Computer Science
University of Manchester

A study of force fields for user navigation in virtual environments

D. Xiao & R.J. Hubbold

Technical Report UMCS-98-8-1

A study of force fields for user navigation in virtual environments *

D. Xiao & R.J. Hubbold

Advanced Interfaces Group
Department of Computer Science
University of Manchester
Oxford Road, Manchester, UK.

June 1998

Technical reports issued by the Department of Computer Science, Manchester University, are available by anonymous ftp from `ftp.cs.man.ac.uk` in the directory `pub/TR`. The files are stored as PostScript, in compressed form, with the report number as filename. Alternatively, reports are available by post from The Computer Library, Department of Computer Science, The University, Oxford Road, Manchester M13 9PL, UK.

Abstract

This report describes a force-field algorithm for assisting navigation in virtual environments (VEs). The fundamental idea of the approach is to surround objects in a VE by forces which gently repel a user's virtual body, providing a steering effect which helps the user to avoid collisions with obstacles. The algorithm also incorporates a 'safe region' detection scheme which provides a conservative, but guaranteed, method of collision avoidance. By adopting an approach based on force fields we also take account of gravitational effects, and methods for ascending and descending stairways. The report describes these algorithms and presents results of testing the approach with human subjects.

Keywords: Virtual environments, navigation, collision avoidance, force fields.

1 Introduction and background

The work reported here formed a part of the PhD research of one of the authors, Dongbo Xiao, supervised by Roger Hubbold [Xia97]. A technical paper, covering some of the ideas, was published at the ACM CHI'98 conference in Los Angeles, in April 1998 [XH98]. This stimulated sufficient interest that we felt it worth issuing this longer Technical Report, which contains more results and material than it was possible to include in the CHI paper.

Since the work was originally undertaken, the key ideas have been further developed and incorporated into our VR software system called MAVERIK [HXG96, CHK97, CHK98], and the approach has been tested with some very large-scale industrial models containing hundreds of thousands of primitive components.

1.1 Problems of navigation in virtual environments

Navigation through a computer-generated synthetic scene – termed a walk-through – is a fundamental aspect of exploring virtual environments. One of the features virtual reality should provide for a user during walk-through is the feeling of presence, which includes not only issues like real time performance and realistic images, but also natural navigation. Natural navigation methods contribute to a user's sense of presence, cited by some researchers as one of the three defining attributes of virtual reality [SU93]. The illusion of presence can be lost through unnatural experiences during navigation. These can be caused by unnatural interaction metaphors or by anything inconsistent with the user's everyday experience of the real world. Efforts have been made in developing new metaphors for walk-through in a virtual environment. Examples are Fairchild *et al*'s leaning metaphor [FLLS93], Brooks' steerable treadmill [Bro92], Imata and Matsuda's roller skates [IM92], and Slater and Usoh's walking-in-place metaphor [SUS95]. However, intuitive interaction metaphors are only able to solve part of the problem of walk-through.

Another part of the problem is related to the properties of a virtual environment. The physical world in which we live is full of solid objects. When we collide with solid objects, we do not penetrate them unless they are broken. In any serious application, a computer generated world will be more convincing if it possesses this property. Unconstrained motion is sometimes desirable, indeed liberating, since less concentration is needed in the control of motion. However, in situations where more realistic, real-world behaviour is required, more constrained navigation methods are called for.

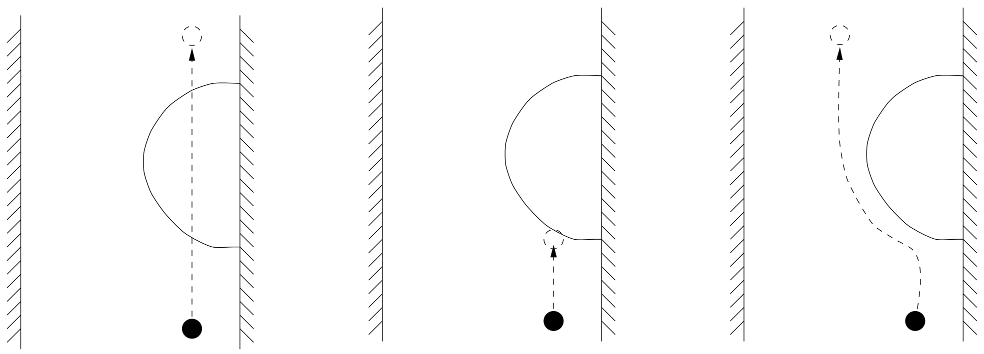
One of the problems of existing navigation systems, particularly in complex or cluttered environments, is that it is very difficult, for a user to control a virtual body which has no mass or other natural attributes. When attempting to find one's way past objects without any haptic feedback, one is relying totally on visual cues, and this is doubly difficult if the field of view is restricted – a common situation with many head-mounted displays. Examples of unnatural navigation, such as walking through objects like walls, abound in most existing systems. Particularly confusing for the user is the act of mistakenly walking backwards through a wall, so that the world suddenly disappears. The usual approach to solving this problem is to introduce some form of collision detection to prevent users from passing through solid objects.

Various techniques and algorithms have been proposed for collision detection in applications such as computer animation, physically-based modelling and robotics. However, as will be discussed later in this report, most of these are not quite appropriate for interactive, real-time walk-through. Furthermore, collision detection on its own is not a satisfactory solution to constrained environments. While it 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 or trapped by small objects which are not in one's immediate field of view. What is desired is a guiding mechanism which assists the user's movement to effectively avoid obstacles.

Difficulty in going upstairs or downstairs is another situation where an unnatural experience may occur and disrupt the sense of presence. One way to handle stairs is to add semantic knowledge to models to specify features of objects. Clearly this is not always practical since many models may not have been generated with such information and adding this new information can be very burdensome. Other than providing affordances in models, more general techniques are required so that not only can the steps of a staircase be stepped onto, but also other solid objects, provided that they are not too high to reach.

Integrating the effect of gravity is also important to enhance the naturalness of the user's experience during a walk-through. In most existing systems, the user is free-floating within the model instead of, say, walking on a floor. This may lead to an inappropriate view of the model for users to correctly locate themselves.

2 A new solution



(a) Unconstrained (b) Constrained (c) Guided

Figure 1: Three types of walk-through

In this report we propose a new walk-through algorithm which introduces artificial force fields into a virtual environment. By surrounding objects in an environment with repulsive force fields, the algorithm generates a walk-through environment where the user's navigation is guided. The algorithm also incorporates a number of natural features into a virtual environment, such as gravity and the constraints of solid objects. The approach is distinguished by its ability to assist the user to make effective movements in order to avoid obstacles during navigation. The differences between the new approach and the unconstrained approach and the simple constrained approach are illustrated in Figure 1 with a simple scene. Suppose the user is moving forward and approaching an obstacle in front of him. 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 simple collision avoidance method (Figure 1b). With some guidance, the user may expect to move slightly sideways, avoid the obstacle, and carry on going forward (Figure 1c).

As a preliminary study, our work has focused on the walk-through problem for a single user; however, the new approach could be used for applications with multiple users and moving objects. The discussion in this report focuses on convex primitives since concave primitives can be divided into convex components.

3 Related Work

3.1 Walk-through

Several groups have experimented with techniques for walking through virtual environments. Although most of them focused on investigating novel interactive metaphors which allow the user to explore a virtual environment in a more natural way, some of them addressed collision detection and complex paths including stairs and ladders.

During a six year project, researchers at the University of North Carolina at Chapel Hill (UNC-CH) [Bro92] have built a succession of six interactive graphics systems which enable a user to experience an architectural model by means of a VR walk-through. Emphasis has been put on methods of enhancing frame update rates, generating realistic images, and supporting a wide variety of devices, including the UNC ceiling tracker, for tracking a user's position and orientation. Their systems also provide more natural interfaces to users. For instance, one can reach out a hand and open doors, pick up and carry certain objects, and move furniture around. More recently, they presented a collision detection library for large environments, named **V-Collide** [CGH+97]. V-Collide is a successor of their I-COLLIDE software (an exact and interactive collision detection system), which will be discussed later. However no updated report on this work has been found which addresses the ability to effectively negotiate stairs.

Slater and Usoh proposed a new interactive metaphor [SUS95] to simulate body movements associated with walking which are tracked by a neural network. Their ingenious walking metaphor allows users to move around in the space defined by the electro-magnetic tracker. To cover a virtual distance which is larger than the physical space afforded by the tracker, users carry out "walking in place". Experimental evidence was given to show that, in comparison with the pointing technique based on a simple hand gesture, the walking-in-place technique tends to increase the subjective sense of presence, provided that there is an association with a virtual body. They also examined paths including "stairs and ladders", but their implementation relied on explicit knowledge of where these were located. In detecting a collision with the bottom step of a staircase, continued walking moves the user up the steps. In most existing models, this kind of information is not available and adding such affordances to them is often impractical. Furthermore, in the real world, not only the steps of a staircase can be stepped onto, but also other solid objects. Therefore, a more general technique is called for.

3.2 Collision avoidance

Collision detection has been studied as a fundamental problem in applications such as computer animation, physically-based modelling and robotics, where an object's motion is constrained by the presence of other objects and other physical constraints. The major objective of collision detection is to discover or prevent geometrical contacts between moving objects or between a moving object and static obstacles. This highlights a very important issue in collision detection: discovering or predicting the collisions between two objects, for which many algorithms have been proposed. Dobkin and Kirkpatrick [DK83] presented an algorithm which is able to detect collisions as well as compute separations between two convex polyhedra A and B in O($\log|A| * \log|B|$), where $|A|$ denotes the number of vertices in polyhedron A and similarly for B. This algorithm decomposed each polyhedron into drums, which are slabs cut by parallel planes. The simplicity of drums makes them similar to two dimensional polygons, so that an algorithm for polygons can be used to find intersection between two drums. A hierarchical structure of all drums provides efficient determination of pairs of drums which need to be tested.

Lin [Lin93] described an efficient algorithm for computing the closest distance between two convex polyhedra which is related to the collision detection problem. Based on the observation that the points on each polyhedron closest to the other polyhedron will only move a small amount when two convex polyhedra move only a small amount relative to each other, the algorithm incrementally keeps track of pairs of closest features between two convex polyhedra. Lin *et al* also extended this idea to handle non-convex objects [LMC93]. Each non-convex object is decomposed into convex components and is represented as a tree structure in a preprocessing computation. An interior node is a convex hull (or dynamic bounding box) of the union of its children, and leaf nodes are convex components. The algorithm recursively examines the trees for possible collisions between objects. If there is no collision between two convex hulls of two nodes on different trees, the two objects do not collide, so their closest-feature pair and the closest distance are updated. If two nodes collide, their children will be tested until either a real collision is found or no collision is reported at same level of the trees. Collision detection for curved objects represented as spline models or piecewise algebraic surfaces are also discussed in Lin's PhD thesis [Lin93].

Collision detection is usually accompanied by strategies to simulate the response of objects involved in collisions. Collision response is generally application dependent, and many algorithms have been suggested for different environments like motion control in animation [MW88], physical simulations by Baraff [Bar92, Bar90], Hahn [Hah88], Pentland and Williams [PW90], or molecular modelling by Turk [Tur89]. However, simulating the physical behaviour, or reaction, of objects due to collision is beyond the scope of the navigation method discussed here. What is important for a walk-through is not how the user will react on hitting a wall, but rather how the user can be guided not to hit the wall in the first place.

To avoid exhaustive testing between all pairs of objects, various subdivision techniques have been used in previous algorithms and systems. These include uniform gridcells [ZPOM93] which partitions 3D space into axis-aligned grids, octrees and variants [Hah88, MW88, Sha92] which act like a hierarchy of grids at different resolutions, and trees [TN87]. The role of these structures is to quickly narrow the range of necessary pairwise testing of collisions.

Collision detection techniques can be classified into two categories: static collision detection algorithms and dynamic collision avoidance algorithms (or path planning). In static schemes, the motions of objects are sampled into a series of static states at a set of time instances. At each time point, the intersections between objects, normally in pairs, are checked. If the result is positive, the positions of objects involved in these intersections are altered to simulate the collisions. The major weakness of static techniques is that collisions occurring between sample time instances can be missed, which is defined as "fixed-time weakness" by Hubbard [Hub95]. Static techniques are not adequate for a walk-through since the user can experience very confusing switches of views if he passes through a wall between time instances and suddenly faces a completely different scene from the previous moment.

To address this problem, some researchers use four dimensional analysis, where the simulation time is represented as the fourth dimension. One of the earliest examples of this idea is presented by Boyse [Boy79]. He derives simple equations for motion of polyhedral edges and faces over time. By assuming that only one polyhedron is moving and its motion is limited to periods with only linear translations and periods of pure rotations around a fixed axis, the algorithm can find the collisions of the moving object and one stationary object without any iterative time-step. However, the assumptions of the algorithm significantly limit its use in practice. Cameron [Cam85] describes several methods for "clash detection" for polyhedra. One of the methods uses a space-time approach, which generates 4D objects from the original 3D objects by adding time as another dimension. Intersections are then detected between 4D objects. If no intersections are detected, the motion along the period of time will be free of collisions. For simple shapes with linear motion, this algorithm performs well. However, more complicated objects and complex motion are difficult to handle. Another method uses sweeping volumes to represent moving objects over a period of time. If objects collide with each other, their sweeping volumes must intersect. This algorithm is very intuitive but rather restrictive. One problem is the difficulty of explicitly sweeping out the envelopes of moving objects unless the motions of objects are completely known in advance. Another problem is the lack of temporal information in the intersection test; the intersection of sweeping volumes of two moving objects does not necessarily indicate a real 'clash' between two objects.

A generate-and-test paradigm has been used in the path planning problem [ELP86]. The paths of moving objects are generated one at a time based on a pre-defined priority. Once a path is generated, the method tests for intersections between it and the existing paths. If an intersection is detected, the object's path is altered and intersections are re-tested with the modified path. If no intersection is discovered, the system moves on and performs the process on the next object.

In Lozano-Péreg's configuration space approach [LP83], the moving object is reduced in size to a point while static objects are enlarged in size proportional to the amount the moving object was reduced. In this way, the complexity of the computation for collisions is reduced because the moving object is always a point. A collision free path is obtained by connecting the start position, a group of the vertices of the grown obstacles, and the goal position with a set of straight line segments, in which none of the line segments intersects with any of the grown obstacles. If the moving object changes its orientation, all configuration space obstacles (the enlarged obstacles) need to be re-computed. This technique is difficult to extend to 3D since the enlargement of the obstacles can be very difficult and time consuming, especially when the objects are of complex shape.

In many respects, the problem addressed in this report, namely collision-free walk-through, has more similarity with path planning in the robotics field, which is classified as dynamic collision avoidance. Firstly, in path planning, normally there is only one object, a robot, which is moving and the rest of the environment is stationary. Secondly, none of the potential collisions will be missed during the real route the robot takes. However, there is a fundamental difference between robot path planning and collision-free walk-through for a human user. That is the user's intention during the walk-through is arbitrary, can change rapidly, and will dominate the whole course of the walk-through, while a robot will passively respond to the environment or follow a given path. In other words, the user's motion is more unpredictable in an interactive system than a robot's motion. This difference makes most path planning techniques inadequate for the walk-through scenario since they all rely on some knowledge of the object's motion, such as its velocity and its goal, which are predefined and will be maintained. None of these techniques can handle motion that is not pre-defined.

To close the gap between path planning and interactive control in real time, Quinlan *et al* [Qui94] proposed a new framework where paths generated by a path planner are deformed in real time according to local sensors. With the concept of elastic bands (deforming collision-free paths), the behaviour of the robot is not completely determined at the planning phase, but also responds to the changes in the environment. With a global control which moves the robot along the elastic band, local behaviour does not limit the ability to achieve global goals. An elastic band is the path connecting a finite series of bubbles, which represents a subset of the free space around the robot. When the robot is in a clear area, the bubble expands, and when the robot is getting near to obstacles the bubble shrinks and leaves a smaller free space for the robot to move.

To address the collision detection problem in an interactive environment, Hubbard proposed a time-critical algorithm which consists of two phases: the broad phase and the narrow phase [Hub95, Hub96]. In the broad phase, the algorithm very rapidly finds the possible collisions by using space-time bounds, four dimensional structures similar to those used in other 4D analysis discussed above. A space-time bound gives a conservative estimate of where an object's bounding sphere might be for the immediate future. By finding intersections between space-time bounds, the algorithm finds possible collisions. The definition of space-time bounds is based on the object's velocity and acceleration, so that it does not require the exact position of the object over time. Thus, the algorithm can handle objects whose motion is determined interactively by human users. The broad phase of the algorithm returns a 'maybe/no' answer and the application has the chance to ask the algorithm to refine the accuracy of the 'maybe' answer which is accomplished in the narrow phase. To discover a real collision, the narrow phase uses a tree structure to represent each object at different levels of detail approximating the object's surface. The simplest 3D geometrical shapes – spheres – are used as the elements of approximations of objects. Each level of the tree structure is a union of spheres which approximates the object and the approximations become more accurate at the deeper levels of the tree. Therefore, the collision detection problem between objects is simplified since checking between spheres is computationally inexpensive. Furthermore, the hierarchical structure allows the application to terminate the process of collision detection at any time with an approximate answer, in order to maintain a target frame rate and latency. Empirical studies showed that, in some situations, the approach can improve the performance of an application by more than two orders of magnitude. However, the construction of the sphere-trees involves a considerable amount of computation and thus can only be built as a preprocessing step. As a result, the algorithm can only handle rigid objects and is not applicable to flexible objects. Furthermore, the approximation of objects by spheres is not suitable for all types of models. For very complex models, the storing of multiple levels of sphere representations can be memory intensive.

Recently, researchers at the University of North Carolina at Chapel Hill presented an exact and interactive collision detection system, **I-COLLIDE**, for large-scale environments [CLMP95]. Their system uses a two-level approach based on pruning multiple-object pairs using bounding boxes and performing exact collision detection between those pairs of polyhedral objects which are selected in the first level of test. Their collision detection library was integrated into a walk-through application, where the scene is composed of polytopes, and the user's hand, consisting of several convex polytopes, moves through this environment, modifying other polytopes in the environment. Their experiments showed no degradation of frame rates when collision detection was added.

However all these approaches only addressed part of the problem in walk-through. The approach presented in the thesis aimed at a further goal – a guiding mechanism in a walk-through.

3.3 Force field approaches

Our idea of using a force field around objects to help guide the user was motivated by the work of Bouvier and Guilloteau [BG96], who used a flocking algorithm [Rey87] for simulation of crowd control in a virtual environment. However, in their paper, they used only normal collision detection mechanisms for preventing collisions with the environment.

Force field approaches have been used previously for path planning in the robotics field. A technique proposed by Khatib [Kha85] used artificial force fields to prevent collisions. In this system, objects (robots or parts of a robot) are given a destination position – the goal they were moving towards once started. Obstacles in the environment are surrounded by repulsive forces, which increase as a moving object approaches the obstacle's surface. At the same time, the goal attracts the moving object with a force proportional to the distance between the object and the goal. When a moving object approaches an obstacle, a force is exerted on it which is the combination of the goal's attractive force and the obstacle's repulsive force. The resultant force prevents the moving object from colliding with obstacles while it moves towards its goal.

Krogh [Kro86] extended Khatib's work by considering not only the distance but also the velocity of the moving object in calculating the potential field. Arkin's motor schema based navigation [Ark89] also used potential force field as the output of a schema to control the movement of a mobile robot. The common basis of all these techniques for robotics is that the robot's target was known in advance, so that a force field for the whole environment can be pre-calculated. With *interactive* navigation, the algorithm must be able to compute movement in response 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.

A similar idea is used in Borenstein and Koren's system [BK89] to achieve real-time obstacle avoidance for a mobile robot. Rather than use an analytical function to control the strength of an obstacle's repulsive force, they used the feedback from sensory instruments mounted on the robot to determine the increase of the force field.

Egbert and Winkler [EW96] proposed a similar technique, which uses repulsive vector fields around objects to prevent collisions among multiple objects during simulation. The system works with a uniform object type – triangle meshes. For each object (a set of triangles), a surrounding volume is defined to specify the bounds of its repulsive vector field, providing a user-specified area of "dead space" between the exterior of the object and the edges of the surrounding volume. This surrounding volume is subdivided into an octree structure with a threshold on the number of triangles in a node. The vector force field of the object is generated by assigning a repulsive vector to each node of the octree with a strength linearly proportional to the depth of the tree. For nodes actually enclosing edges of the object's surface, the vector is increased by an order of magnitude to simulate forces at infinity. The directions of repulsive vectors depend on the characteristics of the object nearest the point of approach of the potentially colliding objects. Even with a given direction and strength of objects' motion, their algorithm cannot guarantee a complete avoidance of collisions. Again, subjects in the vector field are passively reacting or have a fixed intention of motion, and no interactivity has been taken into account.

4 Walk-through based on artificial force fields

4.1 Overview of the new approach

First, it is necessary to establish a user's conceptual model for the navigation method. It is also important that solutions for any issues should maintain the consistency between the user's input and the visual update of the environment in order to increase the sense of presence.

Some of the factors which contribute to the user's conceptual model in our approach are:

- Conceptually, the visual data fed back to the user by the environment should match his intention to the greatest degree. That is, when the user intends to move forward, the view of the environment should be updated correspondingly.
- Responses of the environment to the user's interactions should be as natural and comfortable as possible.
- Changes in orientation should be completely under the user's control, and the force applied by objects will only affect the user's position – that is, translation not rotation.

- When the user is far away from any objects, his movement should be completely and precisely under his own control.
- The user should not be allowed to get too close to certain objects, such as walls, unless he insists on doing so. If the user insists somehow, he should be able to get very close to objects or even touch them. The user should not be allowed, normally, to walk through any objects, no matter how strongly he insists.

In the walk-through system, the whole environment is treated as a force field in which each object is surrounded by a subfield of force which is normally directed outwards to repel the user getting too close to it. The closer a position is to the surface of the object, the stronger the force will be. At any time, objects close enough to the user will apply a force on him. The user's motion is also specified as a force, controlled by the user with some suitable input device (e.g. a 3D mouse). The input device controls the magnitude and direction of the force. At each step, the user's next position is decided by a combination of his input force and surrounding force fields. Collisions are avoided by defining a safe region (described in the next section), which indicates a safe region of motion from the current position. Gravity is incorporated into the method as a force which is always directed downwards. The user will be kept in contact with the floor by gravity until he applies an upward force to counteract gravity. Once the upward force applied by the user is released or reduced, he will fall towards the floor underneath him under the force of gravity. In a sense, the system takes the force of gravity for granted since gravity is just treated as an extra force exerted on the user. To avoid an exhaustive testing of all objects in a model, the algorithm uses hierarchical bounding boxes for locating objects close to the user. 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.

The novel contributions of the work described in this report are:

- Introducing force fields into interactive environments.
- First introduction of a guided walk-through by combining force fields and collision avoidance.
- Simple and effective collision avoidance.
- Incorporating gravity into virtual environments using a force field.
- Comparisons between three different kinds of walk-through.

4.2 Computing the force fields

The basic idea behind this approach is very straightforward: that is, introducing the notion of force into a walk-through and providing a collision-free environment. To successfully achieve this goal, a number design issues have to be solved.

- The user's control with the input device needs to be converted into forces in order to be incorporated into the algorithm.

- The formulation of force fields around objects needs to be designed in a manner which effectively repels the user but at the same time, ensures a comfortable movement. Unlike a simulation of physical phenomena, the force field in this approach is just a tool to guide the user's navigation in a virtual environment. Therefore, the most important thing is the acceptability from the user's point of view. The individual forces need to be resolved to generate realistic and comfortable motion.
- Interactivity is an important goal for a walk-through system. Therefore, it is impractical to calculate the resultant of the entire force field for the whole environment. Instead, computing the force field applied by objects in the immediate vicinity of the user will be sufficient. In addition, extracting those objects close to the user should be quick.
- As discovered by others [EW96], a force field on its own is not able to guarantee collision avoidance. A collision testing strategy has to be proposed to complement the force fields, which needs to be simple and effective to meet the real-time requirement of the navigation problem.
- When testing the approach, it was found that oscillations and unpredictable movement can occur if the user is completely under the influence of the force field. Obviously, protection against this is required.

In this section, solutions to all these issues will be described in detail.

4.2.1 The user's input force

The user expresses his movement intentions by operating a device attached to his body, or more typically, using a hand-held 3D mouse. In the current implementation of the system, the user is provided with a 3D mouse, through which he can indicate desired motions. To integrate the user's intentions into the force field approach, the input data has to be specified as a force moving the user a certain distance in a designated direction. A unit input force corresponds to a movement of a predefined step size D. Given the current position of the user $p_1(x_1, y_1, z_1)$, and the expected next position $p_2(x_2, y_2, z_2)$ indicated by the user, the force will act in the direction of the vector $p_1 p_2$ with a magnitude of $|p_1 p_2| * D$.

In the implementation described here, the user's input is supplied with a 3D mouse containing a Polhemus Fastrak sensor, but it is simple to adapt the control mechanism to use other input devices, including a 2D mouse. 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. After a little practice users become quite adept at controlling their virtual bodies. All mouse motions are measured relative to the mouse's position when the button is first pressed. 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.

4.2.2 The 'vulnerable' space surrounding a user

At any time, the danger of collisions will only exist between the user and objects near to him. Therefore it is not necessary to take into account all objects in a model when computing the force field exerted on the user. Considering the force fields of those objects which are in close proximity to the user will be sufficient. In other words, given a surrounding space, every object inside that space will have a role to play in determining the update of the user's position, while objects outside will be ignored. The volume of the user's surrounding space can be crucial to the effectiveness of the method. If the algorithm defines a large surrounding space, many objects would be detected inside, and therefore contribute their forces to repel the user. This will not only slow the computation but may also lead to unnecessary constraints on the user's motion, if more objects are taken into account than necessary. If the surrounding space is too small, a large movement of the user will lead him to move outside the region where collisions are detected and prevented. Generally, the maximum distance the user can move at each step should be constrained inside the surrounding space and the size of the space can be decided by evaluating the density of the scene and the expected pace of the user's movement. The determination of the size of the surrounding space can also dynamically take into account the velocity and acceleration of the user's movement. In this case, a larger area will be detected if the user moves quickly.

Once the surrounding space is determined, all objects which appear inside the current surrounding space of the user can be identified with the aid of one of the spatial management structures.

4.2.3 Formulation of an object's force field

The force field which artificially exists around each object is inversely proportional to the distance from the surface of the object, and therefore tends to infinity as the user approaches the object's surface. In order to avoid undesirable perturbing forces beyond the object's immediate vicinity, the influence of the force field must be limited to a certain region surrounding the object. Only when the user moves close to an object, will the force from the object begin to repel him. Beyond this pre-defined distance (termed *the region of influence*), the user's movement should be free from the influence of the object.

Objects are constructed from convex primitives, such as cylinders, spheres, cones, boxes and polyhedra. Non-convex shapes can be constructed by assembling convex primitives. If the user is close enough to an object to be inside its *region of influence*, the object creates a repulsive 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. The force should be able to become strong and effective enough, if necessary, to stop the user colliding with any objects. At the same time, it should leave as much freedom as possible to allow the user to navigate freely. These two requirements conflict. An acceptable solution must give priority to collision avoidance.

Initially, we planned to use an inverse square law to control the force, but it was discovered that the sudden increase in repulsion as the user approached objects made this method difficult to control interactively. After experimentation, the approach finally opted for was 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. In the region very close to the surface of objects, special processing is conducted to remove undesired oscillation (for detail see Section 4.2.5). 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.

When the user is inside the influence range of an object force field, the force field around the object will repel him. The force field exerted by the object can be efficiently represented by a single force applied at a specific point on the object's surface and acting on a point on the user's body. To achieve this, the algorithm identifies two points: P_a and P_r. The acting point of the object force, P_a, is a point on the object's surface, called *the actor* of the force, and the receiving point of the force, P_r, is a point on the surface of the user's representation, called *the receiver* of the force. Since force fields are normally formulated correspondingly with the distance between a position and the surface of the object, the actor-receiver pair (P_a, P_r) is one of the pairs of points between the object and the user where the shortest distance exists. In other words, the shortest distance between the object and the user exists between P_a and P_r. The object force between P_a and P_r has a magnitude inversely proportional to $|P_a P_r|$ and is directed from P_a to P_r.

4.2.4 Resolution of object forces

In mechanics, the composite force on the user would be obtained by summing the individual component forces exerted by each object. However, it was 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. After a number of experiments, three rules were derived which govern the force computation:

1. When considering forces from surrounding objects, the algorithm does not sum the individual forces to obtain a resultant force. Instead, only the maximum of the component forces contributed by each object, resolved into each of the positive and negative x, y and z directions, may be applied. This is done to ensure that placing two objects side by side does not generate a larger force field than either of them would alone. A small, solid obstacle generates an identical repulsive force to a large one, its magnitude depending only on the shortest path from the virtual body to the surface of the object. More particularly, two coincident walls do not generate twice the force of a single one. This is important to maintain a smooth movement when the user is repelled away from objects.

2. If the user does not apply any force to the virtual body, with the mouse, then he remains stationary. This is important as it means users are always completely in control of movement. They won't get pushed around if they happen to stand next to objects which apply a force, unless they also apply a force. Without such a rule a user would drift around at the mercy of any local force fields.

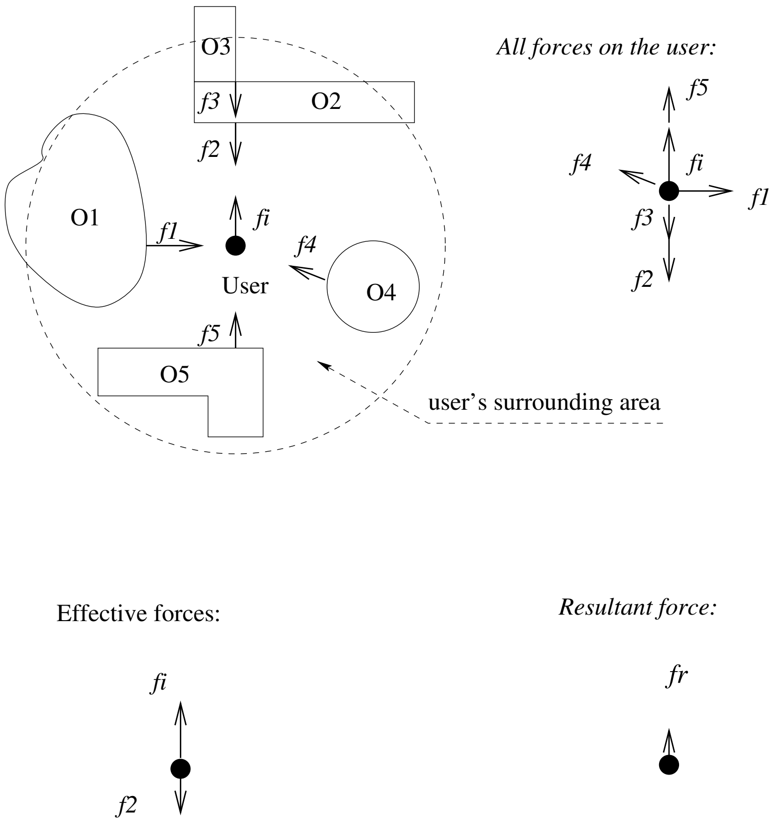


Figure 2: Accumulation of object forces – example 1

3. Objects only push *against* the direction in which the user tries to move. Thus, if a user attempts to move forwards through an object, the movement will be resisted. The same will happen if he tries to go backwards through something. But, if he stands with his back to a wall and then walks forwards, the wall will not exert a force. If the wall were to exert such a force, it would be like receiving a shove in the back. It can be quite disconcerting if environments behave in such an apparently aggressive manner!

Let us examine an example shown in Figure 2. Five objects, labelled O_1, O_2, O_3, O_4, and O_5 are detected to be inside the surrounding space of the user, and each of them exerts a force f_j (j=1..5) on the user. Note that the vertical forces are ignored here for simplicity. There are two objects (O_2 and O_3) adjacent to each other and both of them are close enough to affect the user's movement. The stronger force between f_2 and f_3 will be taken into account when the composite force is computed according to rule 1. The user is moving forwards, the effect of forces f_1 and f_5 are canceled according to rule 3. Similarly, force f_4 is also ineffective. Consequently, there are only two effective forces applied on the user horizontally in the example: f_2, and f_i, where f_i is the user-applied force.

Figure 3 shows another example where the user is surrounded by the same set of objects, but is moving in a different direction. In this case, f_4 is effective since the user is moving

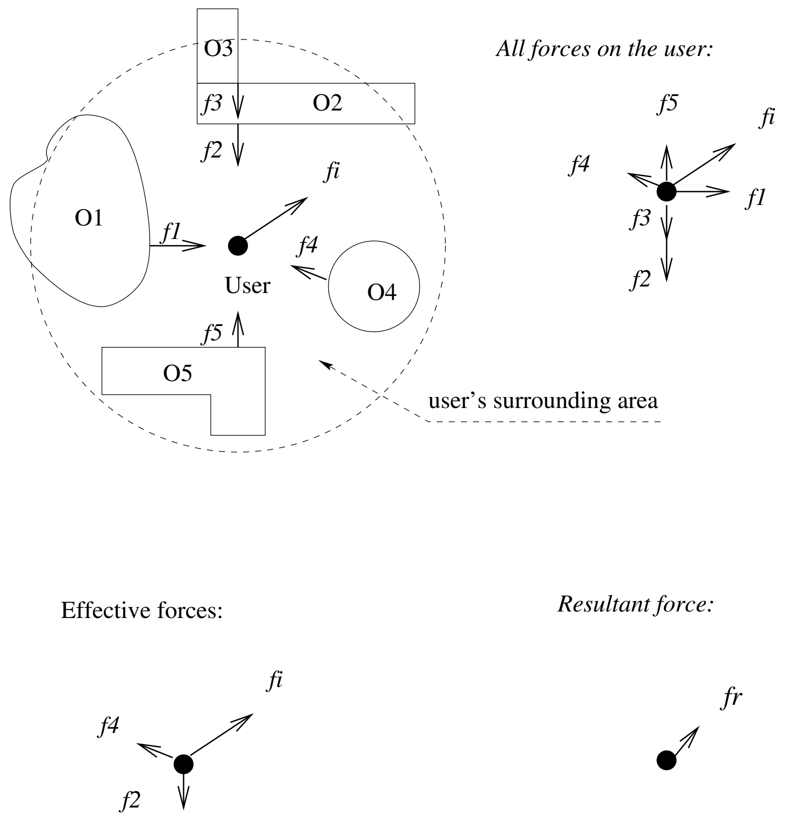


Figure 3: Accumulation of object forces – example 2

towards object O_4.

Consider the x component of the resultant force. The positive component and negative component of the repellent force exerted by objects are f_{rx+} and f_{rx-}, respectively. The user's force and an object's force have x components f_{inputx} and f_{jx}, respectively. The pseudo-code for computing the x component of the composite force of all effective objects O_j is:

f_{rx_-} =0; f_{rx_+} =0;
FOR each O_j in S_o
IF $\left(f_{inputx} > 0 \; AND \; f_{jx} < 0\right)$
  THEN f_{rx_-} = MIN(f_{jx}, f_{rx_-});
  ELSE IF $\left(f_{inputx} < 0 \; AND \; f_{jx} > 0\right)$
      THEN f_{rx_+} = MAX(f_{jx}, f_{rx_+});
END
$f_{rx} = f_{rx_-} + f_{rx_+}$;

4.2.5 Resultant force and oscillation control

Once the algorithm obtains the user-applied input force F(f_{ix},f_{iy},f_{iz}) and the composite repellent force F(f_{rx},f_{ry},f_{rz}), the resultant force F(f_x,f_y,f_z), can be computed correspondingly as follows by vector addition:

$$
\begin{aligned}
f_x &= f_{ix} + f_{rx} \\
f_y &= f_{iy} + f_{ry} \\
f_z &= f_{iz} + f_{rz}
\end{aligned}
$$

However, it was found during experiments that a resultant force computed directly with the combination of the input force and the repellent force may still lead to undesired movements. This usually happens when the repellent force is greater than the user's input force. For example, if the user moves forwards towards an object and the object's repellent force is stronger than the user's force, the resultant force may push the user backwards. Under certain circumstances, this may become so serious that if the user is very close to an object, the user's actual position oscillates. 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 permanent 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. No matter how carefully the input force and object forces are calculated, it is still impossible to avoid oscillations due to the interaction of the user.

This could be solved by setting the corresponding component of the final resultant force to zero once the component of the repellent force is detected to be greater than that of the input force. However, this will also diminish the guiding role of the object forces because it is the overriding component of the objects' forces which cause the user to move away from an object or objects. This is illustrated in Figure 4. The user is approaching an object with a force f_i, the object repels the user with a force f_r, and the x component of the object force f_r overrides the x component of the user's input force f_i. In the left picture, the overriding part is always set to zero. Therefore the y component of f_i will be the only final effective force which will causes the user to move straight forward. In the right picture, the overriding part is, in effect, causing the user to move sideways to avoid the object. In the latter case, if the user really wants to get closer to the object, he can insist on doing so by pushing harder towards the object.

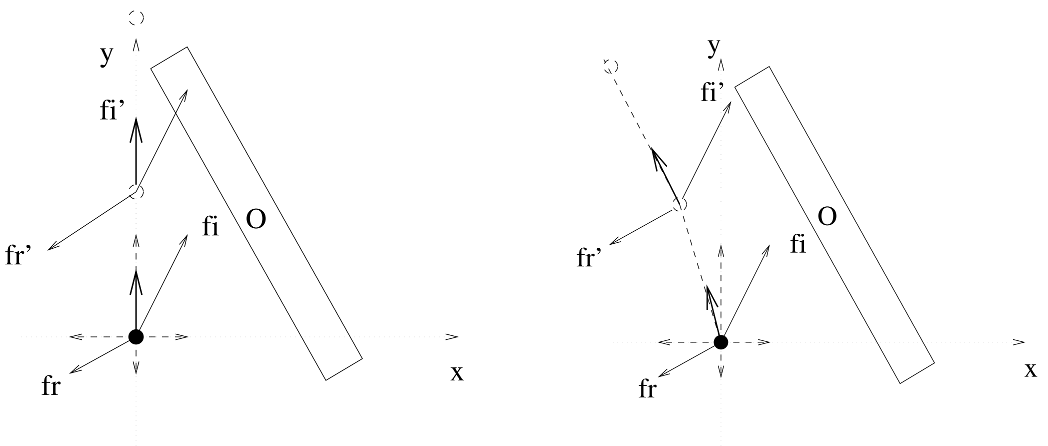


Figure 4: Guiding effect of object force

An ideal solution is to counteract oscillations without losing the 'steering' effect. By experimentation, it was discovered that oscillations only occur when the user is close to an object and that this is only disturbing when it occurs in the direction in which the user intends to move. To remove this kind of oscillation, a critical area is defined with a critical distance from an object's surface, which is adjustable according to requirements of applications. The critical distance indicates the distance beyond which the user's intention of moving closer is treated as an insistence on touching the object. If the user is further from any object than the critical distance, the resultant force is computed directly by resolving the input force and the repellent force. Only when the user is closer to an object than this critical distance should action be taken to avoid oscillations. Then, the user's movement is decomposed into components along three axes.

The upward direction is treated differently, and is discussed in the next section. The axis where the greatest component of the input force exists is identified as the *major direction* of the user's movement. The other axis is the minor direction. If the repellent force is greater than the input force in the *major direction* of the user's movement, the resultant force is set to zero so that oscillations can be avoided. In the minor direction, the resultant component is set to a predefined value, called *the deviating factor* if the object is in the critical distance from the closest object. If the user is closer to an object than the critical distance, the resultant component is scaled to a value which is inversely proportional to the nearest distance between the user and all the surrounding objects: the closer the user is to the object, the less the steering effect is.

The pseudo-code for computing a component (e.g. along the x axis) of the resultant force is:

```
IF (f_ix > 0)
   THEN IF (f_ix + f_rx > 0)
            THEN f_x = f_ix + f_rx;
            ELSE IF x is the major direction
                     THEN f_x =0;
                     ELSE IF (d_min > d_critical)
                              THEN f_x = f_ix + f_rx;
                              ELSE f_x = f_deviator*(d_min/d_critical);
   ELSE IF (f_ix < 0)
            THEN IF (f_ix + f_rx < 0)
                     THEN f_x = f_ix + f_rx;
                     ELSE IF x is the major direction
                              THEN f_x =0
                              ELSE IF  (d_min > d_critical)
                                        THEN f_x = f_ix + f_rx;
                                        ELSE f_x = f_deviator*(d_min/d_critical);
            ELSE f_x = 0;
```

This overriding mechanism guarantees that the user will come to a stop progressively without being too strongly repelled by the object. This makes it possible for a user to approach an object such as a wall directly, without any significant oscillation. With these modifications in place, navigation was found to be quite comfortable and easy to control. 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.

4.3 Collision avoidance

Although the user's motion is constrained by object forces, it is still possible to pass through objects if the user applies a strong force, unless further protective strategies are brought into operation. This is one of the major differences of the algorithm discussed here and other methods, such as [EW96].

Traditional techniques for inter-penetration detection can be used here to protect the user from collisions with, or inter-penetrations of, objects in the scene (an example of this solution is described by Witkin [WB95]). For a polygonal object, the inter-penetration can be detected by checking the plane equations of the polygons comprising the object. For each polygon, the valid side of the plane, which has a normal pointing away from the object, is called the positive side. The other side is called the negative side. All the planes whose normals point towards the user's current position need to be tested. If the expected next position is still on the positive sides of all these polygon planes, a safe position is confirmed, otherwise, a collision is detected. When a collision is detected, one solution would be to leave the user at the current position whenever a collision or inter-penetration is detected. This is inadequate because most of the time there are still safe positions between the current position and this invalid next position. Another would be to interpolate movement from the current position to the expected position in small steps, testing the positions of subsequent steps until a position which is safe, and follows the user's intention to the maximum possible limit, is found.

This method will work well for polygon-based models; however, it may be expensive to test for collisions with complex primitives, especially when primitives are at arbitrary orientations.

4.3.1 Safe region

Instead of employing techniques used in traditional collision detection algorithms, we employ a *safe region* around the user at each step, which reflects the free space between the user and surrounding objects. Between each object and the user, once the closest pair of points is obtained, the whole 3D space is divided into two halves by a plane which involves the actor of the object's force field and has a normal pointing pointing to the user. The half space in the direction of the normal is called the positive half-space. If the user's motion stays in the positive half-space, there will be no danger of collision with the object, therefore the positive half is a safe half-space for the user to move in terms of the presence of this object, provided the object is convex. All safe half-spaces associated with objects inside the user's surrounding space, for a given position of the user, define a safe region for the user's movement. Thus, limited by this safe region, a safe motion for the user can be obtained.

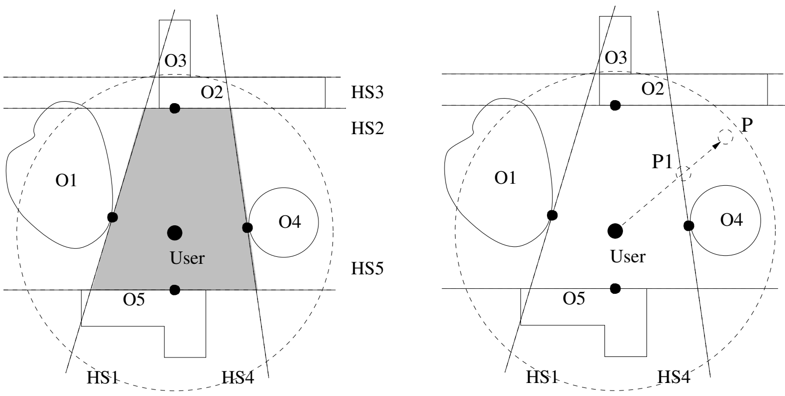


Figure 5: Safe region and Pace control

This is a conservative solution because there are cases where the user would be able to move beyond the safe region without colliding with any objects. An example is illustrated in the right-hand side of Figure 5, where the expected next destination of the user is P, which is outside the safe region, but is actually safe. In this case, the user only moves to P_1. At the new position P_1, the force and safe region will be recomputed and the process continues. As mentioned before, the pace of the user's motion is controlled by a predefined step size. If the safe distance exceeds the predefined step size, his motion will still be limited by the predefined step size. In addition, it is consistent with common sense that when we get closer to objects, we should be more cautious. In our everyday experience, the more crowded a scene is, the slower we will walk or drive. In other words, the pace of the user's movement is related to the density of the scene. In practice, this has proven to be quite acceptable.

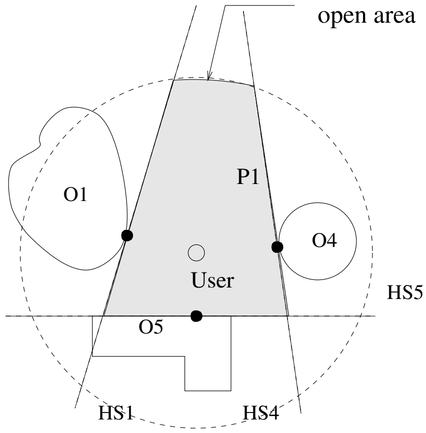


Figure 6: Safe region with open area

If the combined safe region formed by the safe half-spaces is not closed, the relevant portion of the boundary of the surrounding space of the user is used to close the region, see Figure 6.

4.3.2 Fast computation of the safe region

It would be expensive to precisely compute the safe region of the user at each step, especially when there is an open boundary to the safe region. Given the direction of the resultant force, the equivalent problem is to find the maximum distance the user can safely move from the current position up to the boundary of the safe region.

If there is only a single object effective and the user is walking from the safe half-space towards the other side as shown in Figure 7, the safe distance the user can move in the direction of the resultant force is $D_{safe} = D_{min}/cos(\alpha)$, where d_{min} is the length of the shortest path between the object and the user's representation, and α is the angle between the opposite direction of the object force and the direction of the resultant force.

If the angle α is greater than 90 degrees and the user is walking inside the safe half-space, the safe distance is infinity in theory. However, in practice the maximum distance will be the radius of the surrounding space of the user, or more normally the predefined step size.

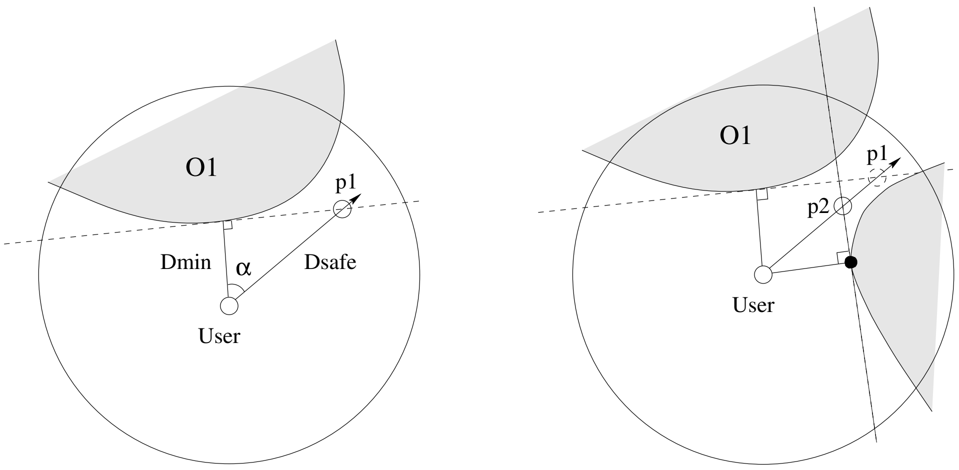


Figure 7: The definite safe distance to move

If there is more than one object inside the surrounding space, the minimum value among all the safe distances would be the genuine safe distance to move. This is shown in Figure 7, where the user will only move to position p_2 as limited by object O_2, instead of moving further to p_1 as limited by only object O_1.

Therefore, given the set of effective objects O_e each having an actor-receiver pair (a,r), and the direction of the resultant force F(x,y,z), the distance the user will be able to safely move from the current position will be:

$$D = \min_{o \in O_e}\left(\frac{d_{(a,r)_o}}{cos(\alpha_o)}\right)$$

Where $d_{(a,r)_o}$ represents the distance between actor a and receiver r, which is also the shortest distance between the object O and the user, and α_o represents the angle between the vector *ra* and the resultant force F.

The user's actual motion will be completely determined by the resultant force, provided the next potential position is inside the safe region, or the distance to move is less than the safe distance. If the distance to move is greater than the safe distance, the motion will be limited to the safe distance.

This method works well for convex objects. Concave objects can be divided into convex components, which in turn can be treated as individual objects.

4.4 Summary

The procedure for determining the next position of the user, given the current position and the input force comprises the following steps:

- Converting the user's input to an input force.

- Identifying all objects inside the user's surrounding space; this is accelerated by using a spatial management structure, such as a hierarchy of bounding boxes (normally, it is the same structure used for culling).
- For each effective object, computing the actor and the receiver of the force and then the force itself.
- Accumulating all effective forces.
- Combining the user's input force with the composite repellent force, taking oscillation control into effect.
- Deciding the next safe position in the direction of the resultant force by using safe regions.

5 The user's body representation

In the previous discussion, the user's body is simplistically represented by a single point. More meaningful representations are necessary, which account for approximate the dimensions of the user's body. Any increase in geometric complexity of the user's representation leads to some increase in the complexity of the computation for forces and safe regions.

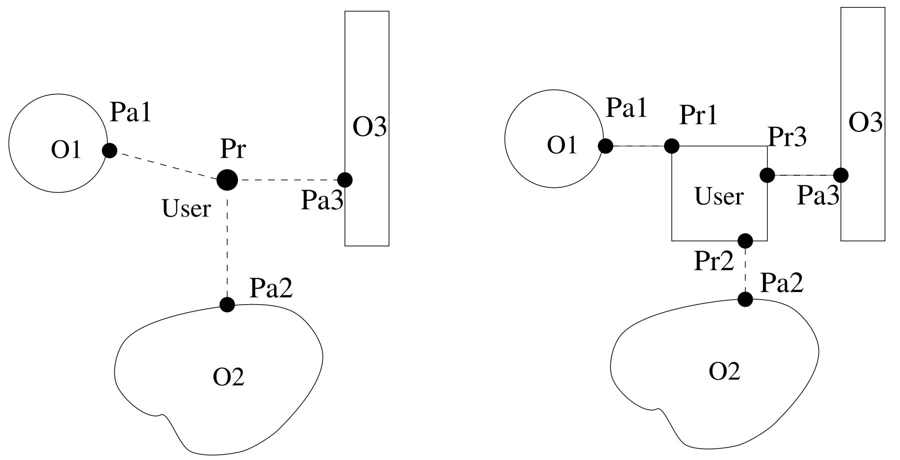
5.1 Multiple receivers

As discussed previously, in order to obtain the force exerted by an object, it is necessary to identify two points for each object: the *actor* and the *receiver*. In the single-point representation of the user, the receiver for all the objects' forces is identical. However the receiver, P_r, of the user can vary for different objects in a more complex representation. Figure 8 shows an example where for the single point representation of the user, P_r is the same for all objects, while for a box representation, there are three receiving points: P_{r1} for P_{a1} of object O_1, P_{r2} for P_{a2} of object O_2, and P_{r3} for P_{a3} of object O_3.

In our scheme, orientation is completely under the control of the user, and the force field will not cause any rotations of the user's body. Once the object forces are computed using corresponding actors and receivers, they can be accumulated in the same way as in the single point representation approach, as though they were all exerted on the user's central point. This implies a rigid body representation for the user.

5.2 Safe region

Having the user's representation with dimensions implies that the receiver is not unique any more. Therefore, the safe region of the user's movement will be a ring-like shape as shown in Figure 9. However, the computation of the safe distance will be the same once different receivers have been used to compute safe distances for different objects. This is because for each object, the safe distance associated with it will be the distance from the receiver to the



(a) point representation of user (b) box representation of user

Figure 8: Acting point(s) and receiving point(s)

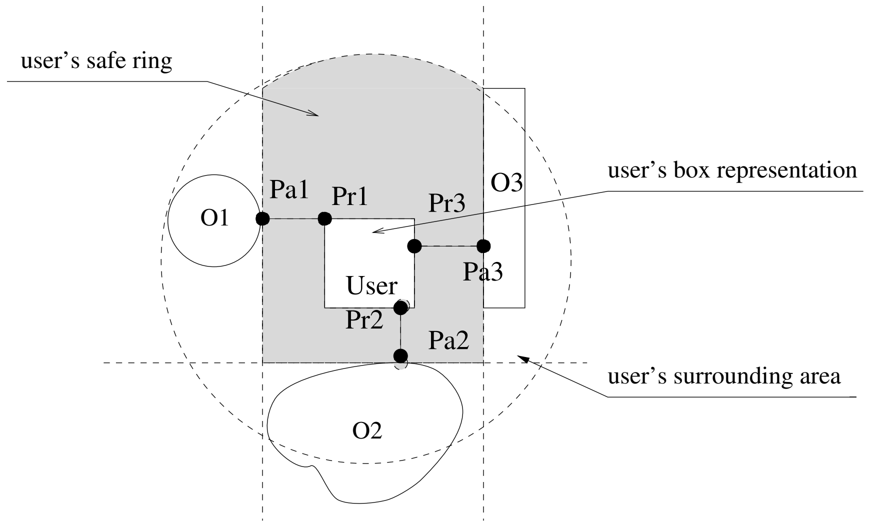


Figure 9: Safe ring of the user

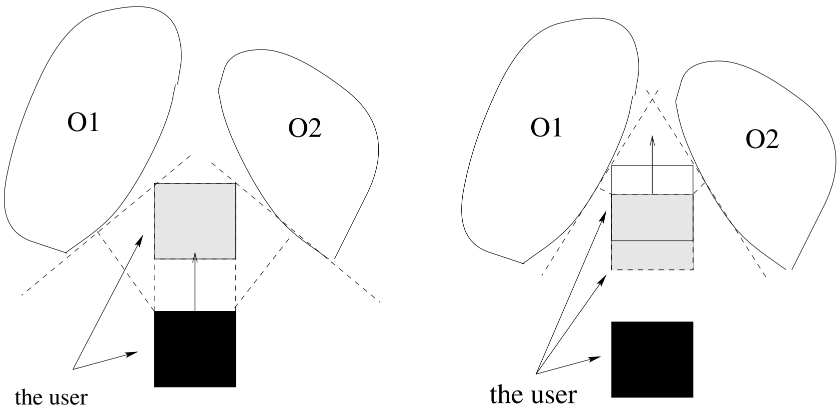


Figure 10: the user is approaching a small gap

safe region boundary which has already taken the user's body dimensions into account. The effectiveness of the safe region can also be demonstrated in protecting the user from going through a gap not large enough for his body. In the example in Figure 10, where the user is approaching a small gap, the safe regions will stop him at a position where there is not enough space ahead for him to pass through, no matter how strongly the user insists on going further.

5.3 More complex body representations

Of course, the user's body representation may be more complicated, including a head, arms, and legs, for example. In such cases, the body representation can be modelled by a hierarchy of convex components. Figure 11 shows an example of a hierarchy of components corresponding to a more complicated body representation.

If movable components of the user's body representation remain stationary relative to the main trunk during a walk-through, computations for force fields and safe regions are straightforward. For each object, the closest point between each component and the object can be computed in a similar way to the case of the single-component representation. Among these close points, the one closest to the specific object will be the receiver of the user's body corresponding to the force applied by this object. The rest of the computation is the same as for the simple representation.

The computation of safe regions can be more complicated if there are components moving independently when the whole body is also moving. In such cases, the motion of the whole body, called global motion, could be separated from the sub-motion of the movable components, called local motion. The global motion can be decided by only considering the effect of the input on the entire body and surrounding object forces, assuming that all components are stationary relative to it. Then, for each of the moving components, a similar computation could be performed just taking the input associated with this component into consideration. The final motion of moving parts will be a combination of the motion of the entire body and their local motion relative to the main part of the body. In this way, our approach can be extended to work

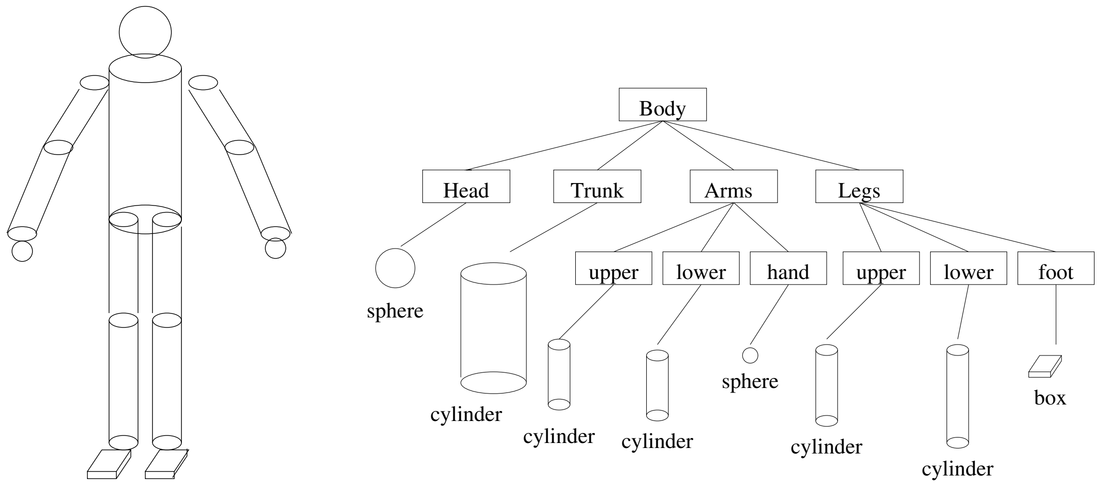


Figure 11: More complex representation of the user's body

for combined motions of the user's body components.

6 Floors and holes – gravity

In our everyday experience, because of the presence of gravity, we always walk on the surface of the earth, on a floor or on any surface wide and strong enough to sustain our weight.

In most systems, vertical movement is ignored; the eye-point of the user is at a fixed distance above the floor and stairs are handled by special codes. The real impact of gravity on a user's experience in virtual environments is still unknown – there is little experimental evidence. With the technique proposed here, it is straightforward to incorporate gravity; gravity is just an extra force exerted on the user's body.

The major function of the force representing gravity is to keep the user on a floor. Here we define a 'floor' to be a flat surface of any object with sufficient horizontal area to sustain the user. For example, we might regard an area as sufficient to support the user if its area covers more than half of the vertically projected profile of the lower part of the user's body. This can be customised, for different applications; for example, we might choose to treat any surface not inclined at more than 45 degrees as horizontal. If the user is on a floor, gravity is just strong enough to counteract the upward force exerted by the object forming the floor. If the user steps over a hole, which is defined as an open area large enough to let the user's body through, he will drop down under the effect of gravity until he reaches another floor. To jump up from a floor, the user has to apply an upward force to counteract gravity.

The force representing gravity always points downwards. Its magnitude is just enough to move the user downwards a distance of a predefined step size. In this study, the same step size is used as that of horizontal movement, but they could be different as long as both of them are not too big to guarantee that the next position is always inside the user's surrounding space, so that the user will not drop into an undetected area. The surrounding space of the user is adjusted to take account of speed. At the same time, vertical movement is also limited by the safe region to avoid collisions.

To make the new approach work well with gravity enabled, upward forces applied by floor surfaces need to be treated differently. This is because such surfaces should not repel the user upwards. Therefore, the upward force from any object is designed to counteract gravity only when the user is on the surface of the object, otherwise, no upward force is applied at all.

7 Stairs

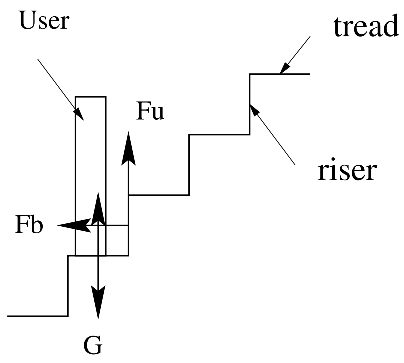
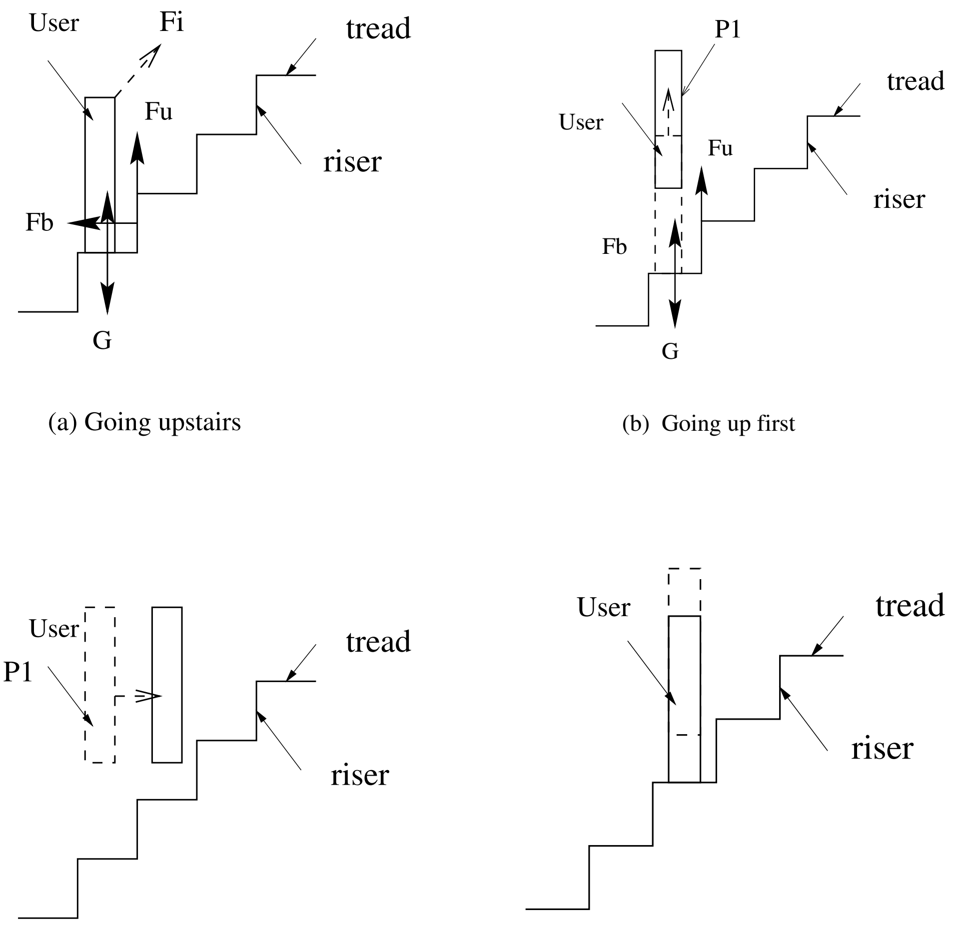


Figure 12: Stairs constructed by polygons

Another interesting problem in VE research is that of negotiating stairs [SUS95]. A staircase usually consists of a set of surfaces, which are either polygons themselves, or surfaces of other kinds of solid objects, such as a surface of a box. Figure 12 shows an example of stairs constructed by polygons and the forces exerted on the user when he is on one of the treads.

When approaching a flight of stairs, the user can express his intention to go upstairs by applying a diagonal force which goes forwards and upwards simultaneously. With such a simple approach, it is almost certain that the user will never move directly to the next tread since a diagonal movement will intersect the front edges of stair treads, violating the collision detection test. To solve this problem, the upstairs action is decomposed into two steps, going straight up first and then going forward. At the next position above the next tread, the user will drop down to the surface of the tread under gravity. At the starting position, forces around the user will be as shown in Figure 13(a). Since the user's motion is upwards, the repellent force from the riser will be ineffective (Figure 13(b)). As a result, the user will go up to the position he desires, provided that he does not encounter any objects above. The P_1 position is only a virtual position for the user to take, gravity should not be effective there. Next, the user will take a forward step with a distance indicated by the forward component of the input force and limited by the safe region at that position (Figure 13(c)). If the forward force is insufficient to move the user forward above the next tread, or if the upward force is insufficient to lift the user



(a) Going upstairs (b) Going up first

(c) Going forwards (d) Dropping to the step

Figure 13: Going upstairs

up above the next tread, the user will drop back to the current tread, instead of progressing to the next tread. This is because forward motion is typically limited by the riser on the next step. Once the forward movement succeeds, the user will drop back to the tread of the current step (Figure 13(d)), if he is floating above the tread.

In order to go downstairs, if gravity is enabled, the user only needs to push forwards as the downward movement will be accomplished by gravity. Pushing forward and downward simultaneously will also lead the user down to the next tread, where the motion is interpolated into two separate steps – forward and then downwards movements.

During the course of going upstairs and downstairs, at each step it has to be decided which stair tread the user will stand on. For this, it is necessary to define the notion of “the user is on a tread or a floor”. In the real world, a person will stay on, or stand on, a floor only if his centre of gravity is inside the region of the floor surface. In a virtual world, if the user's body is represented with a geometry of a certain dimensions, the definition of "the user is on a floor" could be the same as in the real life. The problem with this definition is that the centre of gravity changes when the user makes different gestures, and simulating the falling of the user when his centre of gravity is outside a floor is beyond the scope of our approach. Hereafter, the user is considered 'on a floor' if any part of the representation of his foot is inside the surface region of the floor.

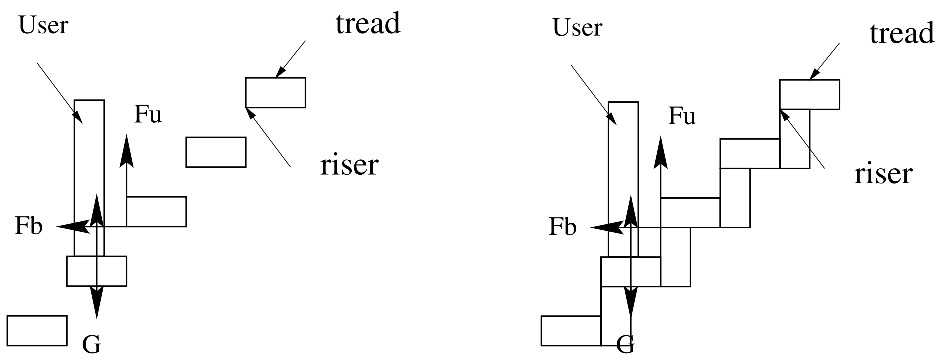


Figure 14: Other construction patterns of stairs

For other possible constructions of stairs, such as those shown in Figure 14, the approach will also work since the pattern of forces exerted on the user is the same.

This approach differs from methods used by other researchers [SUS95], in which the location of stairs is necessarily given in the model and will be detected as a sign for going upstairs or downstairs, the algorithm described here does not rely on any explicit information about stairs. Once the user applies an upward-forward force, he will be moved into a safe position along the direction of the resultant force computed from the input force and object forces. If there is a flat surface wide enough to sustain him and his input force is strong enough to go up and forward to reach the flat surface, the user will be able to jump onto the surface, which could be a tread of stairs or a piece of furniture. Therefore, the algorithm provides a general solution for stairs, and can be used for any existing models without the requirement to modify the models.

8 Experiments and results

Our experiments have focused on the effectiveness of the new technique in comparison with a free walk-through, 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 can move freely in uncluttered regions, but is prevented from passing through objects. As a result, movement towards an object is stopped once the user comes into contact with that object. With such a method it is possible to get stuck in corners; getting out of a corner entails moving backwards (or sideways) away from the constraining objects. The implementation of the simple constrained approach uses the same idea of collision avoidance as that used in the force field approach, namely, safe regions.

A house with two floors, consisting of 669 objects, and approximately 2723 polygons in total, was used as the experimental model for the study. Two typical views of the model are given in Figure 15. The house is furnished with objects like tables and chairs, which have non-axis-aligned shapes, and awkward corners. Some of the gaps between pieces of furniture are so small that good navigation control is required in order to successfully and smoothly pass between them without any collisions in a free walk-through, or without getting stuck in the case of the simple collision avoidance walk-through. There are also two flights of stairs leading to the top floor, where the effectiveness of the different walk-through methods for negotiating stairs can be examined.

Experiments were conducted on a Silicon Graphics Crimson/VGXT workstation. A 3D mouse, containing a Polhemus Fastrak sensor and a number of buttons, tracked the position and orientation of the user's hand, in order to control movement and changes in orientation during a walk-through. The position and orientation of the Polhemus trackers are nominally accurate to within 0.03 inches and 0.15 degrees up to a range of 30 inches, and the maximum range of the trackers is up to 120 inches. Wearing a pair of passive polarising glasses, users viewed stereo images of the model on a Sharp large-screen stereo projector.

Fifteen volunteer subjects were involved in the experiments, with 8 experienced users of virtual environments and 7 inexperienced. Subjects were given the task of accomplishing a collision free walk-through in the house, by following a specified route. During the walk-through, users had to 'walk' around furniture on the lower floor, go upstairs, 'walk' around a table and finally return downstairs. This involved passing between chairs and tables, and moving through the spaces between the furniture and walls. Each subject was tested with the three different types of walk-through (unconstrained, collision detection only, and force-field), but with the order varied, and each experiment was repeated three times. Subjects were asked to complete the same route in each trial in the shortest possible time.

Statistical analysis of the results of the experiments was performed with the SPSS package on Windows NT.

8.1 Direct observations

During the experiment, each subject was observed and qualitative aspects of their performances were noted. As might be expected, free navigation was found to be the worst case; subjects found it very difficult not to collide with furniture. None of them finished the route without any collisions, although for most of them walking through furniture did not seem to be too disruptive because it did not impact on their view of the scene. However, walking through walls, penetrating ceilings and floors proved to be an annoying experience, with two thirds of the subjects passing through these boundary objects, and in some cases, getting lost outside the house. Six of them became seriously lost, and the worst performer got lost five times. In two individual trials, the subjects were so badly lost that they were unable to find their way

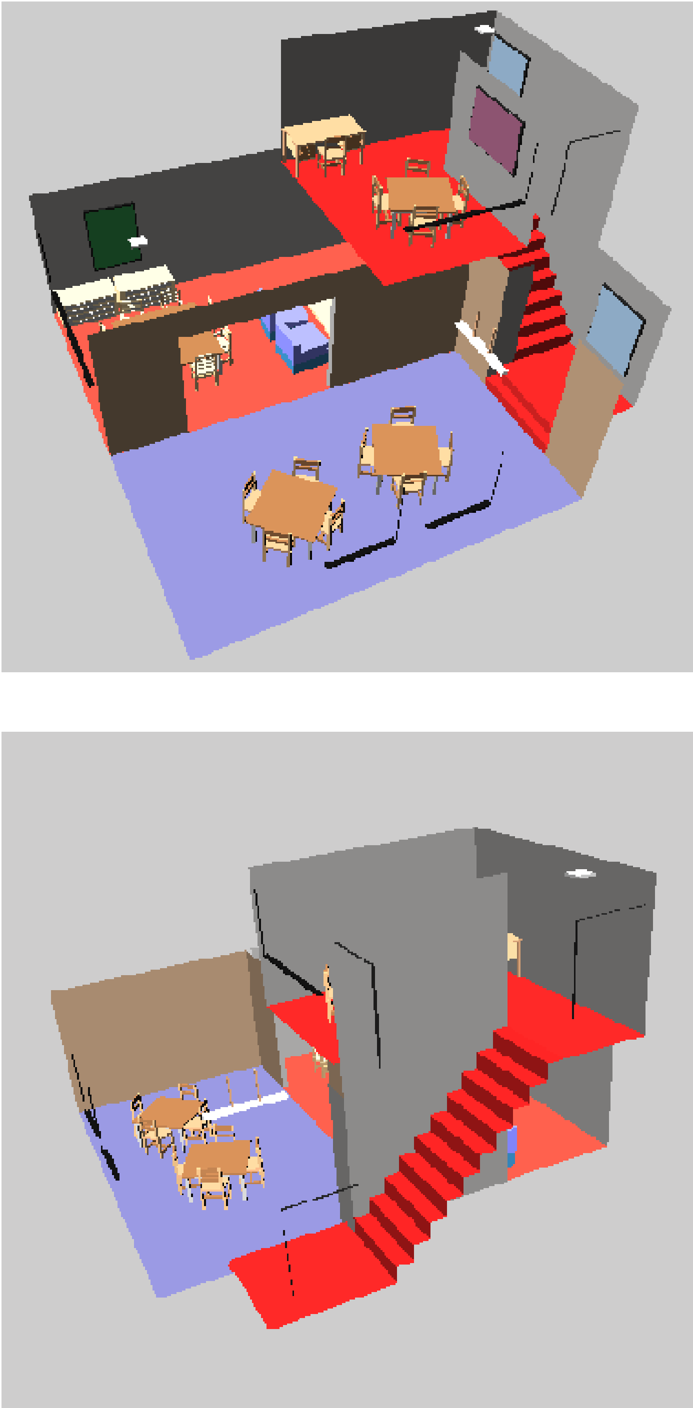


Figure 15: Typical views of the experimental model

back, and they failed to complete the route as required. The staircase proved to be difficult to negotiate with free navigation, especially going downstairs, where careless movement caused users to end up underneath the stairs.

With collision detection enabled, all of the subjects expressed relief that they were protected from walking through walls and did not need to worry about getting lost. However, all of them encountered the main problem with this approach, namely, getting stuck at corners of furniture. They had to make sideways or backward movements to negotiate such situations. This can be difficult if the field of view is constrained. Getting stuck is unnatural, but was found to be marginally less annoying than getting lost outside the house.

During their first attempt, fourteen out of fifteen subjects claimed the force-field guided approach to be the best, and the other person confirmed this during her second attempt. Twelve out of fifteen were aware of the guiding effects of the repelling forces when they were getting close to objects. The other three were aware that they could navigate more easily than with collision detection, although they were not conscious of the reason. All of them found navigation with gravity enabled much easier, since this keeps them on the floor and results in an appropriate view of the scene. Walking on the staircase was significantly easier with gravity enabled, where an appropriate view of the scene is much more important for subjects to properly locate themselves. Three out of fifteen subjects complained about visual jittering when going upstairs, which is due to the presence of gravity, however, the other subjects seemed to like this phenomenon and claimed that this increased the realism of going upstairs.

8.2 Objective evaluations

During the course of each walk-through conducted by subjects, the actual route taken and the time spent for completing the task were recorded. The number of collisions each subject made in the free walk-through was also gathered in a post-processing phase.

The routes taken by a typical subject (number 7) in the three types of walk-through are illustrated in Figures 16, 17, and 18, 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.

8.2.1 Timing results

Method	Mean Time (sec.)	Standard Deviation
Force-Field	3'17"	1'35"
Constrained	4'56"	2'09"
Unconstrained	3'22"	1'19"

Table 1: Time spent with different methods

Table 1 shows the mean times taken by subjects to complete the task using each of the three methods. There is quite a large standard deviation, which is to be expected with a mixture

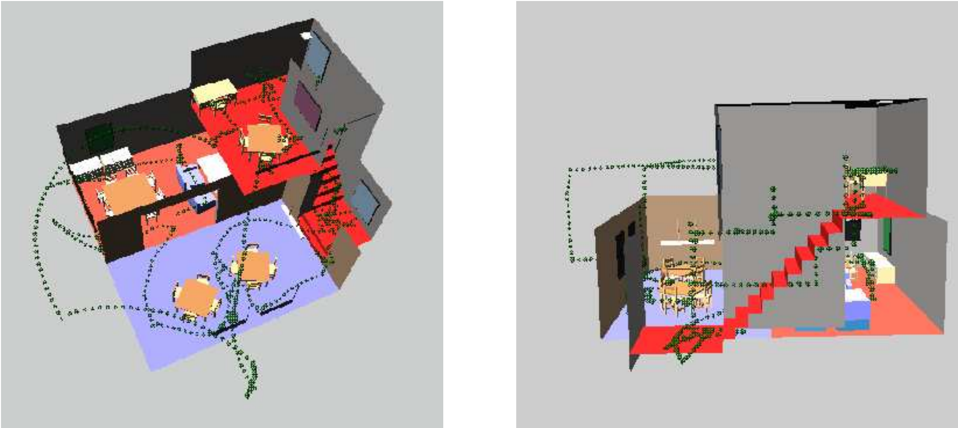


Figure 16: A typical route with the unconstrained approach

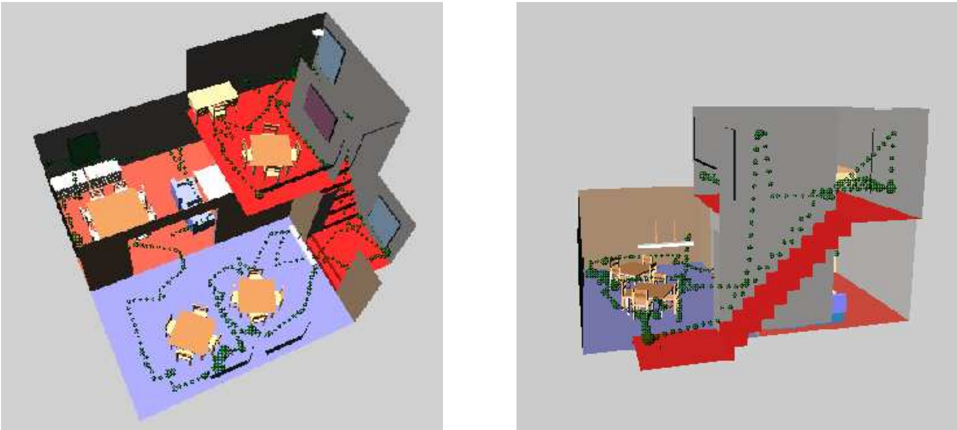


Figure 17: A typical route with the simple constrained approach

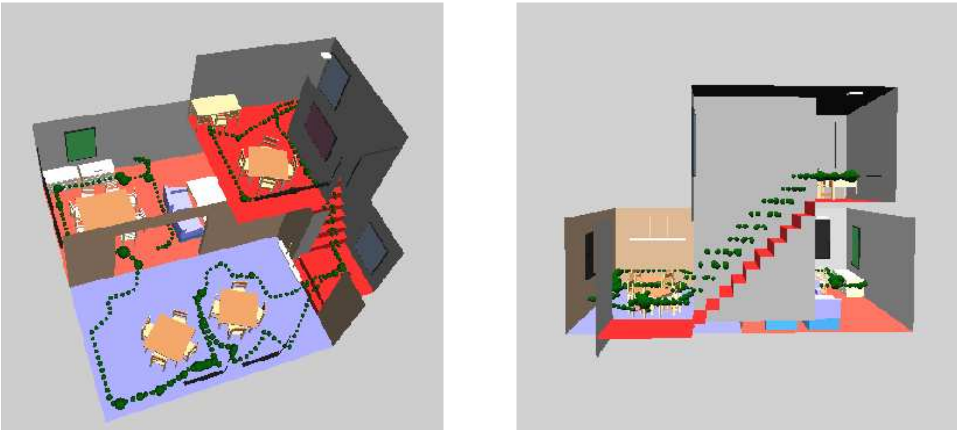


Figure 18: A typical route with the force-field approach

of experts and novices. The simple constrained approach takes the longest period of time, which matches our on-the-spot observation that subjects tended to move sluggishly due to the hindrance of obstacles, especially at the corners of furniture. Free navigation resulted in similar times to the force field approach, which was initially surprising, because the force field was expected to give a much better performance. However, users were not aware which method they were testing, nor were they told in advance how any of the methods worked. Therefore, their strategy was to move very slowly and carefully. We would expect the force-field technique to yield much quicker times if users knew that they did not need to be cautious.

In addition, the computations necessary for collision detection and force field generation slowed down the performance of the constrained walk-through and force field approaches. In a post-experiment analysis, it was found that the computation for generating one frame took an average of 15 ms for unconstrained walk-through, 32.7 ms for constrained walk-through, and 34.5 ms for the force-field walk-through. For the unconstrained case, this includes identifying objects inside the frustum using hierarchical bounding boxes, and displaying all the visible objects. The extra time spent in the constrained and force force field approaches is for identifying surrounding objects in order to generate the safe regions. The additional time was spent in the force field approach for actually resolving the forces around objects. The phase of collision avoidance seems to have taken a longer period of time than expected, and the reason for this is that the current implementation has not made use of the outcome of the culling processing; instead, the collision avoidance searched the spatial structure again to select all objects which may affect the user's movement. The computation for collision avoidance could be improved by using a more efficient algorithm, such as combining culling and collision avoidance into a single search of the spatial structure.

Method	**Maximum Stay (sec.)**		**Mean Stay (sec.)**	
	Mean	**Standard Deviation**	**Mean**	**Standard Deviation**
Force-Field	1.82	1.35	0.098	0.043
Constrained	3.88	3.15	0.138	0.55
Unconstrained	1.38	0.62	0.067	0.021

Table 2: Lengths of time subjects stayed at individual positions

The lengths of time subjects stayed in each individual position are summarised in Table 2. The "maximum stay" is the maximum time a user stayed at each individual position along the whole route, which is indicative of the worst-case difficulty of navigating in an environment. The "mean stay" of a trial indicates the overall difficulty of navigation for a complete walk-through. It is evidential that getting lost significantly degraded the user's performance. Even though the unconstrained approach took less than half of the computation time taken by force field approach, and gave a shorter maximum stay and average stay, it resulted in a similar overall time for a user to complete the task.

Although requiring slightly less computation than the force field approach, the simple constrained approach resulted in a much longer maximum stay (the longest was 16 sec.) and a longer average stay. The cause of this can be traced to the situation where users got stuck against obstacles. Under the influence of force fields around objects, the force field approach gave a smoother movement than the simple constrained approach. However, it is not surprising that the force field approach gave a longer maximum stay and average stay than the unconstrained approach, for the two factors indicated above, namely, the extra computation for collision avoidance and force field generation, and unnecessary hesitation of the user (due to the fact that they did not know which method they were using).

8.2.2 Length of route

Although the unconstrained approach gave the shortest maximum stay and mean stay, it resulted in a longer overall time than the force field approach. The reason is that with the unconstrained approach, subjects usually took a longer route caused by becoming lost outside the house. The average length of the route actually taken by subjects with the unconstrained approach is 320.75 units, which is longer than that of the force field approach, 292.57 units. This indicates that mistakenly going outside the house resulted in a longer track, although some subjects took shortcuts by cutting through corners of furniture.

The actual routes taken with the simple constrained walk-through and the force field approach will basically be similar in length if the user tries to follow the designated route. However, the time-weighted length of the route can be used to estimate how well the approaches assisted subjects' navigation. Given the total number of steps N in a route, the time t_i spent at the position p_i, and the length l_i from p_{i-1} to p_i, the time-weighted length of the route is:

$$L_t = \frac{\sum_{i=1}^{N}(t_i * l_i)}{N}$$

The average time-weighted length of the routes taken by all subjects are 37.63 units for the simple constrained approach, and 24.74 units for the force field approach. This confirms that force field significantly assists the user's navigation so that it led to smoother navigation.

8.2.3 Collisions and getting trapped

The constrained and force-field methods were guaranteed to yield a collision-free route – that is, the methods guaranteed that users could not pass *through* objects. With the unconstrained navigation, none of the 15 subjects successfully completed the given task without any collisions. 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. This can be attributed 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. This kind of collision is defined as a *minor violation*. However, penetration of external boundaries usually results in a discontinuity in the view, which is defined as a *boundary violation*. 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 unconstrained method, in only 3 tests out of the total of 45 were subjects able to complete the route without penetrating boundary objects. On average each subject collided with furniture 7.3 times and passed through external boundaries (walls and ceilings) 2.7 times. More than half of the boundary violations 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 and became seriously disorientated in 20% of cases. Figure 19 shows examples where the subject became lost. It might be argued that this illustrates that the control parameters for the unconstrained navigation were too sensitive, but these were the same for each of the methods. It should also be remembered that users were trying to complete the task in the minimum time.

Without the assistance of force fields, all subjects confronted the situation, in the simple constrained approach, where they were stuck against obstacles and were not able to move forwards without moving sideways or backwards first. Figures 20 and 21 illustrate the routes taken at the corners of furniture in both the simple constrained approach and the force-field approach. With the simple constrained method, on average, subjects were stuck 2.76 times in each trial, with the poorest performer stuck 6 times.

8.3 Subjective evaluations

As well as collecting data about timings and collisions, subjects were asked for their subjective views of the methods. To estimate the features of the three different types of walk-through, subjects were required to give scores on four aspects of each method: general ease or difficulty of navigating, visual comfort, ease of going upstairs and downstairs. It is necessary to emphasise that we are dealing with a user following a specified route. Random walk-throughs have not been considered, because an unconstrained walk-through will generally give the best results for this.

The scoring system is given in Table 3. Each subject was asked to rate each aspect of each trial on a scale of one to five. Subjects were also invited to indicate any different features they discovered during each walk-through trial and point out the major difficulties they encountered when they completed the task with the different methods.

Table 4 lists the average overall scores (the sum of scores for four aspects) given by each user for all three methods, followed by the mean response for each method. The force field approach was scored the highest by all the subjects, and therefore gained the highest mean overall response. Although the unconstrained walk-through yields a similar time, its overall score is no better than the constrained walk-through, and much worse than the force field approach. This shows that users were disturbed by some unnatural experience in free navigation, although they finished the route in a marginally shorter period of time. The histograms of the scores given by subjects to the force field approach, constrained approach, and unconstrained approach are presented in Figure 22. The lowest overall response is 10 for the force field approach, 7 for the

Figure 19: Examples of subjects being lost with the unconstrained approach

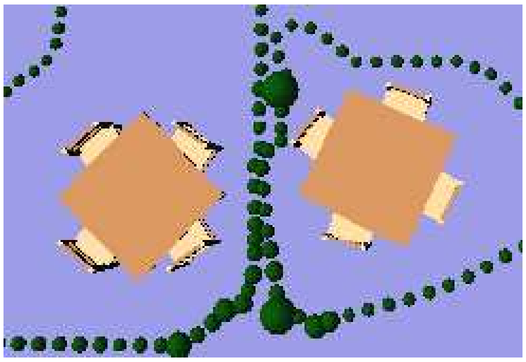


Figure 20: Smooth track with the force field approach

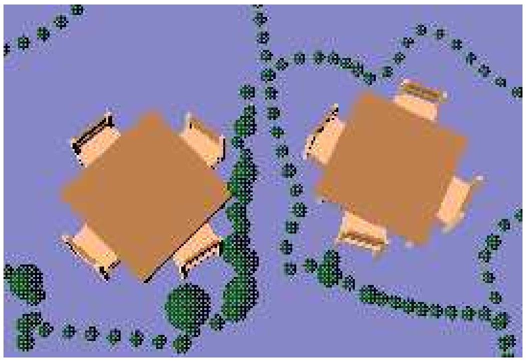


Figure 21: Being stuck at corners of furniture with the constrained approach

Aspect	Score
Easy to navigate	1 2 3 4 5
Visual Comfort	1 2 3 4 5
Easy to go upstairs	1 2 3 4 5
Easy to go downstairs	1 2 3 4 5
1 – very bad (very difficult to navigate, very uncomfortable, . . .) *2 – bad (quite difficult to navigate, quite uncomfortable, . . .)* *3 – average* *4 – good (easy to navigate, comfortable, . . .)* *5 – very good (very easy to navigate, very comfortable, . . .)*	

Table 3: Scoring system

simple constrained approach and 4 for the unconstrained approach.

Table 5 summarises separate answers to the four aspects for each method. There is no significant difference in visual comfort between the three methods. The force field made the best impression for all four aspects. The free walk-through received the lowest mean response on every aspect. This is attributable to difficulties on the stairs, especially going downstairs, which has a score less than half as good as the force field approach.

8.4 Correlation analysis

As observed during the experiments, a number of factors affected the user's decision for scoring a given method and the user's performance during walking through the scene. For an unconstrained environment, it appeared that the most disturbing phenomenon is penetrating objects. It was shown that colliding with normal objects, such as pieces of furniture, did not obviously disturb the user's navigation. It was the boundary objects, such as walls, floors, and ceilings, which caused serious disorientations when penetrated. For a simple constrained environment, it was discovered that the most disturbing phenomenon was getting trapped by objects. This was confirmed by the results collected in the experiments, which are analysed in the following section.

8.4.1 Collision-performance relationship

The effect on the score given by users of colliding with objects and the time they took to complete the route were examined for the unconstrained walk-through. The distribution of subjects' scores and times compared with the number of collisions they committed with furniture and penetrations with boundary objects are illustrated in Figures 23 and 24. There is no clear relationship between collisions and scores, or between time and collisions. However, there is a clearer link between the scores and times associated with a subject and the number of boundary violations committed (Figure 24). The more frequently a user penetrated through boundary objects, the lower the score given, and the longer the time taken.

No of Subject	*Force-Field*	*Constrained*	*Unconstrained*
1	18.7	15.0	12.3
2	14	9.0	8.7
3	15.7	9.7	9.7
4	19	14.7	14.0
5	14.7	13.7	13.3
6	18.7	15.0	7.0
7	14.3	8.0	5.3
8	16.3	9.7	12.0
9	18.7	15.0	9.7
10	12.7	12.0	8.3
11	16.0	15.7	11.0
12	16.0	13.7	11.3
13	16.3	14.7	12.3
14	13.3	11.3	9.0
15	14.7	13.0	14.3
Average	16.00	12.40	10.60

Table 4: Mean overall responses for the different methods

Method	*Mean Response*			
	Navigating	*Visual*	*Upstairs*	*Downstairs*
Unconstrained	2.47	3.15	2.10	2.05
Constrained	2.85	3.14	2.86	2.90
Force-Field	4.29	3.70	4.19	4.28

Table 5: Mean response for four aspects

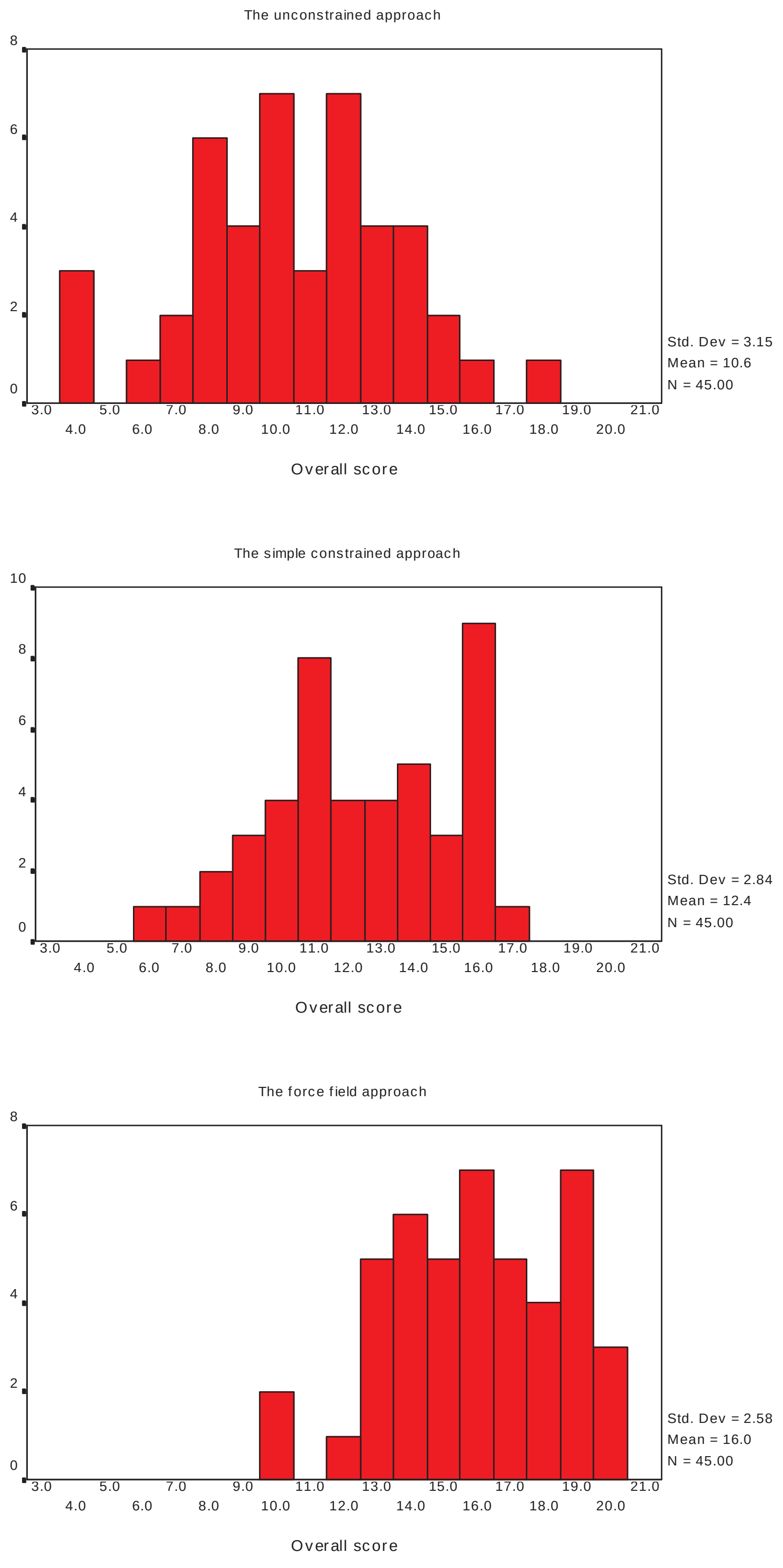


Figure 22: Histograms of responses to three methods

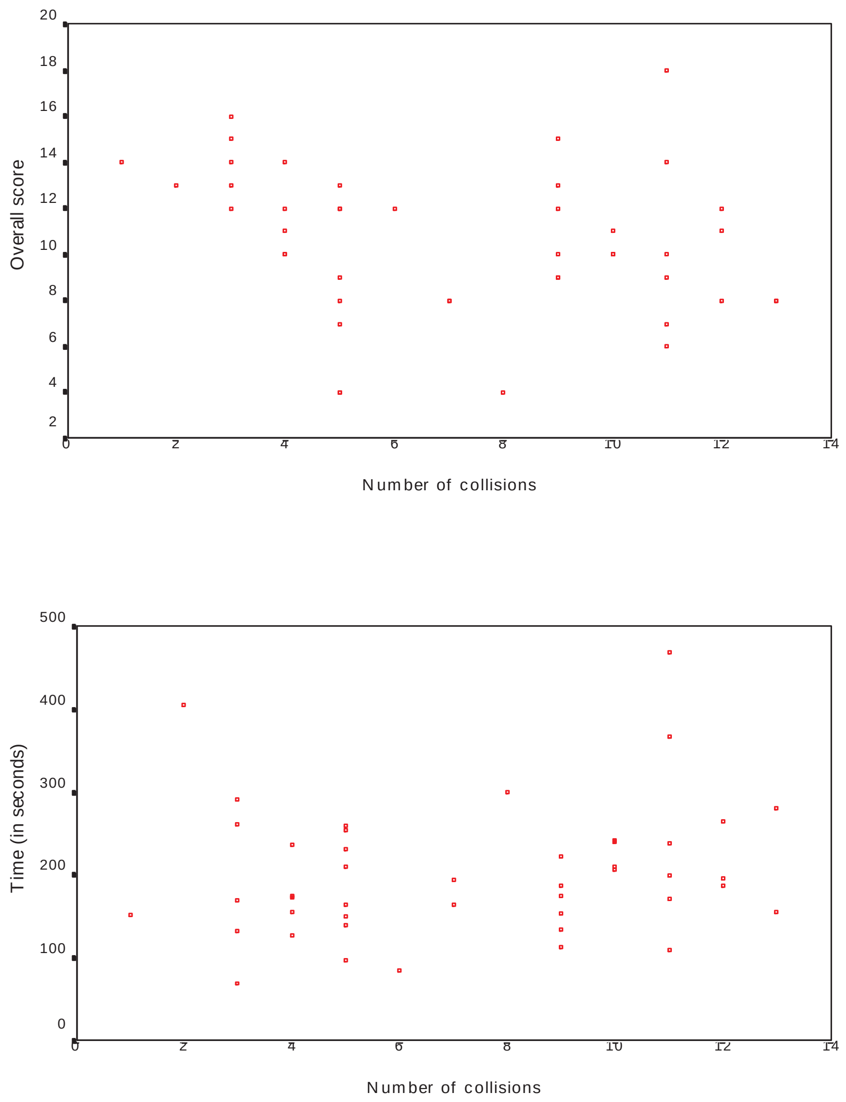


Figure 23: Impact of the number of collisions on performance (top: overall score, bottom: total time for completing the route)

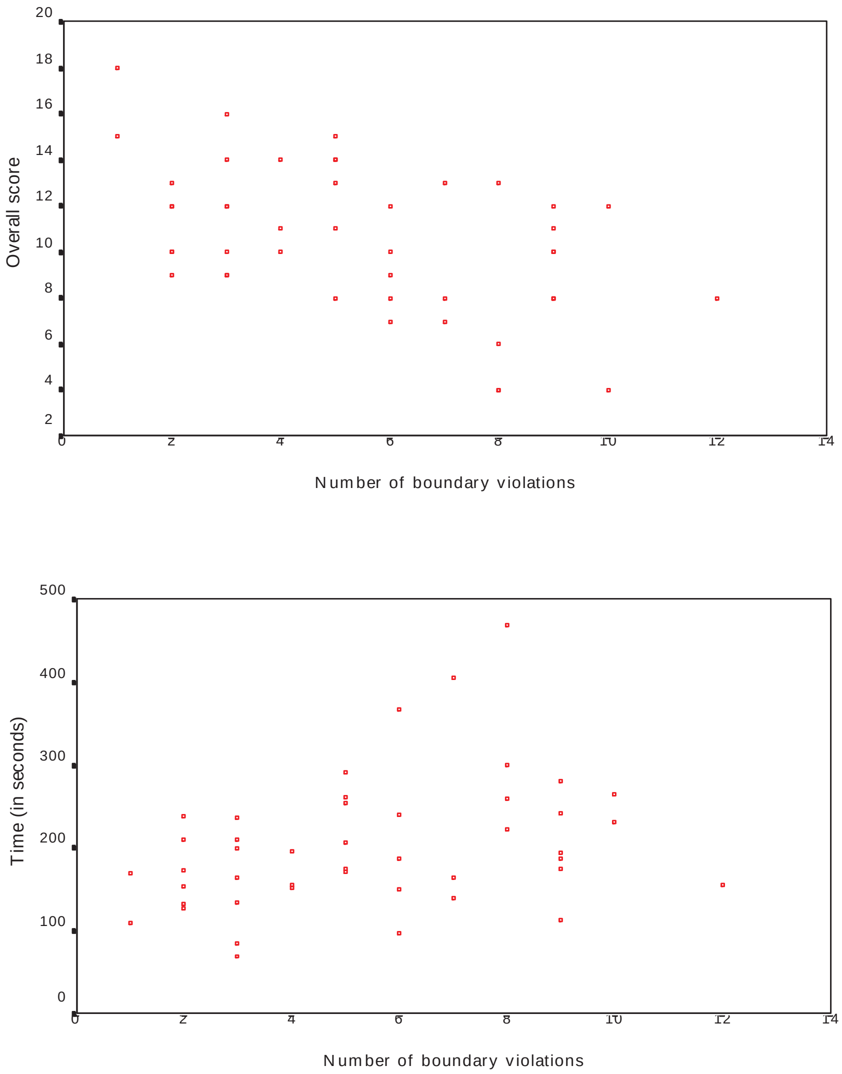


Figure 24: Impact of the number of boundary violations on performance (top: overall score, bottom: total time for completing the route)

8.4.2 Impact of getting lost

As observed during the experiments, with the unconstrained method getting lost outside the house should be the major factor in prolonging the user's navigation, resulting in low scores. This is confirmed by Figure 25. This shows a general tendency that as the number of times a subject got lost increases, the score given declines, and the time spent to finish the trial increases.

8.4.3 Relationship between performance and getting trapped

A similar relationship was expected in the simple constrained walk-through between the number of times a subject got trapped by furniture and the score given and the time taken to complete the corresponding trial. However this was not clearly confirmed by the data collected (see Figure 26). In fact, it is the seriousness of being trapped that impacts the user's performance in the navigation. A number of small incidences of being trapped by furniture will not be as disturbing or cost as much time as one serious incidence.

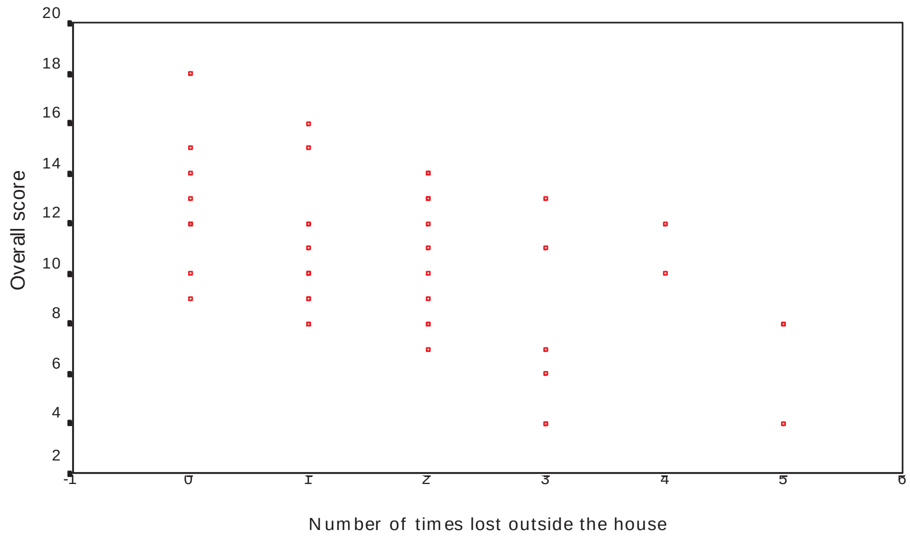
8.5 Improvement with practice

As outlined in Table 6, after practice with the unconstrained method, subjects showed little improvement on avoiding obstacles, and no decline in the number of times they got lost outside the house.

Round	*Collisions*	*Penetrations*		*Lost*
		Total Number	*On Staircase*	
Round1	8.13	5.53	3.33	3.07
Round2	7.27	5.07	3.13	1.53
Round3	6.47	5.80	3.40	1.60

Table 6: Statistics of collisions and penetrations in different rounds

However, the scores given by the subjects improved with all three types of navigation to varying degrees. Although the scores for all three methods showed some improvement throughout the three rounds (as outlined in Table 7), the highest scores gained by the unconstrained and simple constrained approaches are still much smaller than the score of the force field received in the first round. With twice the practice, the unconstrained and simple constrained approaches are still poorer than the first attempt with the force field from the user's point of view. We attribute this to the fact that much greater concentration was needed with the other two methods, subjects became tired quickly and consequently made more mistakes. Often, this resulted in them ending up outside the house in the unconstrained approach. In the constrained approach the number of times subjects were stuck were 2.87, 2.8 and 2.6 in each round; little improvement has been made. This is because although subjects only needed to worry about keeping away from objects, they still needed to concentrate to avoid getting stuck. With the



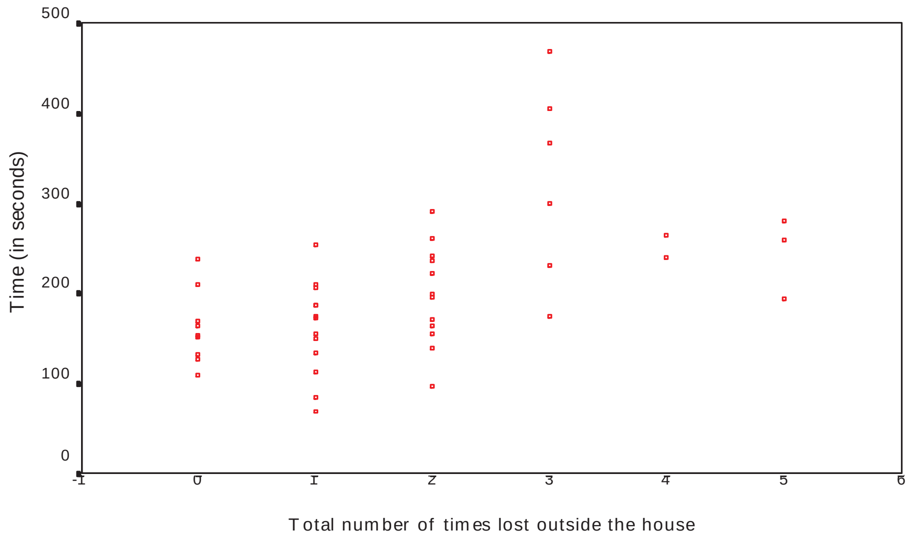
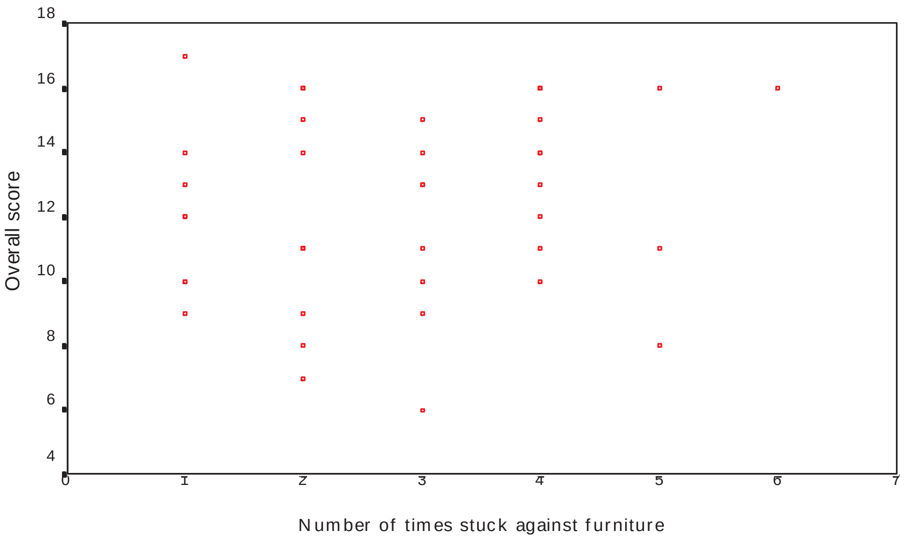


Figure 25: Impact of the number of times lost on performance (top: overall score, bottom: total time for completing the route)



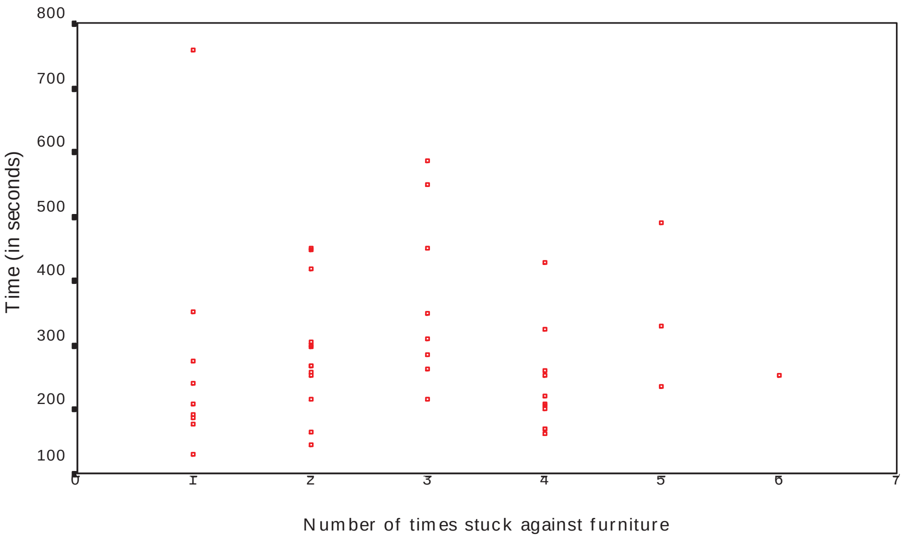


Figure 26: Impact of the number of times getting stuck on performance (top: overall score, bottom: total time for completing the route)

Method	*Overall Response*			*Range*		
	Round1	*Round2*	*Round3*	*Round1*	*Round2*	*Round3*
Unconstrained	8.67	10.33	11.67	4-14	4-12	8-16
Constrained	11.47	12.33	13.25	7-16	8-16	6-17
Force-Field	14.08	16.42	17.25	10-19	13–20	14-20

Table 7: Mean overall response in three trials

force-field method, after practice, they seemed more relaxed as they realised they could rely on the system to assist their navigation, therefore the improvement was greater.

From the time point of view (see Table 8), the three methods exhibited a similar improvement. Considering that the computation time in the force field approach is the highest, this leaves less room for improvement, while with the other two approaches, the difference between the real time and computation time is bigger, and hence there is more scope for improvement. Even so, the improvements with these two methods are still inferior to the force field method.

	Time		
	Round1	*Round2*	*Round3*
Unconstrained	4’	3’11”	2’59”
Constrained	6’07”	4’34”	4’13”
Force-Field	3’43”	3’08”	2’57”

Table 8: Average time taken in three trials

8.6 Experienced versus inexperienced users

It is important for a good walk-through system to provide assistance not only to users who are experts in the VR field, but also to inexperienced users. What was expected was that the force field should result in a good response for inexperienced and experienced users.

Volunteers in the experiments were classified into two groups: eight out of fifteen were people working in the VR or Computer Graphics fields with some experience of using 3D trackers, and seven were people had no experience of VR and had never used 3D trackers before.

Generally speaking, subjects in the inexperienced group had more difficulty in operating the trackers in the beginning, as might be expected. As seen in Table 9, inexperienced subjects were more likely to penetrate boundary objects, and to get lost outside the building. This is because they had less sense of space in a virtual environment, and lacked the skills for controlling 3D trackers. An experienced user found it was not a serious problem to come back inside, while an inexperienced user would panic when they got lost and was usually not sure how to get back inside. From this point of view, an environment with constraints and guidance should be of

Group	*Collisions*	*Penetrations*		*Lost*
		Total	*On Staircase*	
Experienced	6.75	4.12	2.75	1.13
Inexperienced	7.9	6.67	3.9	2.43

Table 9: Statistics of collisions and penetrations for different groups

more benefit to users in the inexperienced group. This is confirmed by the scores they gave in Table 10.

Method	*Mean Response*		
	Unconstrained	*Constrained*	*Force Field*
Inexperienced	9.95	12.52	15.00
Experienced	11.08	12.38	16.00

Table 10: Overall response to different methods

The improvements in time taken by two groups for completing the task are quite similar, as listed in Table 11. Experienced subjects performed better than inexperienced, which is not surprising.

Method	*Overall Time*		
	Unconstrained	*Constrained*	*Force Field*
Inexperienced	4’03”	5’35”	3’55”
Experienced	2’41”	4’19”	2’09”

Table 11: Overall time for completing the route by different methods

9 Conclusions

In this report we have outlined a new force-field navigation method with a number of novel features. First, force fields were introduced into interactive VR environments to help to guide the user past objects in the environment. Second, the algorithm performed a simple but efficient form of collision detection which guarantees that users cannot pass through solid objects. In combination, the system not only guarantees that there are no collisions with any objects in the scene during walk-through, but, more importantly, it assists the user in avoiding obstacles during navigation. And, thirdly, the method offers a simple way of including gravitational effects. The use of gravitational effects, combined with a 3D mouse-controlled interface, allows the user to move naturally up and down staircases, as well as walking across floors, while avoiding obstacles. The technique used for climbing stairs can also be used to climb on top of other objects – a table, for example – provided that it is low enough for the user to "step" onto. The technique presented here is a general solution: it does not rely on any explicit (semantic) knowledge of the virtual environment, such as "this is a staircase". The algorithm can be implemented for a variety of primitive shapes as long as they are convex, 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, stereo displays, and normal 3D workstations. Objects carried by the user can be modelled in the component hierarchy of the user's body representation as extended parts of the user's body. The user can make separate movements of those parts while walking along with the whole body.

We have also assessed from the user's point of view three different types of navigation, namely unconstrained walk-through, simply constrained walk-through, and force-field guided walk-through. The aim was to discover the degree to which each of the approaches contributes to the user's navigational proficiency in a virtual environment. A number of interesting findings were discovered by experimentation.

It is clear that a completely free environment is not adequate for certain kinds of applications, in which a more realistic walk-through is required. In this kind of application, collision detection or avoidance has to be provided. Our experiments show that, for applications in which models have confined areas separated by boundary objects, penetrating through boundary objects is the most disorienting experience, while colliding with or even walking through normal objects is much less disturbing. An example of this kind of application is walking through architectural CAD models, where penetrating walls, ceilings, or floors is much more disturbing than colliding with tables or chairs. Our results suggested that it is probably too strict for a general walk-through to make all objects solid. However, the results also suggested that making boundary objects such walls, floors, ceilings solid is very important. In applications where absolute collision avoidance is not necessary, a better solution is to make some of the objects solid, such as walls, floors, etc, and to leave other objects penetrable. Clearly, the extent to which objects in a VE should be solid is application-dependent.

It is also evident that simple collision avoidance alone is not an acceptable solution for applications where a constrained walk-through is necessary. A supplementary mechanism to assist users in avoiding obstacles is highly desirable, because getting trapped by objects has proven to be as disturbing as getting lost with an unconstrained approach.

Another finding is that incorporating gravity into a virtual environment, for applications where 'flying' through the scene is not suitable, significantly improved the user's sense of location, especially when the user's navigation involves vertical motion, such as negotiating stairways.

Finally, it was observed during our experiments, that when the user's body representation has dimensions, embodiment of the user (materialising part of the user's virtual body in the user's view) will be helpful for the user to get a sense of the spatial relationship with objects around him, such as the distance between him and obstacles, the location of an object trapping him etc.

Acknowledgements

The authors would like to thank members of the Advanced Interfaces Group for their friendship, encouragement, and many stimulating discussions during the course of this work.

References

[Ark89] Ronald C. Arkin. Motor Scheme-based Mobile Robot Navigation. *The International Journal of Robotics Research*, pages 92–112, August 1989.

[Bar90] David Baraff. Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation. *ACM Computer Graphics*, 24(4):19–28, 1990.

[Bar92] David Baraff. *Dynamic Simulation of Non-penetrating Rigid Bodies*. PhD thesis, Computer Science Department, Cornell University, 1992. Technical Report 92-1275.

[BG96] 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. ISBN 3-211-82886-9.

[BK89] 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.

[Boy79] John W. Boyse. Interference Detection among Solids and Surfaces. *Communications of the ACM*, 22(1):3–9, 1979.

[Bro92] 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.

[Cam85] S. Cameron. A Study of the Clash Detection Problem in Robotics. In *Proc. IEEE International Conf. on Robotics and Automation*, pages 488–493. IEEE Press, Piscataway, N.J., March 1985.

[CGH+97] Jonathan D. Cohen, Stefan Gottschalk, Tom Hudson, Amol Pattekar, Ming C. Lin, and Dinesh Manocha. I-Collide: Collision Detection for Arbitrary Polygonal Objects, 1997. Available at web site: http://www.cs.unc.edu/ geom/V_COLLIDE/.

[CHK97] Jon Cook, Roger Hubbold, and Martin Keates. Virtual reality for large-scale industrial applications. In *Proc. EuroVR 97 Conference*, Amsterdam, November 1997.

[CHK98] Jon Cook, Roger Hubbold, and Martin Keates. Virtual reality for large-scale industrial applications. *Future Generation Computing Systems*, 1998. In press.

[CLMP95] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav K. Ponamgi. I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments. In *Proceedings of ACM Int. 3D Graphics Conference*, pages 189–196, 1995.

[DK83] David P. Dobkin and David G. Kirkpatrick. Fast Detection of Polyhedral Intersection. *Theoretical Computer Science*, 27(3):241–253, December 1983.

[ELP86] M. Erdmann and T. Lozano-Perez. On Multiple Moving Objects. In *Proceedings IEEE Int'l Conf. on Robotics and Automation Conference*, volume 3, pages 1419–1424. IEEE Press, Piscataway, N.J., Apr 1986.

[EW96] Parris K. Egbert and H. Winkler. Collision-free Object Movement Using Vector Fields. *IEEE Computer Graphics and Applications*, 16(4):18–24, July 1996.

[FLLS93] 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.

[Hah88] J. K. Hahn. Realistic Animation of Rigid Bodies. *ACM Computer Graphics*, 22(4):299–308, 1988.

[Hub95] Philip M. Hubbard. *Collision Detection for Interactive Graphics*. PhD thesis, Department of Computer Science, Brown University, april 1995.

[Hub96] Philip M. Hubbard. Approximation Polyhedra with Spheres for Time-Critical Collision Detection. *ACM Trans. on Graphics*, 15(3):179–210, July 1996.

[HXG96] 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.

[IM92] 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.

[Kha85] O. Khatib. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. In *Proceedings IEEE Robotics and Automation Conference*, pages 500–505, 1985.

[Kro86] 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.

[Lin93] Ming C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1993.

[LMC93] Ming C. Lin, Dinesh Manocha, and John F. Canny. Fast Collision Detection Between Geometric Models. Technical Report TR93-004, Computer Science Department, University of North Carolina at Chapel Hill, Jan 1993.

[LP83] Tomás Lozano-Péreg. Spatial Planning: A Configuration Space Approach. *IEEE Trans. on Computers*, C-32(2):108–120, February 1983.

[MW88] M. Moore and J. Wilhelms. Collision Detection and Response for Computer Animation. *ACM Computer Graphics*, 22(4):289–298, 1988.

[PW90] A. Pentland and J. Williams. Good Vibrations: Model Dynamics for Graphics and Animation. *Computer Graphics*, 23(3):185–192, 1990.

[Qui94] Sean Quinlan. *Real-Time Modification of Collision-Free Paths*. PhD thesis, Robotics Laboratory, Stanford University, 1994.

[Rey87] Craig W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. *ACM Computer Graphics*, 21(4):25–33, 1987.

[Sha92] Clifford A. Shaffer. A Real-Time Robot Arm Collision Avoidance System. *IEEE Transaction on Robotics*, 8(2):149–160, 1992.

[SU93] Mel Slater and Martin Usoh. Representations Systems, Perceptual Position, and Presence in Immersive Virtual Environments. *Presence*, 2(3):221–233, 1993.

[SUS95] 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.

[TN87] William C. Thibault and Bruce F. Naylor. Set Operations on Polyhedra Using Binary Space Partitioning Trees. *Computer Graphics (SIGGRAPH'87)*, 21(4):153–162, July 1987.

[Tur89] G. Turk. Interactive Collision Detection for Molecular Graphics. Master's thesis, Computer Science Department, University of North Carolina at Chapel Hill, 1989.

[WB95] Andrew Witkin and David Baraff. An Introduction to Physical Based Modelling. *ACM SIGGRAPH'95 Course Notes*, 1995. Available at web site: http://www.cs.cmu.edu/afs/cs/user/baraff/www/pbm/pbm.html.

[XH98] Dongbo Xiao and Roger Hubbold. Navigation guided by artificial force fields. In Joelle Coutaz and John Karat, editors, *Proceedings CHI '98 Technical Programme*, pages 179–186. ACM SIGCHI, ACM, April 1998.

[Xia97] Dongbo Xiao. *Interactive display and intelligent walk-through in a virtual environment*. PhD thesis, Department of Computer Science, University of Manchester, October 1997.

[ZPOM93] M. J. Zyda, D. R. Pratt, W. D. Osborne, and J. G. Monahan. NPSNET: Real-Time Collision Detection and Response. *Visualization and Computer Animation*, 4(1):13–24, January 1993.