Navigation Guided by Artificial Force Fields

Dongbo Xiao and Roger Hubbold

Department of Computer Science
University of Manchester
Manchester M13 9PL, United Kingdom
roger@cs.man.ac.uk

ABSTRACT

This paper presents a new technique for controlling a user's navigation in a virtual environment. The approach introduces artificial force fields which act upon the user's virtual body such that he is guided around obstacles, rather than penetrating or colliding with them. The technique is extended to incorporate gravity into the environment. The problem of negotiating stairs during a walk-through has also been investigated with the new approach. Human subjects were tested in experiments in which they experienced three different kinds of navigation: unconstrained, simple constrained and assisted by force fields. The results demonstrate that the force-field technique is an effective approach for effective, comfortable navigation.

KEYWORDS: 3D interfaces, virtual environments, collision avoidance, navigation, force fields.

INTRODUCTION

An important feature of virtual reality is the facility for users to move through a virtual environment in a natural and easily controlled manner. Natural navigation methods contribute to a sense of presence, cited by some researchers as a defining attribute of VR [1]. The illusion of presence can be lost through unnatural experiences during a walk-through. This can be caused by poor interactive metaphors or by experiences which are not consistent with the user's everyday understanding of the real world. Several attempts have been made to develop new metaphors for walking through virtual environments [2, 3, 4, 5]. However, intuitive metaphors are only able to solve part of the walk-through problem. The other part concerns how to provide a virtual environment with more realistic properties so that the user's walk-through can be more natural and comfortable. Unconstrained motion is sometimes desirable, indeed liberating, since less concentration is needed in the control of motion. However, the physical world is full of solid objects and we do not expect to penetrate these when we collide with them. Thus, in situations where realistic, real-world behaviour is required, more constrained navigation methods are called for.

In this paper, we present a technique for assisting a user's navigation in a virtual environment. The method we have devised incorporates several new features. We use *force fields* to guide a user past objects in the environment. This idea is combined with a simple, but efficient, form of *collision avoidance* which guarantees that users cannot pass through solid objects. The method also offers a straightforward way of incorporating *gravitational effects*, and effective assistance to negotiate stairs.

The technique we present is a general solution: it does not rely on any explicit (semantic) knowledge of the virtual environment, such as 'this is a staircase'. The technique for climbing stairs can also be employed to climb on top of other objects – a chair, for example – provided that it is low enough for the user to 'step' on. The algorithm can be implemented for a variety of primitive shapes as long as they are convex (non-convex shapes can be constructed by assembling convex primitives), and works well with complex and dense environments. It does not require any complicated interactive metaphors or devices, and can be used for immersive environments as well normal 3D workstations.

We present results from experiments we have performed which compare the new algorithm with two other methods: unconstrained navigation, and navigation using only collision detection. These demonstrate that the force-field technique is an effective approach for comfortable navigation.

RELATED WORK

One of the problems of unconstrained navigation, particularly in complex or cluttered environments, is that it is very difficult for a user to control a virtual body which has no mass or other natural attributes. When attempting to find one's way past objects without any haptic feedback, one is relying totally on visual cues, and this is doubly difficult

if the field of view is restricted – a common situation with many head-mounted displays. As a result, the user often encounters unnatural experiences, such as walking through objects like walls. Particularly confusing is the act of mistakenly walking backwards through a wall, when the world suddenly disappears from view. A direct approach to solving this problem is to make objects solid, that is to introduce some form of collision detection [6] to prevent users from passing through objects. While this avoids the more obvious problems, such as walking through a wall backwards, it does not make the actual task of steering a course through an environment any easier. It is still relatively easy to get stuck against small objects which are not in one's immediate field of view.

Consideration of this problem led us to the idea of using a force field around objects to guide the user. The idea was motivated by the work of Bouvier and Guilloteau [7], who used a flocking algorithm [8] for simulation of crowd control in a virtual environment. Force-field methods have been applied previously to path planning in robotics [9, 10, 11, 12]. In robotics, in addition to repulsive forces surrounding objects, an attractive force from a designated target plays an important role in 'pulling' the robot in the desired direction. However, generally, the robot's target is known in advance, so that a force field for the whole environment can be precalculated. With *interactive* navigation, the control of movement should respond to the user's input and this cannot be pre-planned. Instead of passively responding to the environment or following a given path, the user's intention during the walk-through may be arbitrary and varying, so a different approach is needed. Egbert and Winkler [13] proposed a force-field technique for computer animation, which used a vector field around objects to prevent collisions between them. However, interaction with a user was not considered. Furthermore, as we shall see later, a force field alone is not sufficient to prevent collisions with objects.

Since the work reported here was conducted, our attention has been drawn to the work of Jacobson and Lewis [14]. Our experiments and ideas are very similar to theirs, although they do not use force fields.

THE FORCE-FIELD GUIDED APPROACH

The fundamental idea in the new approach is that each object in a virtual environment is surrounded by a repulsive force field. Each field extends to a pre-determined distance – a *region of influence* – beyond which no force is applied. If the user is close enough to an object to be inside its region of influence, the object creates a repulsive force to resist any attempt of the user to get closer to the object in question. In this way, the force field around an object forms a barrier to prevent the user from getting too close to the surface of the object, and also assists the user to make effective movement in order to avoid the object.

The differences between the new approach, the unconstrained approach, and another which simply detects collisions, are illustrated in Figure 1 with a simple scene. Suppose the user is moving forwards and approaching an obstacle. The user will walk through and penetrate the obstacle with the unconstrained method (Figure 1a), or be stopped at the edge of the obstacle with the simple collision avoidance method (Figure 1b). With the guidance of a force field, the user will move slightly sideways, avoid the obstacle, and carry on going forward (Figure 1c).

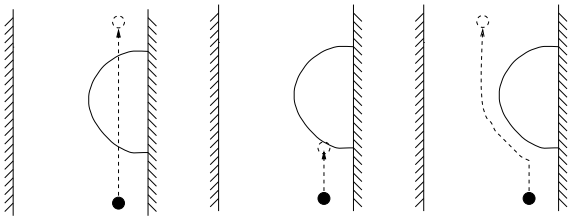


Figure 1: Three types of walk-through

User's Control of the Motion

The user's intended motion is specified interactively as a force, which is determined by the user's virtual body orientation, velocity and acceleration. The user's input is supplied with a 3D mouse containing a Polhemus Fastrak sensor. Inputs from the mouse can be mapped to suitable ranges to give easy control of direction and velocity. To move forwards, the user pushes the mouse forwards, while keeping a button depressed. This results in a forward force applied to the virtual body. Pushing the mouse further forward generates a larger force, causing acceleration. Pulling it backwards generates a backward force. Similarly, a sideways translation of the mouse results in a corresponding sideways force on the virtual body. Twisting the mouse about a vertical axis induces a rotational (steering) movement, permitting a change of direction.

These forward or backward, sideways and rotational movements can all be applied simultaneously, giving quite precise control. Releasing the button allows the mouse to be moved without generating any input forces. Thus, at any time, a user can stop moving by releasing the mouse button, and can restart with a new frame of reference by pressing the button again.

The Force Field around an Object

The force field exerted by an object is represented by a single force along the shortest path between the object and the user's virtual body. The magnitude of the force is a function of this path length.

Initially, we planned to use an inverse square law to control the force, but we discovered that the sudden increase in repulsion as the user approached objects made this method

difficult to control interactively. After experimentation, we finally opted for a linear force field, in which the repulsive force is inversely proportional to the shortest distance between the user's virtual body and the object in question. At the edge of the region of influence the force drops to zero. The force and distance parameters can be scaled to suit the user's virtual body size within the environment.

Computing the Resultant Force on the Virtual Body

If the user is not close to any objects – that is, if he or she is outside their regions of influence – then movement is completely controlled by the user. Otherwise, the next 'step' is decided by combining the force applied by the user with the forces exerted by nearby objects. We discovered that naively combining all of the forces, to obtain a resultant force upon the virtual body, led to cases where the result was uncomfortable for the user. For example, the mere presence of a nearby object could cause movement when the user actually wanted to stand still. Three rules were therefore derived to govern the force computation:

1. If the user does not apply any force to the virtual body, with the mouse, then it remains stationary. This is important as it means users are always completely in control of movement. You won't get pushed around if you happen to stand next to objects which apply a force, unless you also apply a force.

2. Objects only push *against* the direction in which the user tries to move. Thus, if you attempt to move forwards through an object, the movement will be resisted. The same will happen if you try to go backwards through something. But, if you stand with your back to a wall and then walk forwards, the wall will not exert a force.

3. In simple mechanics, component forces would normally be summed to produce a resultant force. Here, the effect of this would be that two adjacent objects would create a double force, repelling the user strongly. In regions of the environment with many small objects clustered together, unacceptably large forces would result. To prevent this, we resolve each separate object force into components in the direction of the user's input force – the direction he would move in the absence of any other forces, which we term the *major direction* – and the directions normal to this. Only the *maximum* positive and negative components in each direction are then considered. Any force from behind the user is ignored and the other components are resolved to produce a final direction and distance of movement.

There remains one final detail of motion control which must be addressed: the problem of oscillation. Consider the case where a user is approaching a wall. As the wall gets closer it exerts an increasing, repulsive force. If the wall is approached cautiously, the forward force applied by the user and the repulsive force from the wall reach a point of balance and movement stops. But if the wall is approached more rapidly, it is easy to pass beyond this equilibrium state, so that the force exerted by the wall greatly exceeds that applied by the user. When this happens, the user is repelled strongly and bounces back from the wall. This backward movement reduces the opposing force, and the user moves forwards again. Unless some kind of damping is introduced the user may end up in a state of oscillation. In practice, this can be considerably exacerbated by the user's natural reaction, which is often to move the mouse either forwards or backwards to try to bring the situation under control. Such oscillations are visually distressing and make precise control very difficult.

To counteract oscillations, we take the following steps. First, we identify the *major direction* of the user's motion. Note that this is not necessarily the direction in which the user's virtual body is pointing, because sideways as well as forward and backward movements are permitted. We discovered empirically that oscillation control is only necessary in the major direction. In fact, we do not wish to damp forces which are normal to this, as it is these which provide the predominant 'steering' effect from the environment.

Next, in the major direction, if the opposing force from an obstacle is greater than that applied by the user, we limit the magnitude of the net force in this direction. This is done by scaling (damping) the force difference according to the shortest distance to the obstacle, thereby reducing the net repulsion. This override mechanism guarantees that the user will come to a stop progressively without being too strongly repelled by the object. The parameters which control the relative strengths of the force fields, the oscillation damping, and the user input can all be tuned to achieve an appropriate 'feel' for a given type of environment.

Collision Avoidance

If we were to use only force fields it would still be possible for a user to collide with objects, or even to pass completely through them, by 'pushing' hard enough with the 3D mouse. This is because our algorithm works incrementally, re-evaluating the force fields each time a step is made. To prevent such violations we establish a *safe region* around the user at each step. As long as the user's next step is inside this safe region, we can guarantee that no collision will occur. Any movement which is to move the user further beyond the safe region will be limited.

Figure 2 shows how the safe region is computed. First consider the left-hand half of the figure, which shows a single object, **O1**, and the current position of the user. Note that for simplicity, the user has been represented as a single point, although in the implementation more complicated body representations are also supported. The circle surrounding the user represents the range of movement which could be made under the control of the user's input, assuming that there are

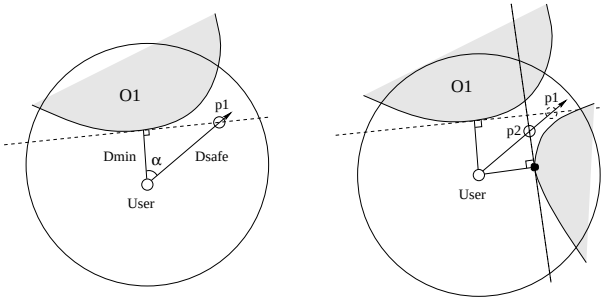


Figure 2: Safe region

no objects nearby. In this case, after taking account of the net forces acting upon the user, the intended direction of movement is towards the point **p1**. But we must check whether this would result in a collision with **O1**. Because objects are convex (or can be decomposed into assemblies of convex shapes), it is safe to move anywhere in the half-space defined by the tangent plane at the point on the object closest to the user, shown dotted in the figure. Thus, the user can safely move as far as **p1** in this example.

The right-hand half of Figure 2 shows how the addition of a second object, **O2**, further limits safe movement to the point **p2**, computed in the same manner. In fact, it is easy to see that the safe region is the intersection of the half-spaces defined by the shortest distances to each object, and that the safe step is the minimum of the distances computed in this way.

This is a conservative solution because there are cases where a user is able to move beyond the computed safe region without any collisions with objects. But this has proved to be quite acceptable in practice. The user is slowed down when close to objects and this feels very natural. It can be seen that the safe distance, **Dsafe** depends on the angle between the intended motion and the tangent plane, shown as α in the figure. Therefore, if the user moves parallel to a surface, such as a wall, no slowing down results. Once all nearby objects have been checked in this way, the user's position is updated, and the whole process is repeated.

Gravity and Staircases

A benefit of the force-field approach is that it is very simple to incorporate gravity. During normal operation, a downward gravitational force is applied to the virtual body. Horizontal (or inclined) surfaces, such as floors or stair treads, apply an upward force which counteracts this gravitational force. Unless the user provides an additional upward force, the virtual body's 'weight' and the upward force from the surface balance and the user can move smoothly across horizontal surfaces. Walking over a cliff or a hole in the floor removes the supporting force and the user falls vertically downwards. Thus, gravity can be included very inexpensively by treating it as an extra, constant force applied to the user's virtual body.

Vertical movement of the 3D mouse controls an additional upward force which can be used to overcome gravity. To facilitate smooth motion over flat surfaces, the vertical input from the mouse has a built-in 'dead space' so that small movements have no effect. Supplying a larger upward movement with the mouse overcomes gravity and permits users to 'jump in the air'. However, this takes them away from the supporting surface, and the repulsive force is therefore reduced as the distance increases. So, unless this jump causes them to land on another surface, the gravitational force will bring them down again.

Walking down a staircase is simple – the user moves forwards and the gravitational force causes them to drop from one step to the next. Thus, no special processing is necessary to deal with walking downstairs. However, to climb a staircase, the user must push forwards and upwards with the mouse. It is difficult to move the virtual body to the next step without colliding with the front of the next step. Therefore, in order to be able to clear a step, the upward movement is computed and applied *before* the forward motion. Then, provided that the user has arrived above a step, the application of gravity will cause them to drop back onto its horizontal surface. Before display, these vertical and horizontal movements are combined to form a single motion of the virtual body.

Searching for Nearby Objects

To avoid exhaustive testing of all objects in a model, we use hierarchical bounding volumes [15] for locating objects in the vicinity of the user. The search space is adjusted according to the user's speed of movement. The same data structures are used for control of culling and level-of-detail display. Therefore the algorithm is efficient, both for display and force computation.

Our implementation uses a system called MAVERIK [16], in which the behaviour of environments is customised using callback functions. MAVERIK already supports a number of different geometric primitives, including polygons, polygon meshes, cylinders, spheres, cones, tori, and boxes. Adding new primitives is straightforward, requiring only that appropriate methods be registered for certain operations (such as a display method). In order to compute the shortest distances and half-spaces for force evaluation and safe region testing, we register a callback function to perform these evaluations for each type of primitive.

EXPERIMENTS AND RESULTS

To test the new technique. we have conducted experiments with fifteen volunteer subjects, including both experienced users of VR and complete novices. We set them the task of following a prescribed route around a model of a house containing two floors, a dog-leg staircase, and some furniture. The route involved walking around and between tables and chairs, passing between the furniture and the walls, going upstairs, passing around more furniture, going back

downstairs, and returning to the starting position. The gaps between some pieces of furniture were deliberately made quite small, so that good navigation control was necessary to squeeze between them.

In the experiments we compared the force-field method with free navigation, which is totally unconstrained, and also with simple collision detection. In the former case users could walk through objects and the goal was to see how well they could follow a specified route, to measure the time taken and the number of collisions. In the case of simple collision detection, the user could move freely in uncluttered regions, but movement towards an object was prevented once the user came into contact with that object. The routes taken by a typical subject (number 7) in the three types of navigation are illustrated in Figures 4, 5, and 6, together with snapshots of the user's track on the stairs. The radius of the point plotted at each position is proportional to the time the subject stayed at the position, either for looking around, turning, or because of getting stuck there.

Each subject was tested with the three different methods – unconstrained (UC), collision detection (CD), and force-field (FF) – but with the order varied. Each experiment was repeated three times, totalling nine tests for each person, and 135 tests in the complete set. Each time, subjects were asked to complete the same route in the shortest possible time. The route and the time taken were recorded, and codes were instrumented to measure display frame rates and collisions in a post-processing phase. From these we were able to analyse other factors, such as the time spent without moving (looking around, or stuck in corners), and the processing requirements of the algorithms. We also interviewed each subject and asked them to rate their subjective views of each trial. During the experiments they were given no information about which method they were testing, but they were allowed a brief, preliminary practice session, using unconstrained motion, to familiarise themselves with the basic motion control using the forward, backward and twisting movements of the mouse.

The experiments were conducted using stereoscopic projection onto a screen of 2.5 metres diagonal dimension, viewed with passive, polarising glasses. Subjects sat in front of the screen and controlled motion with a 3D mouse containing a Polhemus sensor, as described previously. The software executed on a Silicon Graphics Crimson VGXT workstation with output through a videosplitter interface.

Here, we have space only to summarise some of the main results. A more detailed analysis is available from the authors [17], including correlation analysis, and a separate analysis of the performance of experienced and inexperienced users.

Timing Results

Table 1 shows the mean times taken by subjects to complete the route using each of the three methods. There is quite a large standard deviation, which is to be expected with a mixture of experts and novices. It can be seen that the force-field method is marginally quicker than totally unconstrained navigation and significantly better than the collision detection method. Further analysis allows us to draw other conclusions about the time taken for the different methods:

Method	Mean Time	Standard Deviation
UC	3'22"	1'19"
CD	4'56"	2'09"
FF	3'17"	1'35"

Table 1: Timings for different methods

- With the unconstrained method, users were able to move through the scene quite rapidly. Although they were told that they should attempt to avoid furniture and other obstacles, they did in fact make many collisions. One reason why the force-field method comes out best is that with unconstrained navigation users strayed outside the house or underneath the stairs (see Figure 4), which was very disorientating and therefore cost them time. As well as the simple timings, we have analysed the routes taken by users, and measured the times taken along the routes. Cases where users were disorientated show up very clearly in this analysis. The absolute path length with the UC method was 320 units, compared with 286 for CD and 292 for FF. The longer path length for the UC method is a result of users straying off-course. Although the path length for the CD method was shorter, users spent a significant amount of time without moving, either because they were stuck, or because they needed to look around to get their bearings.

- With the CD method, users frequently became stuck in corners (Figure 5). Corners exist where chairs protrude from underneath tables and where furniture is placed against the wall. With CD, the only way to escape such corners is to move backwards or sideways. With the FF approach, the force fields tended to guide them past such situations, resulting in a superior performance (Figure 6).

As an aside, it is interesting to note that because users have a more restricted field of view with the projection system than they would have in the real world, they tended to wrongly estimate the positions of objects which had passed out of their field of view. Thus, in turning to walk around a table, users would often turn *before* they had actually cleared the obstacle. With UC navigation, this resulted in a collision, and with CD it sometimes caused them to get stuck.

Collisions

The CD and FF methods guaranteed that users could not pass *through* objects. In analysing collisions in the unconstrained method, we distinguished penetration of external boundaries – passing through walls, for example – from colliding with furniture. The reason for this is that users were relatively unconcerned about the effects of passing through furniture. We attribute this to the fact that their eye-level was above the top of the furniture, so that such collisions did not affect their view of the scene. However, penetration of external boundaries usually results in a discontinuity in the view. In the results which follow, penetration of the floor is only included in the statistics if the viewpoint goes below the floor. Minor violations, where the lower part of the body penetrates the floor have not been included, as they do not affect the view and do not appear to seriously disorientate the user, although they can result in a 'child's-eye' view of the scene.

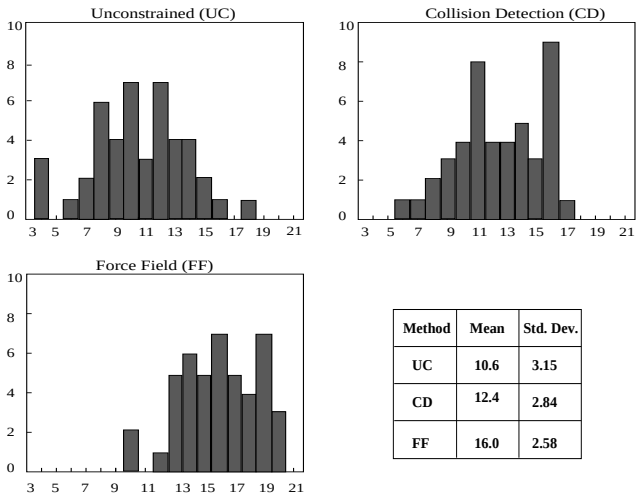
For the UC method, in only 3 tests out of the total of 45 were subjects able to complete the route without penetrating boundary objects. On average in each test the subject collided with furniture 7.3 times and passed through external boundaries (floors, walls and ceilings) 2.7 times. More than half of the penetrations of boundaries occurred on the staircase (1.6 times on average), indicating the difficulty of negotiating the stairs, while at the same time staying inside the walls on either side. Users ended up outside the house 1.7 times on average.

Subjective Analysis

As well as collecting data about timings and collisions, we asked our users for their subjective views of the methods. They were asked to rate four aspects of each method: general ease or difficulty of navigating on the level, ease of going upstairs and downstairs, and visual comfort. Under each heading, users were asked to score their experience on a scale of 1 to 5:

1	Very bad (very difficult to navigate, . . .)
2	Bad (quite difficult to navigate, . . .)
3	Average
4	Good (easy to navigate, . . .)
5	Very good (very easy to navigate, . . .)

Figure 3 shows a histogram of the aggregate scores for the four aspects, for each of the three different methods (maximum score of 20), while Table 2 summarises the separate scores (maximum score 5 for each). There was no significant difference in visual comfort, although the FF approach scored slightly above the other methods. We attribute this to the smoothing effect of the guided motion. But for the other three scores, which are all related to ease of control, the FF algorithm was rated significantly above the UC and CD methods. Force-field navigation obtained the highest mean overall response with the smallest standard deviation. Subjectively, it was clear that the greatest difficulty occurred



Method	Mean	Std. Dev.
UC	10.6	3.15
CD	12.4	2.84
FF	16.0	2.58

Figure 3: Histograms of overall responses to three methods

while negotiating the stairs. Going downstairs is particularly easy with the FF method – the user simply moves forwards and drops down under the force of gravity.

Method	*Mean Response*			
	Visual	*Navigating*	*Upstairs*	*Downstairs*
UC	3.15	2.47	2.10	2.05
CD	3.14	2.85	2.86	2.90
FF	3.70	4.29	4.19	4.28

Table 2: Mean scores for four aspects

Improvement with Practice

To see whether users managed to improve with practice, we compared their performances in each round of tests. As might be expected, the time required to complete the route improved in each case. Table 3 summarises the mean times. By the third series of tests there seems little to choose between the unconstrained and force-field methods.

	Time		
Method	*Round1*	*Round2*	*Round3*
UC	4'	3'11"	2'59"
CD	6'07"	4'34"	4'13"
FF	3'43"	3'08"	2'57"

Table 3: Average time taken in three rounds

However, as noted previously, with the unconstrained method, users made many collisions with furniture and penetrations of boundary objects. Table 4 shows how performance varied over the three rounds with the unconstrained method. Although there was some improvement in collisions with furniture, penetrations of boundaries actually became worse as users attempted to complete the task more quickly. With unconstrained movement, considerable concentration is required to avoid collisions, and users found this quite tiring.

Round	*Collisions with Furniture*	*Boundary Penetrations*	
		Total Number	*On Stairs*
Round1	8.13	5.53	3.33
Round2	7.27	5.07	3.13
Round3	6.47	5.80	3.40

Table 4: Collisions and penetrations in different rounds

CONCLUSION

We have proposed a new method for navigation in virtual environments which combines force fields and collision avoidance. Experiments in which we compared the new method with unconstrained navigation and simple collision detection show that the force-field algorithm out-performs these other methods, both objectively and subjectively. It guarantees a collision-free walk-through, while providing assistance in navigating around obstacles and negotiating stairs, and gravity comes free. The algorithm is simple to implement and efficient in execution.

ACKNOWLEDGEMENTS

The authors wish to thank their colleagues in the Advanced Interfaces Group for their support, enthusiasm and encouragement. Continued work on these methods is funded by the UK Engineering and Physical Sciences Research Council, grant number GR/K99701. We are also grateful to one of our industrial collaborators, Sharp Laboratories of Europe Ltd, who provided us with the stereoscopic projection system.

REFERENCES

[1] Mel Slater and Martin Usoh. Representations systems, perceptual position, and presence in immersive virtual environments. *Presence*, 2(3):221–233, 1993.

[2] K. M. Fairchild, B. H. Lee, J. Loo, and L. Serra. The heaven and earth virtual reality: Designing applications for novice users. In *Proc. IEEE Virtual Reality Annual International Symposium(VRAIS)*, pages 47–53, sept 1993.

[3] Frederick P. Brooks Jr. Walkthrough Project: Final Technical Report to National Science Foundation Computer and Information Science and Engineering. Technical Report TR92-026, Computer Science Department, University of North Carolina at Chapel Hill, 1992.

[4] H. Iwata and K. Matsuda. Haptic walkthrough simulator: Its design and application to studies on cognitive map. In *The 2nd International Conference on Artificial Reality and Tele-existance, ICAT 92*, pages 185–192, 1992.

[5] Mel Slater, Martin Usoh, and Anthony Steed. Taking steps: The influence of walking technique on presence in virtual reality. *ACM Trans. on Computer-human Interaction*, 2(3):201–219, September 1995.

[6] Steven M. Drucker and David Zelter. Intelligent camera control in a virtual environment. In Wayne A. Davis and Barry Joe, editors, *Proc. Graphics Interface '94*, pages 190–199. Canadian Human-Computer Communications Society, May 1994.

[7] E. Bouvier and P. Guilloteau. Crowd simulation in immersive space management. In M. Göbel and J. David and P. Slavik and J.J. van Wijk, editor, *Virtual Environments and Scientific Visualization '96*, pages 104–110. Springer-Verlag/Wien, 1996.

[8] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 25–34, July 1987.

[9] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings IEEE Robotics and Automation Conference*, pages 500–505, 1985.

[10] Ronald C. Arkin. Motor Scheme-based Mobile Robot Navigation. *The International Journal of Robotics Research*, pages 92–112, August 1989.

[11] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE trans. on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[12] B.H. Krogh. Integrated path planning and dynamic steering control from autonomous vehicles. *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 1664–1669, April 1986.

[13] Parris K. Egbert and H. Winkler. Collision-free object movement using vector fields. *IEEE Computer Graphics and Applications*, 16(4):18–24, July 1996.

[14] Jeffrey Jacobson and Michael Lewis. An experimental comparison of three methods for collision handling in virtual environments. In *Proc. Human Factors and Ergonomics Society Conference*, 1997.

[15] Timothy Kay and James Kajiya. Ray tracing complex scenes. *ACM Computer Graphics*, 20(4):269–278, 1986.

[16] Roger Hubbold, Dongbo Xiao, and Simon Gibson. MAVERIK – The Manchester Virtual Environment Interface Kernel. In M. Göbel and J. David and P. Slavik and J.J. van Wijk, editor, *Virtual Environments and Scientific Visualization '96*, pages 11–20. Springer-Verlag/Wien, 1996. ISBN 3-211-82886-9.

[17] Dongbo Xiao. *Interactive display and intelligent walkthrough in a virtual environment*. PhD thesis, Department of Computer Science, University of Manchester, October 1997.

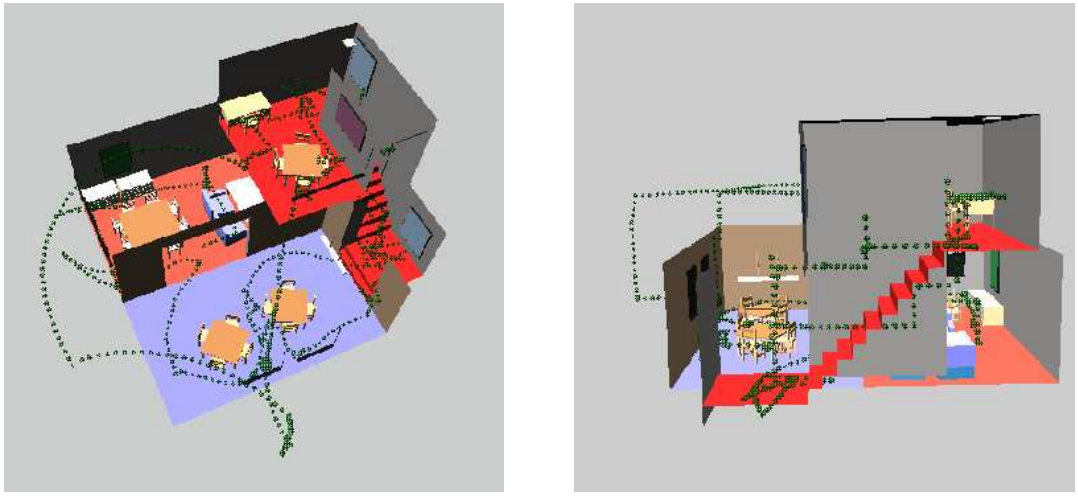


Figure 4: A typical route with the unconstrained approach

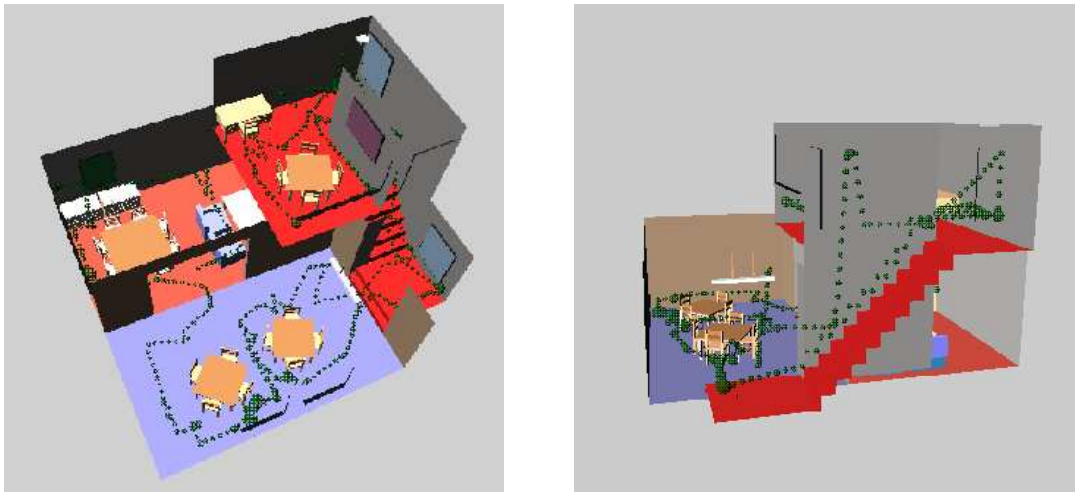


Figure 5: A typical route with the simple constrained approach

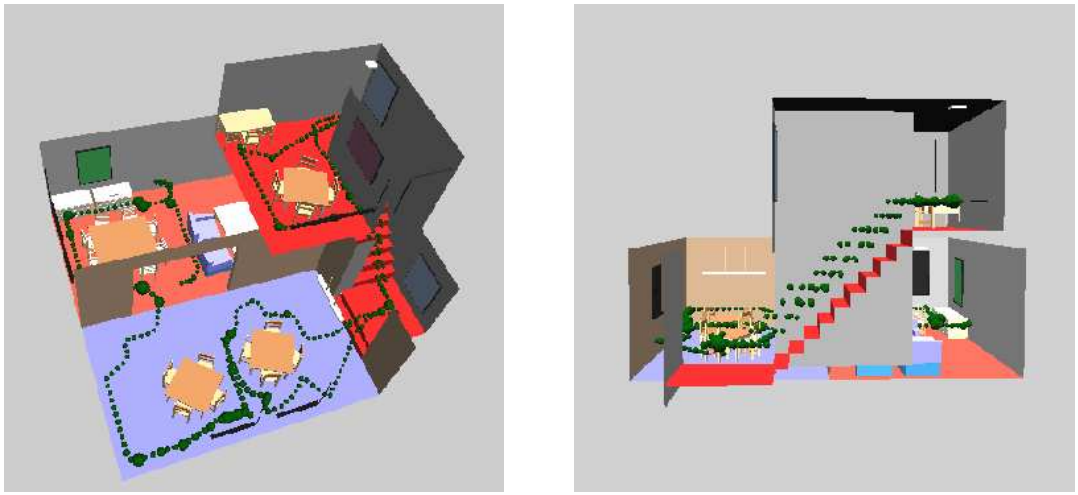


Figure 6: A typical route with the force-field approach