring Random Tree (RRT).

4.2. The RRT Structure

In [9], we have used RRT as the data structure for incremental path planning in large environments. The RRT structure was proposed by Lavalle[7] to solve difficult motion planning problems such as the kinodynamics problems. The RRT structure is one form of roadmap built in freespace (collision-free portion of a configuration space). Its main difference from traditional probabilistic roadmaps is on that RRT grows outward from a tree although configurations are sampled randomly in the freespace. As depicted in Figure 4, the growing process starts by selecting a random configuration, q_{rand}, as the growing direction. The nearest configuration, q_{near}, in the current RRT to q_{rand} is determined, and a new configuration, q_{new}, that is ε-distance away from q_{near}, is computed and added into the RRT. This process is called EXTEND.

In [6], an efficient single-query planning algorithm, called RRT-Connect, uses RRT as the main data structure to connect the given initial and goal configurations (q_i and q_g). Two RRT's, rooted at q_i and q_g, respectively, are used to connect to each other. At each step of the growing process, a random configuration q_{rand} is sampled in the freespace. One RRT uses the EXTEND procedure to add to itself a new configuration, q_{new}, while the other RRT uses another procedure called CONNECT to grow (EXTEND) toward q_{new} as much as possible. If CONNECT can bring the RRT to reach q_{new}, then the two RRT's have been successfully connected and a feasible path is returned. Otherwise, the two RRT's swap to allow them to grow in the other direction.

```
MERGE_RRTs($T_A$, $q_{new}$)
1  for each $T$ in forest
2    if ($T \neq T_A$)
3      if (CONNECT($T$, $q_{new}$) = Reached)
4            REVERSE_PARENT($T$, $q_{new}$);
5        forest.remove($T$);
6  return;
RRF_CONNECT($q_i$, $q_a$, $K$)
1   $T_i$.init($q_i$); $T_g$.init($q_a$);
2   forest.add($T_i$); forest.add($T_g$);
3   MERGE_RRTs($T_a$, $q_a$);
4   MERGE_RRTs($T_i$, $q_i$);
5   if ($T_i$.tree_id = $T_g$.tree_id)
6    return PATH($q_i$, $q_a$);
7   for $k$ =1 to $K$ do
8    $q_{rand} \leftarrow$ RANDOM_CONF( );
9    if (EXTEND ($T_i$, $q_{rand}$) $\neq$ Trapped)
10     MERGE_RRTs ($T_i$, $T_i.q_{new}$);
11     if ($T_i$.tree_id = $T_g$.tree_id)
12        return PATH($q_i$, $q_a$);
13   SWAP($T_i$, $T_a$);
14  return Failure;
```

Figure 5: The RRF_ CONNECT algorithm

4.3. The RRF_CONNECT algorithm

Many efficient roadmap-based algorithms have been proposed in the literature. This type of planner usually consists of two phases: *learning phase* and *query phase*. In the learning phase, the planner usually needs to build a representative roadmap for the freespace through random sampling. This phase usually takes much more time than the query phase. In our case, the viewpoint control problem is not formed until the user has picked the focus object, and the configuration space is not determined until then. However, one cannot build a complete roadmap at run time while maintaining an interactive frame rate for the graphical user interface. Therefore, we need a planner that can learn the freespace and build a roadmap progressively as the viewpoint move along. In [11], we have proposed such an algorithm called RRF_CONNECT.

Figure 5 shows the RRF_CONNECT planning algorithm. The algorithm assumes a global data structure called *forest* to store the list of currently maintained trees. A main subprocedure used in RRF_CONNECT is called MERGE_RRTs. This procedure tries to connect each tree in the forest, except for the currently considered tree T_A, to the designated new configuration, q_{new}, via the CONNECT procedure. The tree is merged with T_A if the connection is successful. In the RRF_CONNECT algorithm, after the trees rooted at q_i and q_g are initialized, we first call the MERGE_RRTs procedure to see if we can connect the two configurations to the forest without adding addi-

```
DUPLICATE_RRF(r)
1   F = GET_RRF(r)
2   if (F is null)
3       F_near = NEAREST_RRF(r)
4       COPY_ROADMAP(F_near, F);
5   if (F is null)
6       F.init()
7   else
8       VALIDATE_ROADMAP(F);
9   return F
```

Figure 6. The DUPLICATE_RRF algorithm

tional configurations. If this is not successful, a randomly sampled configuration, q_{rand}, will be selected to extend T_i, and MERGE_RRT will then be called again. This process will repeat until the T_i and T_g are merged (success) or a predefined maximal number of sample configurations are reached (failure).

4.4. Multi-layer roadmap for zooming operation

Many browsers support manipulation modes other than the EXAMINE mode. With the RRF-CONNECT algorithm proposed in the previous subsection, we are able to build the roadmap in the freespace as the viewpoint moves on a spherical surface. However, as one zooms in or out to get a better view of the focus object, the configuration space as well as the roadmap built for it needs to be reconstructed. Since our RRF-CONNECT planner is a progressive planner that can be used as a single-query planner, one can always start building the roadmap from scratch. However, we think the RRF roadmaps for nearby layers are more likely to be similar since most of the time the obstacles possess continuity in shape.

Thus, we propose to build a new layer of RRF roadmap by duplicating the roadmap from a neighboring layer if exists. The idea of this procedure is shown as pseudocode in Figure 6. Whenever a new layer is going to be constructed, we check its neighboring layers to see if we can copy the existing layer over. If not, we simply build the RRF from scratch. If there exist nearby RRF's, we perform a validation process on the duplicated RRF. In the validation routine, we have to remove invalid nodes in the RRF, which could result in splitting a tree in RRF into several subtrees.

5. Implementation and Experiments

5.1. Implementation

We have implemented our ideas on a java-based VRML browser, which is an open source program that could be downloaded from [14]. The libraries we use include Java SDK and Java3D SDK. Most of our modi-

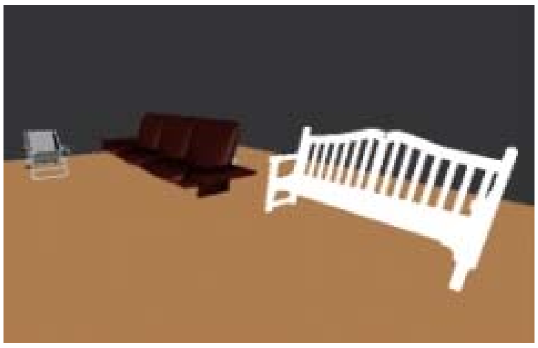


Figure 7. An example scene for testing

fication is on the input handling routines for the EXAMINE mode. We intercept the mouse input and predict where the user would like to move the viewpoint to. Whenever the goal configuration of the viewpoint is obstructed by obstacles, we will call the path planner to generate a detour collision-free path. We have fully implemented the path planner described in Section 4. The RRF layers for different radii are discretized according to some resolution specified by the user.

5.2. Line of sight collision detection

Collision detection is usually the most time-consuming routine for motion planners. It is also one of the most critical routines in our system because we demand real-time performance for our interactive application. In our case, we have chosen to perform some preprocessing to reduce the collision detection time at run time.

In the EXAMINE mode a configuration is considered illegal if the line of sight connecting the viewpoint to the focus object collide with other objects in the environment. Therefore, the most basic collision-detection routine is the check the interference between a line segment (line of sight) and object models. In order to reduce the processing time, we use a scan-conversion routine to discretize the object models into a 3D bitmap grid representing the workspace. A '1' cell represents an occupied cell while a '0' cell represents freespace. Consequently, we can reduce the collision-detection problem into check a line segment and a bitmap. This type of computation is much cheaper because collision detection becomes a sequence of table lookups for the points lying on the line segment.

In order to speed up the computation further, we compute a distance map, denoted by DM, to be used in the collision detection routine. DM is a 3D grid storing the shortest L1 distance from obstacles for each free cell in the workspace. Whenever we would like to perform a collision check between a bitmap and a line segment, we discretize the line segment into a sequence of points. Although we can check the whole sequence

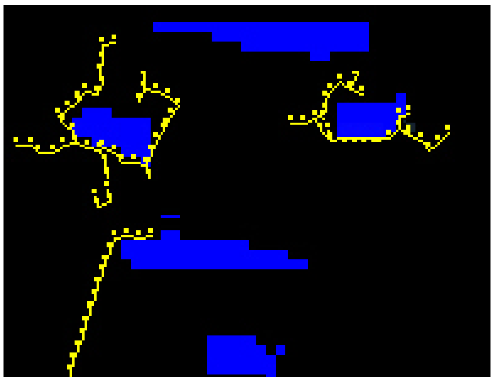


Figure 8. A portion of the incrementally built RRF in an unfold spherical configuration space

of points along the line segment, we hope that we only check a few points selectively to save time. The idea is that if the distance value for a cell in DM is *d*, then there is no need to check collisions for the next *d* points.

5.3. Experiments

This experiment is taken on the personal computer with PentiumIII800 CPU, NVIDIA GeForce2 MX display card, and 256Mb RAM. The scene for our experiments is similar to the one in Figure 7 about furniture objects. Figure 8 shows portion of a layer of RRF in an unfold spherical configuration space.

We run the experiments through two scenes. Scene 1 is the easier one because we make the obstacles are all located between the viewpoint and focus object. On the other hand, obstacles in scene 2 are located on the spherical surface in the EXAMINE mode. Since the obstructive objects on the spherical surface are harder to be seen in the view frustum, scene 2 is considered to be more difficult to work with.

We are not aware of any benchmark scenarios for comparing the efficiency of such intelligent user interface, and each user's control behavior could be very different. Nevertheless, the effectiveness of the assisting user interface has been observed by the subjects in our experiments. Since we have demonstrated the efficiency improvement of this kind of intelligent user interface, we will focus our experiments on the analysis of how the planner is used.

Table 1 shows the statistic data from experimenting the EXAMINE mode with the RRF-CONNECT planner. All experiments are run for about three minutes and 1500 steps. N1 shows the number of frames that can be updated per second, which mainly indicates the responsiveness of the interface as well as the difficulty of the scenes. In both cases, the frame rates are all acceptable for interactive use. N2 indicate that the average time spent for each planning can be incorporated into the control loop without causing significant sluggishness. N3, the path length, shows that most of the planning problems are not very difficult, and a short path suffice to help the user. N4 shows that the number of calls to the planner is only a small portion of the total number of steps. Most of the calls will succeed (N5) and some of the planned paths are canceled (N6). (We allow a user to cancel the planned path if he/she deviates input from the original one by some amount.)

Table 1. Statistic data of the RRF-CONNECT planner for different scenes.

	Scene 1	**Scene 2**
N1	17.17Hz	6.96Hz
N2	71.27 ms	54 ms
N3	5.74 steps	6 steps
N4	86 times	24 times
N5	63%	66%
N6	20.4%	25%

N1: update rate (Hz) for the interface control loop
N2: average planning time in ms.
N3: average length for a generated path.
N4: number of planning evoked.
N5: percentage (%) of planning evoked succeeds.
N6: percentage (%) of planning being canceled.

We have also done some experiments to measure the efficiency of the proposed multi-layer RRF approach in Section 4 and the collision detection method proposed in the previous subsection. We compare the average planning times for building RRF from scratch (91ms) and by duplicating neighboring layers (78ms). We found that the efficiency greatly depends on how long the user remains on a layer. In other words, the longer one can stay one a layer, the shorter the average planning time since much time has been invested on copying. As zooming is usually an independent operation, the system can use the time of switching modes to copy the roadmap. Therefore, the planning time at run time is actually much shorter. We also compare the time spent in collision detections by counting the table lookups on the 3D bitmap. The one without the helps of distance map will lookup about 4.4 times of the points examined with the speedup method. Since the collision detection routine is the most expensive routine in planning, this speedup contributes much to the interactivity of our approach.

6. Conclusion and Future work

We have successfully extended the intelligent 3D user interface control to the EXAMINE mode. The great challenge is from the efficiency of the planner in 3D environments. We have taken a progressive approach to distribute the planning cost along all the possible calls to the planner. This progressive approach is based the RRF-CONNECT algorithm and the multi-layer roadmap duplication approach. However, the longer the user uses the interface, the larger the RRF roadmap would be. Therefore, managing the roadmaps while maintaining a good coverage rate as proposed in [11] would be the most apparent future work. Moreover, predicting user's intention in the EXAMINE mode is another issue that can be further studies since the prediction method might require different treatments for different navigation modes.

References

[1] Chen, Mountford, and Sellen, "A Study in Interactive 3D Rotation Using 2D Control Devices, " in *Proceedings of Computer Graphics (SIGGRAPH88)*, 22(4):121-128, 1988.

[2] S. M. Drucker and D. Zeltzer, "Intelligent Camera Control in a Virtual Environment," *Graphics Interface'94*, pp. 190-199, 1994.

[3] L. Hong, S. Muraki, A. Kaufman, D. Bartz and T. He, "Virtual voyage: interactive navigation in the human colon," in *Proceedings of Computer Graphics (SIGGRAPH97)*, pp.27-34, 1997.

[4] M. R. Jung, D. Paik, D. Kim, "A Camera Control Interface Based on the Visualization of Subspaces of the 6D Motion Space of the Camera," in *Proceedings of IEEE Pacific Graphics'98, 1998.*

[5] J.J. Kuffner. "Goal-Directed Navigation for Animated Characters Using Real-Time Path Planning and Control," *Proc. of CAPTECH '98: Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, Geneva, Switzerland, Nov. 26-28, 1998.

[6] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient appraoach to signal-query path planning," In *Proceeding IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000.

[7] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Computer Science Dept., Iowa State University.

[8] T. Y. Li, and H. K. Ting., "An Intelligent User Interface with Motion Planning for 3D Navigation," *Proceeding of the IEEE Virtual Reality 2000 Conference*, March 2000.

[9] T. Y. Li, and C. C. Chang, "Path Planning with Incremental Roadmap Update for Large Environments," *Proceeding of 2001 the IEEE International Conference on Robotics and Automation*, May 2001.

[10] T. Y. Li, and H. C. Chou, "Improving Navigation Efficiency with Artificial Force Field," *Proceeding of 2001 IPPR Conference on Computer Vision, Graphics, and Image Processing*, Taiwan, 2001.

[11] T.Y. Li and Y.C. Shie, "An Incremental Learning Approach to Motion Planning with Roadmap Management," *Proceedings of International Conference on Robotics and Automation*, Washington, 2002.

[12] M. Maybury and W. Wahster (eds), *Readings in Intelligent User Interfaces*, Morgan Kaufmann: Menlo Park, CA.

[13] Neilson and Olsen, "Direct Manipulation Techniques for 3D Objects Using 2D Locator Devices," in *Proceeding Of the 1986 Workshop on Interactive 3D Graphics*, pp175-182, 1986.

[14] The Java3D and VRML task groups, http://www.web3d.org/TaskGroups/source/xj3d.html

[15] VRML97 International Standard, URL: http://www.web3d.org/technicalinfo/specifications/specifications.htm

[16] D. Xiao, R. Hubbold, "Navigation Guided by Artificial Force Fields, " in *Proceedings of the ACM CHI'98 Conference*, pp179-186, 1998.