

## Simulating Virtual Human Crowds with a Leader-Follower Model

Tsai-Yen Li, Ying-Jiun Jeng, and Shih-I Chang  
Computer Science Department, National Chengchi University  
64, Sec.2, Chih-Nan Road, Taipei, Taiwan 11605, ROC  
{li, s8633, s8624}@cs.nccu.edu.tw

### Abstract

Although virtual human has been an active research topic for many years, most researches focus on simulating various aspects of humanoid instead of a crowd of people. Many early researches in computer animation address the issue of simulating flocking behavior for creatures such as birds and fishes, but these results do not apply directly to virtual crowd simply because human beings possesses a higher degree of intelligence than other animals. In this paper, we report our progresses on generating collision-free gross motions for multiple virtual crowds. Each virtual crowd consists of a leader and many followers. The leader is in charge of generating motions for its own group with the motions of other crowds taken into account. The followers use artificial life principles to follow the leader as it moves to the goal. The high degrees of freedom involved in the crowd simulation system present a great challenge for gross motion planning. We adopt a decoupled path-planning approach, in which the paths being executed by the other leaders become the motion constraint of the current leader under consideration. In addition, we take a unified view of search space to extend the planner to consider a whole crowd instead of its leader only. Experimental results show that our planner can efficiently generate satisfactory motions. We believe that such a planning system is a good addition to controlling groups of avatars in a 3D virtual environment.

**Keywords:** Virtual Crowd Simulation, Shared Virtual Environment, Decoupled Path Planning, Artificial Life, and Motion Planning for Multiple Moving Objects.

### 1. Introduction

3D graphics has been one of the key technologies in bringing rich multimedia contents to our daily life via broadband network services. Among the possible applications, 3D shared virtual environment is becoming prevalent in the cyberspace. Consequently, the demands for better authoring tools to control groups of avatars in a virtual environment are also increasing. For example, a well-

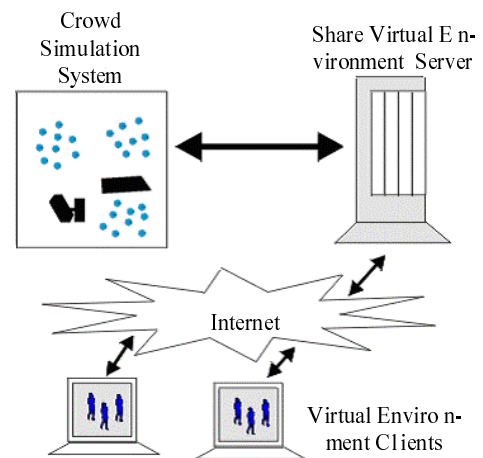


Figure 1. Overall system architecture.

controlled crowd of people will increase the realism of virtual shopping in a 3D shopping mall. Therefore, the owner of a virtual shop might want to hire virtual crowds to attract real users to their stores. Similar needs might also arise in a distributed networked game, where the computer might simulate multiple virtual players simultaneously. However, generating the avatar motions for a virtual crowd in real time remains a great challenge after these years of researches in Artificial Intelligence and computer animation. Some previous work suggests a distributed approach of adopting reactive rules for each virtual creature. These emergent behaviors have been shown to work well in simulating the motion for animals such as birds and fishes. However, there are no guarantees on the existence of a feasible global plan that can bring these virtual creatures to their goals. In contrast to these emergent behaviors, human beings 'plan' in addition to 'react'. Therefore, a more intelligent mechanism is needed to simulate realistic crowd motions in a virtual environment.

A common client-server architecture for a shared virtual environment (SVE) system is depicted in Figure 1. The crowd simulation system is the main focus of this paper. We will present a distributed path-planning scheme for

simulating multiple virtual crowds. We assume that each virtual crowd is led by a *leader*, who is in charge of planning the gross motions for its own group with a given goal location specified at run time. The other members of the group, called *followers*, will follow the leader closely as it moves to the goal. By *gross* motions, we mean that we will only consider the locations of the avatars and assume that the corresponding locomotion and character animation can be generated accordingly as long as the gross motions are continuous. Although this leader-follower model may not be applied to all situations of crowd simulation, it serves as a good experimental model to simulate organized groups of people in contexts such as virtual shopping scenarios.

Since we allow multiple groups to move simultaneously, the computational complexity for such a path-planning problem could be rather high. Therefore, we use a decoupled planning approach, in which the paths being executed by other leaders become the motion constraint of the current leader under consideration. The leader's motion will then become one of the driving forces that attract the followers to follow its motion closely. The path planner for a leader also estimates the shape and space occupied by each group during path execution so that the motion for the whole group, not just the leader, can be considered. This planner has been integrated with a commercial 3D virtual world, and the adequacy and efficiency of such a planning scheme are demonstrated through several examples to be presented in this paper.

We organize the remaining of the paper as follows. In the next section, we will review the researches pertaining to our work in computer animation, artificial life and path planning. In the third section, we will give an overview of the planning problem considered in our system and our decouple approach to this problem. In the fourth section, we will present some principles in artificial life for follower motions and describe how to avoid local minima traps in the artificial force field. In the fifth section, we will describe how we estimate the shape and size of a virtual crowd at a given location and how we account for this size in the path planner. Then we will address some issues on implementation and experiments. Finally, we will conclude the paper in the last section.

## 2. Related work

Simulating emergent behaviors such as flocking has been widely used in creating realistic animations for groups of virtual creatures such as fishes or birds [17][18][19]. By applying simple emergent rules to each character, one can simulate realistic flocking behavior for animals. However, it is difficult to simulate a crowd of people simply with these principles because human, as an

intelligent character, possesses higher degree of intelligence.

In recent years, there have been many research efforts in incorporating practical artificial intelligence techniques to create real-time animations. For example, a cognition model has been proposed in [7] to use a more complete control loop to simulate an intelligent character. Researches in virtual human also consider the problem of creating realistic humanoid group motions through various levels of controls [5][14][15][16]. However, most of these researches do not account for geometric reasoning capability such as path planning. On the other hand, motion-planning techniques have been successfully used in automatic movie generation [8] or customized tour guiding [13], although they usually focus on generating dexterous motions for a single human only. Our earlier work [12] proposed the idea of using motion planning techniques as well as the principles from artificial life [9] to direct virtual crowds interactively. However, the planning capability for an avatar in that work is limited to the leader only instead of the whole group. In addition, since the motions for the followers are only guided by artificial forces, they could easily get trapped due to environmental obstacles.

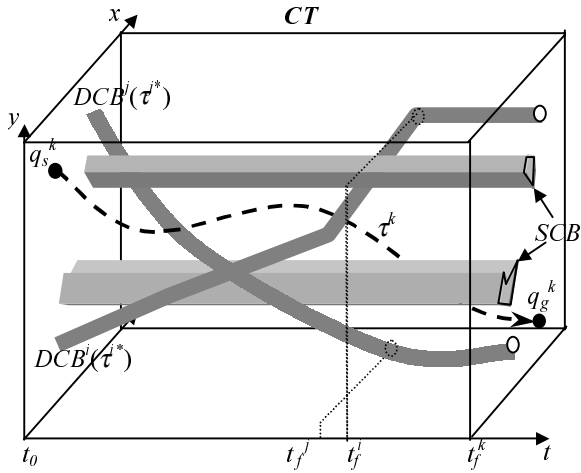
In robotics, efficient motion planning algorithms have been proposed to control more than one robot arm in an on-line manner [11]. However, we have not seen similar work been applied to simulating human crowds. Distributed interactive simulation (DIS) and shared virtual environments (SVE) have also been active research fields for more than a decade. Most research efforts focus on system scalability, transmission efficiency, and scene management. In recent years, more and more SVE systems (such as ActiveWorlds [1] and Blaxxun's [4]) include programming interfaces for implementing virtual avatars (or called *bots*). However, they do not have a systematic way to simulate virtual crowds.

## 3. Decoupled planning for multiple leaders

In this section, we consider the basic problem of generating collision-free gross motions for multiple group leaders. We will first give a general description of the planning problem and necessary assumptions we made. Then we will present our approach to this basic problem and conclude this section by some examples from our experiments.

### 3.1. Basic path-planning problem

We are given a 2D polygonal description of the obstacles in a virtual world. The virtual world is also crowded with groups of avatars simulated by our system. Each group consists of a *leader* and multiple *followers*. In this

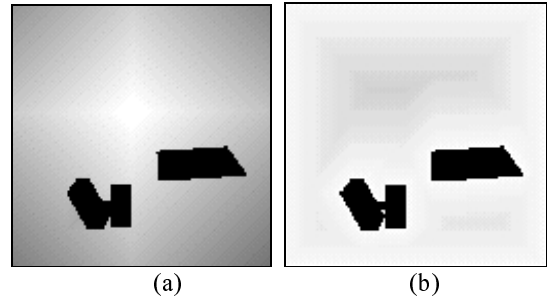


**Figure 2. Searching for a feasible path amongst obstacle regions in the CT-space.**

section, we will only consider the motions for the leaders and take the followers into account in the next sections. We assume that each leader has three degrees of freedom (DOF's)  $(x, y, \theta)$  when they move on the ground. The parameter space for the  $i$ th leader, called the *Configuration Space* (or *C-space* for short), is denoted by  $C_i$ . The overall C-space for the whole system, denoted by  $C$ , is the composite space of each individual C-space  $(C_1 \times C_2 \times \dots \times C_m)$ , where  $m$  is the number of leaders in the world. At any time during the simulation, our system has to make sure that the generated motions for all leaders are realistic and safe. In other words, the motions must be continuous in  $C$  and collision-free from other leaders and obstacles in the environment. Each leader, when becoming idle, will be assigned a new goal configuration at run time through an interactive interface or a script.

In order to reduce the complexity of the planning problem, we further assume that each avatar (leader or follower) is represented by an enclosing circle of radius  $r$ . Due to the geometric symmetry of a circle, we can reduce the DOF's for each leader to two by temporarily ignoring the  $\theta$ -dimension. The value for  $\theta$  will be computed in a post-processing step after a path has been generated. For example, we can require that a leader always face the tangential direction of a path. In addition, in order to make collision detections more efficient during planning, we use a discrete approach by representing the polygonal obstacles with a bitmap and then grow the obstacle boundary by  $r$  to form the C-space obstacles for each leader. This computation only needs to be done once initially when obstacle configurations become known.

Although the path-planning problem has been widely studied for the past three decades, one still cannot escape the curse of dimensionality when solving such a problem.



**Figure 3. Examples of (a) artificial potential field and (b) distance map in a workspace (values in gray scale).**

Indeed, the planning problem becomes very difficult for systems with high DOF's such as coordinating the motions of multiple mobile robots. The scenario of virtual crowds inherently has such high complexity. For example, the dimension of the composite C-space ( $C$ ) for the whole virtual avatars is  $2m$ , where  $m$  is the number of virtual avatars. Since the size of a C-space grows exponentially in the number of dimensions, a complete planning system deems to be infeasible.

### 3.2. Decoupled planning approach

In the robotics literature, the approaches to solving the path-planning problem for multiple robots fall into two categories: *centralized* and *decoupled*. The centralized approaches consider the composite C-space of the whole system, which could be impractical to search exhaustively. Some practical randomized algorithms sacrificing completeness are often used in this approach [2]. On the other hand, the decoupled approaches usually only consider one robot at a time. In one such decoupled approach, each robot is planned independently and then their motions are coordinated by velocity tuning techniques [6]. Another decoupled approach assumes that robot motions are generated sequentially, and each robot is planned under the constraint of other robots whose motions are generated earlier.

In our crowd simulation system, we take the last decoupled approach by decomposing the overall planning problem for multiple leaders into smaller subproblems. Each of these subproblems considers one leader at a time under the constraints of other leaders' motions. The same approach has been successfully used in planning the motions of multiple robotic arms in an on-line manner [10]. Although this approach is not complete in nature, it fits our application quite well since the needs for path planning happen sequentially. The planner is called on demand when the goals for the leaders are specified interactively at run time by a user or by a script. Therefore, it becomes

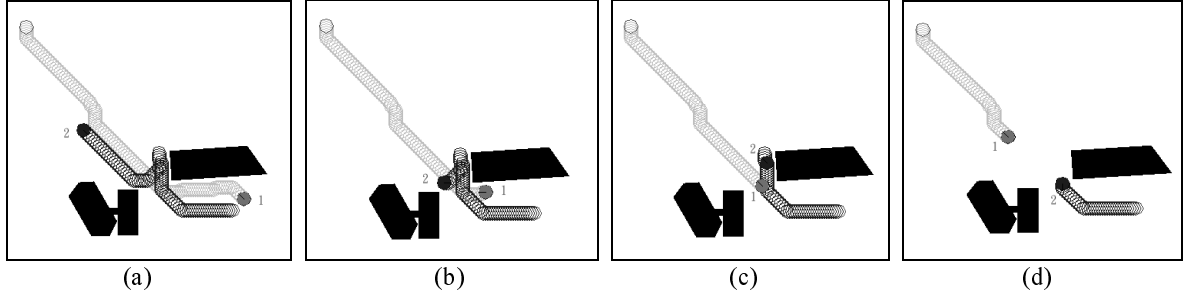


Figure 4. Example 1: an example of the basic planning problem for leaders' coordinated motions.

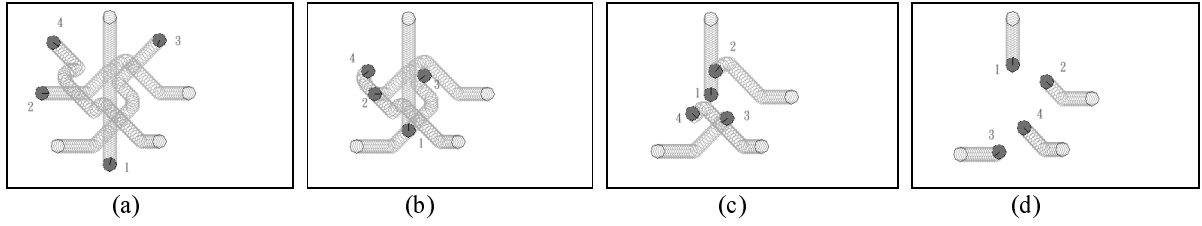


Figure 5. Example 2: an example of coordinated crossing motions for four leaders.

unnecessary to determine the order in which the leaders are planned.

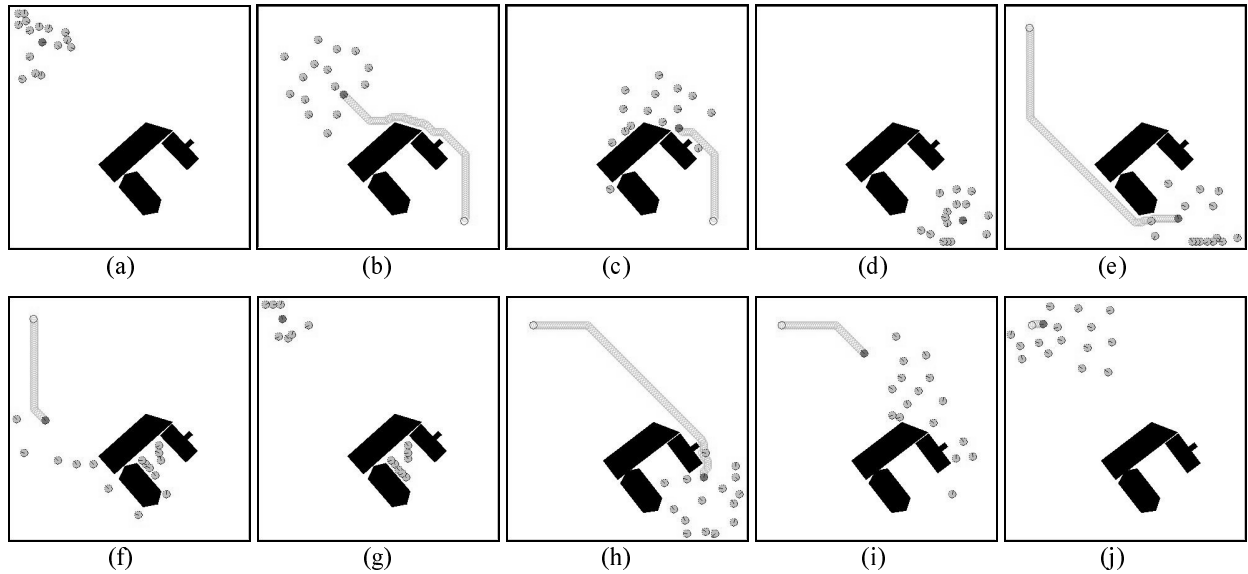
At the instant when the goal for a leader is specified, the planner will try to compute a path for the leader that does not cause any collisions with other group leaders as well as any static obstacles in the environment. The path of the  $i$ th group leader (denoted by  $\tau^i$ ,  $i = 1$  to  $m$ ) is known as a function of time  $t$ , including when it is static. In our decoupled approach, for  $k$ th leader under consideration we augment its C-space by the time dimension to form the so-called *Configuration-Time Space (CT-space)*. A conceptual example is depicted in Figure 2. There are two types of forbidden regions in the CT-space representing obstacle regions that the leader should avoid entering. One (denoted by *SCB*) is due to the static obstacles while the other (denoted by *DCB*) is due to other moving leaders. Note that *SCB* is axis-parallel extrusion of 2D obstacles in time while *DCB*'s are curve extrusions of the obstacle regions imposed by other moving leaders. When the  $i$ th leader finishes its motion at time  $t_f^i$ , we assume that it will stay there unless otherwise instructed. Equivalently, we are extending the path for the  $i$ th leader to infinity and this extended path is denoted by  $\tau^{i*}$ . The objective of the path planner is to find a collision-free path  $\tau^k$  for the  $k$ th leader in the CT-space that can connect the current ( $q_s^k$ ) configuration at the current time ( $t_0$ ) to the specified goal configuration ( $q_g^k$ ) at some time ( $t_f$ ) in the future. For realistic simulation, the velocity of an avatar must be within some reasonable limit; therefore, the slope at any point along a legal path in this CT-space must be positive (because time

is not reversible) and less than some user-specified value (maximal velocity).

With the constraints mentioned above, we conduct the search in the CT-space in a best-first fashion based on the potential value of an artificial potential field. This type of potential field is widely accepted as a good heuristic for motion-planning problems [3]. For efficiency consideration, we only construct a 2D potential field accounting for static obstacles. An example of the potential field computed by the so-called NF1 algorithm is shown in Figure 3(a). This potential field algorithm and the standard best-first search algorithm for path planning can be found in [10]. The algorithm returns a legal collision-free path when the search succeeds and gives up when all possible configurations have been visited. Note that a path is legal only if it can remain collision-free for the whole period when all other avatars are active. Therefore, we require that a goal configuration in the CT-space must have a time value that is equal to or greater than the latest finish time of all other leaders. For instance, in Figure 2,  $t_f^k$  (the final time for  $\tau^k$ ) must be greater than  $t_f^i$ , for all  $i \neq k$ .

### 3.3. Examples of coordinated motions

In Figures 4-5, we show two examples of coordinated motions generated by our planner. Four snapshots are taken in each example. The paths remaining to be executed in each instance are shown as circle traces while the current configurations are in solid dots. The numbers beside the leaders indicate the order in which they are asked to plan.



**Figure 6. Examples of follower motions in a virtual crowd. (a)-(d) show regular flocking behavior (e)-(g) show motions with trapped followers, and (h)-(j) show a better tracking method escaping the trap.**

The planning times of these examples are measured on a regular PC with 600MHz D uron CPU and 128MB RAM.

In Example 1 (Figure 4), leader 1 's motion was planned before leader 2 's, and it became a motion constraint for leader 2. Consequently, leader 2 has to wait for leader 1 to pass the narrow passage amongst obstacles before it can proceed. The planning times are 20ms and 220ms for leader 1 and 2, respectively. Example 2 is a complex example involving four leaders moving across a common region from different directions. As the number of leaders become large, it becomes almost impossible for a human to coordinate their motions by inspection. The average planning time for each avatar in this example is only 105ms.

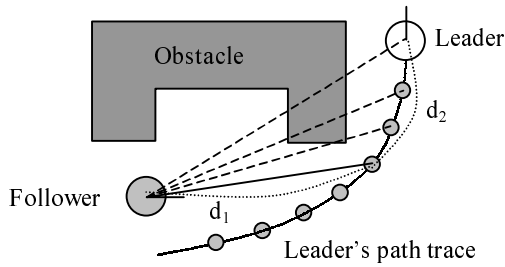
#### 4. Follower motions in a virtual crowd

Compared to other animals, human crowd possess characteristics unique to human beings. For example, unlike other animals' flocking behavior where the group relation is formed automatically, a leader, acting as a tour guide in a human group, is specified explicitly, and it is self-conscious of being a leader. The followers will then follow the motion of their designated leader closely. In this section we will first describe how we use the basic principles from artificial life to compute steering forces for controlling the followers' motions. Then we will show with an example how to modify the forces to avoid the problem of local minima. In the last subsection, we will

use an example to demonstrate a simple extension of the steering forces to simulate emergent evacuation.

##### 4.1. Basic flocking behavior

To simulate human grouping behavior, we adopt a strategy similar to the one proposed in [18]. The strategy uses various attractive and repulsive forces to create steering behaviors. In each control cycle of the simulation, an avatar perceives other avatars and environmental obstacles in its limited view cone and reacts by adjusting its velocity according to the composite force. For example, three steering forces (*separation*, *cohesion*, and *alignment*) were suggested to determine how an avatar reacts to other avatars in its neighborhood. Separation force is computed according to the repulsive forces exerted by all of its nearby avatars within the view cone. Cohesion is computed by applying an attractive force from the average position of its neighbor avatars in the same group. Alignment is computed by averaging the velocity of the nearby avatars of the same group. Note that only avatars in the same group exert the cohesion and alignment forces to each other while avatars in different groups can still affect each other with the separation forces. In addition to these three forces, we also apply a repulsive force to an avatar according to its distances from the nearby environmental obstacles. Furthermore, the leader of a group also applies a major attractive force to its followers. This attractive force, proportional to the Euclidean distance from the leader, drives the followers to their leader.



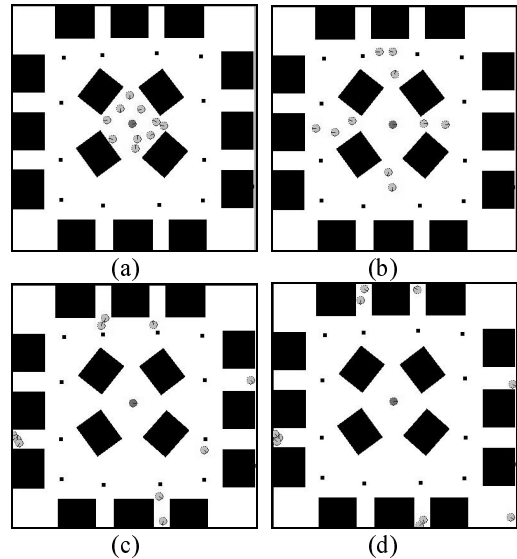
**Figure 7. Escaping local minima by tracking leader's path trace.**

These five forces altogether are normalized and then re-weighted before they are composed. The weight of each force is dynamically adjusted according to the current world status and the past history. For example, if the force causes a follower to collide with an obstacle, the weight of the repulsive force from the obstacle will be increased. When the collision disappears, the weight for this force will incrementally go down to its nominal value. However, a follower still may bump into obstacles because the repulsive and attractive forces cancel each other. In this case, a sliding force along the obstacle boundary is applied to pull the followers toward the leader. Furthermore, collisions between two followers sometimes might not be avoidable, and the situation may even result in deadlocks in the worst case. Therefore, whenever collisions between followers are detected, we make the attractive forces from various sources inactive for a few cycles and resume afterward. During this time, each follower will attempt to move in a different random direction in order to resolve the deadlock situation. An example of crowd motions with the flocking behavior is shown in Figures 6(a) -6(d).

#### 4.2. Escaping local minima

Although the principles described in the previous subsection work very well for simulating flocking behavior for human crowds, there are still situations where followers may get trapped in a local minima of the force field formed by several sources. For example, in Figures 6(e)-6(g), followers are trapped in the U-shaped obstacles. The local minima exist mainly due to the cancellation of the attractive force from the leader and the repulsive force from the obstacles. The randomized approach adopted in the previous subsection for resolving collisions between avatars might not work here since the local minima could be very deep and difficult to escape.

Intuitively, a human follower can easily track a leader even if he/she cannot see the leader for a short period of time. When a human follower cannot see the leader, he/she usually moves toward the location where the leader was



**Figure 8. An example of emergent evacuation behavior**

last seen. With this observation we modify the attractive source to the leader's last-seen location. We record the path trace of the leader and try to connect the follower to the latest point in this path where a collision-free straight-line segment can be connected as depicted in Figure 7. Furthermore, since the magnitude of attractive force is computed according to the real distance, not Euclidean distance, from the leader, we have to modify the distance computation routine accordingly. For example, the distance between the leader and the follower in Figure 7 is set to  $d_1 + d_2$ . Figures 6(h)-6(j) show an example similar to the trapped case shown in Figures 6(e)-6(g). With the improved strategy the followers can easily escape the local minima and quickly catch up the leader as it moves to the goal.

#### 4.3. Emergent evacuation behavior

Although the principles described in this section have been shown to be an effective way for simulating flocking behavior in a virtual crowd, it remains an empirical process in tuning the weights of various artificial forces to obtain a desirable emergent behavior. On the other hand, the flexibility of tuning these forces allows one to experiment with various settings. An interesting example, as shown in Figure 8, can be created by changing the sign of the forces for flocking behavior to simulate emergent evacuation behavior. In this example, the followers gather around the leader at the initial configuration and start to run away from the leader when the mode is switched to the evacuation mode. Other interesting modes for simulating various

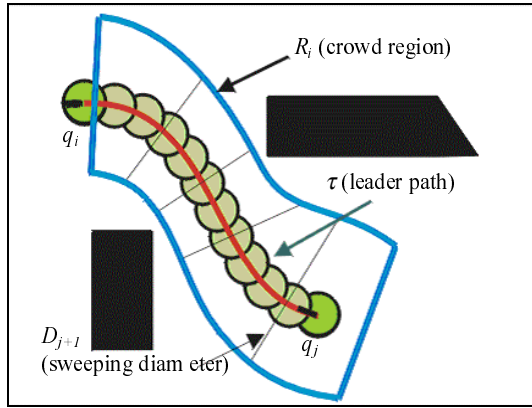


Figure 9. Crowd region estimation.

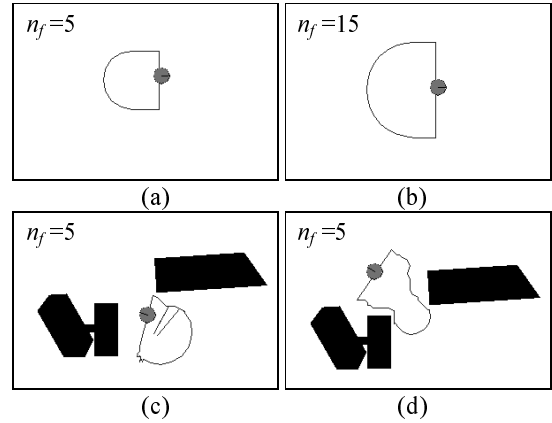


Figure 10. Examples of estimated crowd regions.

behaviors of human crowds could be further developed with a similar approach.

## 5. Motion planning for multiple crowds

In this section we will extend the path planner described in the Section 3 to consider the motions for not only the leaders but also the crowds they are leading. We need to address two new issues in dealing with this challenging problem. First, we need to have an effective way to model the shape of the crowds. Second, we have to plan for the crowds with the shape model we adopt.

### 5.1. Problem description

According to the artificial rules adopted for the followers, we can assume that they will closely follow their leader and form a crowd of some shape behind the leader. However, since the crowd is flexible and conforming to the environment at run time, it is difficult to predict their shapes. Even if we can model the shape of a crowd, the planning problem is still not straightforward. In fact, we should account for crowd motions with two objectives in mind. First, when a leader plans its motion, it should take into account not only other leaders but also the crowds they lead. Second, a leader should not only plan for itself, but also its followers behind. In other words, we are extending the path-planning problem for multiple objects of fixed geometry to the problem for multiple objects of variable shapes (each crowd is treated as a flexible object). However, we do not know any planners in the literature that can plan for this type of flexible objects.

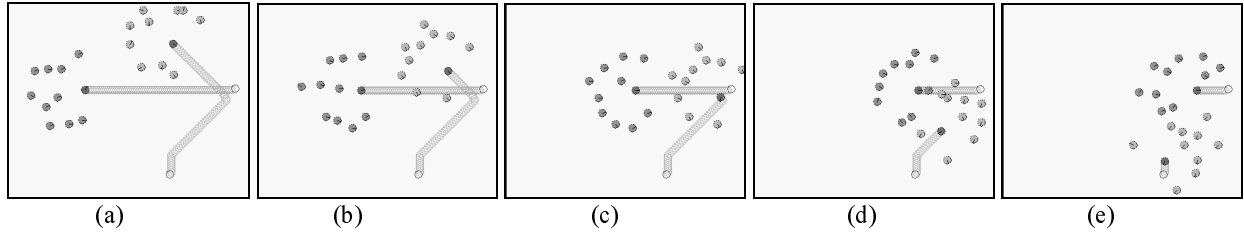
Although generating a precise plan for multiple flexible objects is difficult, we think, intuitively, a real human leader in a group does have some principles in planning the motions for its group. For example, a leader normally

should not lead its crowd to cut through other crowds or be cut through by other crowds. In this section, we will describe our attempt to model this planning principle and to generate a reasonably good plan for crowd motions.

### 5.2. Crowd shape estimation

We will try to model the shape of a crowd region, estimate its size, and then obtain estimation on the longitudinal depth of the moving crowd in this subsection. We assume that the followers will gather behind the leader; therefore, the trace of the leader forms an axis for the crowd's estimated shape as shown in Figure 9. The longer the path segment we consider, the larger the region behind the leader that can accommodate followers. The minimal length of this path segment depends not only on the number of followers but also on the location of this segment in the environment.

We use a half circle of radius  $D_i$  behind the leader to estimate the area for accommodating followers.  $D_i$  is a function of the number of followers as illustrated in Figures 10(a) and 10(b), where  $n_f$  is the number of followers in the group. If the area cannot accommodate all followers, we will sweep the half circle backward along the trace of the leader's path until the area for the crowd region ( $R_i$ ) is large enough. In addition, instead of being fixed,  $D_i$  could be further limited due to the leader's location and the surrounding environmental obstacles. To get an idea of how far the leader is from the obstacles, we compute a numerical L1 distance map (denoted by  $DM$ ) as shown in Figure 3(b). Each cell in  $DM$  stores its minimal distance from obstacle boundaries, which is a good indication of the radius of the collision-free circle around the location. If this distance value is smaller than  $D_i$ , then  $D_i$  will be reset to this value. As depicted in Figures 10(c) and 10(d), the crowd region changes its shape as the leader moves



**Figure 11. An example of motion plans for two virtual crowds consisting of 10 followers each.**

through the narrow passage. Once we have determined the path for a leader, we can use this method to compute the length ( $l_i$ ) of path segment for a configuration  $q_i$  that we do not want other crowds to cut through. We call  $l_i$  the *longitudinal depth* of a crowd when the leader is at configuration  $q_i$ .

### 5.3. Planning crowd motions

We extend the decoupled path planner described in Section 3 to account for the motions of other crowds in addition to their leaders. The motions for other leaders are known and used as dynamic obstacles for the leader under consideration. Now the dynamic obstacles are extended from a single configuration to a trace of configurations for some length. This length is the longitudinal depth that a crowd extends and can be computed by the above method for crowd shape estimation. Therefore, all we have to do is modify the collision detection routine in the planner to treat each leader as an extended obstacle along a list of configurations. The number of configurations to check for each leader will depend on the longitudinal depth of its group at the instant. By doing so, we can ensure that we will plan a path that does not cut through other crowds.

However, how do we know that a leader will not lead its group to a situation where other groups will cut it through? In fact, we cannot guarantee so because the subject of our planning is the leader only. To address this problem, we have to consider more than the leader. For example, we can plan for a list of leader configurations along the longitudinal depth of the crowd. However, we do not know the past trace of a leader at a given configuration simply because we are still in the process of searching for a feasible path. Consequently, in this case we assume that  $l_i$  be a fixed value proportional to the number of followers. During the search, a configuration is legal only if the leader will remain collision-free for duration of  $l_i$  starting from the current configuration in the CT-space.

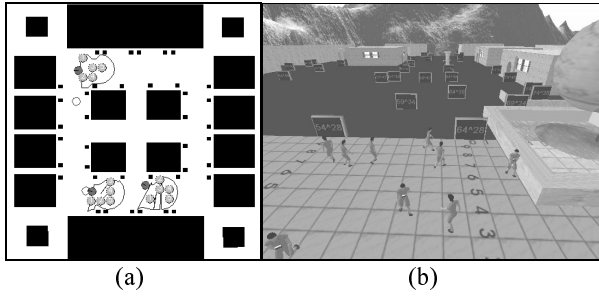
### 5.4. Considerations for planning efficiency

Assume that there are  $m$  crowds in the environment. The longitudinal depth for the crowd under consideration is  $l^c$ , and the average longitudinal depth for other crowds is  $l^o$ . Then according to the extension described in the previous subsection, we have to increase the number of calls to the collision detection routine between avatars for each configuration we visit from  $m-1$  to  $(m-1) \cdot l^c \cdot l^o$ . The performance for such a planner will definitely suffer because it depends on the sizes of the crowds, and collision detections are the most fundamental routine in the search process.

However, with the following observations, we can reduce the run-time complexity of collision checks between leaders to constant time. We note that the shape of C-obstacle between two leaders (two smaller circles) is an enlarged circle. Therefore, instead of checking collisions between two circles at run time, we can build a CT-space, such as the one shown in Figure 2, by marking the trace of the enlarged circle as obstacles along the time dimension. In addition, when we consider the longitudinal depth of another crowd at a configuration, we can actually copy the enlarged circle for that configuration *forward* in time for the required length. Furthermore, when we consider the longitudinal depth of the crowd under planning, we have to make sure that a configuration remains collision-free for a fixed period of time. This is equivalent to copy all obstacle regions in CT-space *backward* in time for that duration. Therefore, by copying obstacles regions forward and backward in CT-space, we can reduce the planning problem to searching for a collision-free path for a point in the extended CT-space. This unified view of obstacles in the extended CT-space can greatly reduce the computation time for planning crowd motions.

### 5.5. Examples

In Figure 11, we show snapshots of crowd motions generated by our planner for 10 followers. Two crowds, depicted in green and blue, respectively, are trying to move across a common area at the same time. For clarity, no obstacles are placed in the workspace. The motion for the first crowd is straightforward since the second crowd is static at the time of planning. Thus, a straight-line path



**Figure 12. Graphical user interfaces for crowd simulation: (a) 2D control interface (b) 3D Active Worlds browser.**

toward to the right is generated. When the planner is evoked for the second leader, it has to take the motions of the first crowd and its own followers into account. Consequently, the planner for the second crowd decides to cross the path segment on the right for the first crowd. Note that if we consider the leader only, the second leader should not have moved its crowd to the right to such an extent. Instead, it can move directly downward without colliding with the first leader. However, if this is the case, these two crowds will tangle for a long time before they can be separated again. Therefore, the example in Figure 11 demonstrates that our planner can generate an effective motion strategy that might be taken by a real human leader in leading a group of people.

## 6. Implementation and experiments

We have implemented the crowd motion planner and simulator described above in Java as a standalone simulation module. We have also integrated the planner with a virtual environment system called Active Worlds via its Java API. In Figure 12 we show sample screen dumps of this system with its 2D and 3D graphical user interface for a more realistic world. The 2D interface is presented by the planner at the server side for interactive crowd controls while the 3D browser, presented by Active Worlds, appears at the client machine. The planner only needs to set the position of an avatar, and the browser will take care of generating continuous locomotion and appropriate character animation.

The planner is initially given a geometric description of the obstacles in the world from a 2D -layout file. The world is represented by a grid of 128x128, and the same resolution is used in the CT-space to search for a feasible path. The planning time for each call to the planner is usually only fractions of a second, which is appropriate for our interactive use. For example, for the case in Figure 11, the planning times for the first and second crowds are 500ms and 2000ms, respectively, on a considered low-end PC.

**Table 1. Performance improvement with extended CT-space computation.**

	straightforward collision-check method	collision checks w/ extended CT-space
Test Case	Planning time (ms)	Planning time (ms)
1	880	60
2	4770	270
3	5930	550
4	1370	110
5	15650	990
Average	5720	396

Note that the planning time for a given problem instance is independent of the complexity of environmental geometry since we use a precomputed bitmap for collision checks with the environment. The planning time depends on the number of followers since this number determines the size of estimated crowd region. In addition, the planning time also depends on the number of leaders coexisting in the virtual world since it affects the number of inter-avatar collision checks for each visited configuration.

If one compares the cases of generating coordinated motion for a leader only and for the whole crowd, the second one is no doubt more complex and time consuming. However, for the examples reported in previous section, the planning time for the second case seems to remain at the same degree as for the first one. The main reason for this efficiency is mainly due to the unified view of obstacles in the extended CT-space as explained in the previous section. Through efficient implementation of duplicating obstacle regions, we can reduce the original problem to a search problem in the CT-space. In Table 1, we compare the performance improvement of this new method over the original straightforward method in five test cases. The new method is about an order of magnitude faster than the original one. The planning times reported here are taken for examples with two crowds of size six each. Although the number of crowds and their sizes will affect planning times, the table focuses on comparing the relative efficiency of using different collision-detection methods. In fact, the larger the number of followers, the more the saved time.

## 7. Conclusions

In conclusion, we have proposed a planning system capable of generating coordinated gross motions for multiple crowds in a virtual environment. Such a planning problem is difficult because of lacking an effective crowd model and the high degrees of freedom involved. We have presented our initial attempt to address this planning problem

to come up a good, if not guaranteed, motion plan that can not only bring the crowds to their goals but also avoid interference between crowds. This planner features a crowd shape estimation module, integrated view of crowd motion planning, and an efficient implementation of search routines. We also have proposed a successful motion strategy that allows the followers to closely follow the leader without being trapped by environmental obstacles. The planning capability and efficiency have been successfully demonstrated by several examples presented in this paper. The planner has also been integrated with a commercial virtual world to simulate realistic crowd motions. We believe that such a planning system can not only enhance the functions of a shared virtual environment but also open a new direction for real-time computer animation for human crowds.

## Acknowledgement

This work was partially supported by grants from the National Science Council, ROC, under contract NSC89 - 2218-E-004-008.

## References

- [1] ActiveWorlds, URL: <http://www.activeworlds.com>.
- [2] J. Barraquand, L. Kavraki, J.C. Latombe, T.Y. Li, and P. Raghavan, "A Random Sampling Scheme for Path Planning," *International Journal of Robotics Research*, 16(6):759-774, Dec. 1997.
- [3] J. Barraquand and J. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *International Journal of Robotics Research*, 10:628-649, 1991.
- [4] Blaxxun Community Server, URL: <http://www.blaxxun.com>.
- [5] T.K. Capin, I.S. Pandzic, N. Magnenat-Thalmann, D. Thalmann, "Integration of Avatars and Autonomous Virtual Humans in: Networked Virtual Environments", *Proceedings of ISCI 98*, IOS Press, pp. 326 - 333, Amsterdam, Netherlands, 1998.
- [6] M. Erdmann and T. Lozano-Perez, "On Multiple Moving Objects," AI Memo No. 883, Artificial Intelligence Laboratory, MIT, 1986.
- [7] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive Modeling: Knowledge, Reasoning, and Planning for Intelligent Characters," *Proceedings of ACM SIGGRAPH*, pp. 29-38, 1999.
- [8] Y. Koga, K. Kondo, J. Kuffner and J.-C. Latombe, "Planning motions with intentions," *Proceedings of ACM SIGGRAPH'94*, pp. 395-408, 1994.
- [9] C.G. Langton, "Artificial Life," in C. G. Langton, editor, *Artificial Life, Volume VI of SFI Studies in the Sciences of Complexity*, pp. 1-47, Addison-Wesley, Redwood City, CA, 1989.
- [10] J.C. Latombe, *Robot Motion Planning*, Kluwer, Boston, MA, 1991.
- [11] T.Y. Li and J.C. Latombe, "Online Manipulation Planning for Two Robot Arms in a Dynamic Environment," *International Journal of Robotics Research*, 16(2):144-167, 1997.
- [12] T.Y. Li, J.W. Lin, Y.L. Liu, and C.M. Hsu, "Interactively Directing Virtual Crowds in a Virtual Environment," *Proceedings of the Tenth International Conference on Artificial Reality and Tele-existence*, Taipei, 2000.
- [13] T.Y. Li, J.M. Lien, S.Y. Chiu, and T.H. Yu, "Automatically Generating Virtual Guided Tours," *Proceedings of the Computer Animation '99 Conference*, Geneva, Switzerland, pp. 99-106, 1999.
- [14] S.R. Musse, F. Garat, D. Thalmann, "Guiding and Interacting with Virtual Crowds in Real-time," *Proceeding of Eurographics Workshop on Animation and Simulation '99 (CAS '99)*, pp. 23-34, Milan, Italy, Springer, 1999.
- [15] S. R. Musse and D. Thalmann, "A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis," *Proceedings of Eurographics Workshop on Computer Animation and Simulation '97*, Springer Verlag, pp.39-51, 1997.
- [16] I.S. Pandzic, T.K. Capin, E. Lee, N. Magnenat-Thalmann, D. Thalmann "Autonomous Actors in Networked Collaborative Virtual Environments", *Proceedings of MultiMedia Modeling '98*, pp. 138-145, IEEE Computer Society Press, 1998.
- [17] C. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Proceedings of ACM SIGGRAPH'87*, pp.25-34, 1987.
- [18] C. Reynolds, "Steering Behaviors For Autonomous Characters," *Proceedings of Game Developers Conference*, 1999.
- [19] X. Tu and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior," *Proceedings of ACM SIGGRAPH'94*, 1994.