Appear in *Proceedings of International Conference on Distributed Multimedia System (DMS2001)*, Taipei, Taiwan, Sept. 2001

Planning Coordinated Motions for Multiple Virtual Human Crowds

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

Abstract

Simulation of virtual humans has been an active research topic in the past few years. Most of these researches focus on simulating various aspects of humanoid instead of a crowd of people. On the other hand, several early researches in computer animation address the issue of simulating flocking behaviors for creatures such as birds and fishes. However, these results do not apply directly to virtual crowd simulation because human beings possess higher degree of intelligence than other animals. In this paper, we attempt to address the issue of generating collision-free motions in an on-line manner for multiple virtual crowds. Each virtual crowd is led by a group leader, who is in charge of generating motions for its own group with the motions of other crowds taken into account. These motions should be collision-free from obstacles in the environment as well as other crowds of variable size. The high degrees of freedom involved in the crowd simulation system present a great challenge for path 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 avatar motions in a 3D virtual environment.

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

1. Introduction

As computer graphics hardware and network communication technologies are rapidly evolving, 3D shared virtual environments have become prevalent in the cyberspace. Consequently, demands for better authoring tools to control groups of avatars in a virtual environment are also increasing. For example, a well-controlled crowd of people will increase the realism of virtual shopping in a 3D shopping mall. 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 some virtual players. However, generating the motions of a virtual human crowd in real time remains a challenging research after these years of development in Artificial Intelligence. Some previous work suggests a distributed approach of adopting reactive rules for each individual avatar. 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.

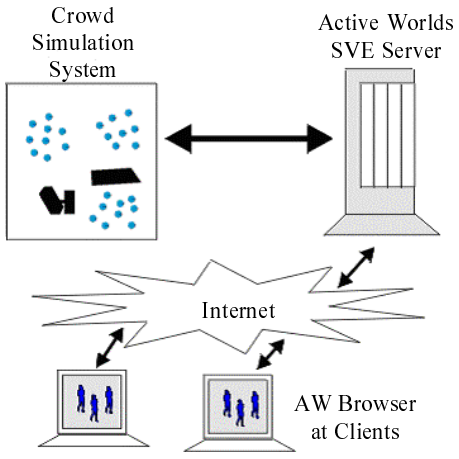


Figure 1. Overall system architecture

A common client-sever 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 generating the motions of multiple virtual crowds. Each group of virtual humans is led by a *leader*, who is in charge of planning the motion for its own group with a given goal location specified at run time. Since we allow multiple groups to move simultaneously, the computational complexity for such a path-planning problem is rather high.

Therefore, we adopt a decoupled planning approach, in which the paths being executed by other leaders become the motion constraint of the current leader under consideration. The planner can also estimate 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 artificial life and path planning. In the third section, we will give an overview of the planning problem considered in our system and our decoupled approach to this problem. In the fourth 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 present some implementation and experimental issues. Finally, we will conclude the paper in the last section.

2. Related Work

Simulation of emergent behaviors such as flocking has been widely used in creating realistic animations for groups of virtual creatures such as fishes or birds. [16][17] 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 animation. For example, a cognition model has been proposed in [8] 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][6][14][15] However, most of these researches do not account for geometric reasoning problems such as path planning. On the other hand, motion-planning techniques have been successfully used in automatic movie generation[9] 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 of artificial life [10] to direct the virtual crowds interactively. However, the planning capability for a leader avatar in that work is limited to the leader only instead of the whole group.

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 simulate 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 focused 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 systematic ways to simulate virtual crowds.

3. Decoupled Planning for Multiple Leaders

In this section, we consider the basic problem of generating collision-free 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. We will also 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 crowded with groups of avatars simulated by our system. Each group consists of a *leader* and multiple *followers*. In this section, we will only consider the motions for leaders and take followers into account in the next section. We assume that each leader has three degrees of freedom (DOF's) (x, y, θ) when they move on the ground. The parameter space for each 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 ($C_1 \times C_2 \times \ldots \times C_m$), where m is the number of leaders. At any time during the simulation, our system has to make sure that the generated motions for each leader is realistic and safe. In other words, the motions must be continuous in C and collision-free from other leaders and the obstacles in the environment. Each leader, when becoming idle, will be assigned a goal configuration at run time through an interactive interface or a script.

In order to reduce the complexity of the planning problem, we 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 θ-dimension. The value for θ 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

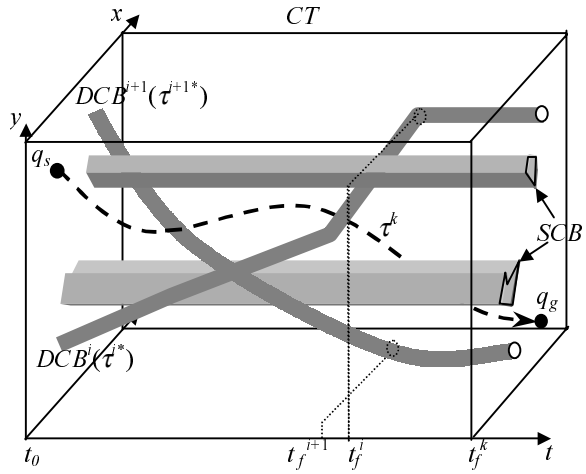


Figure 2. Searching for a feasible path amongst obstacle regions in the CT-space

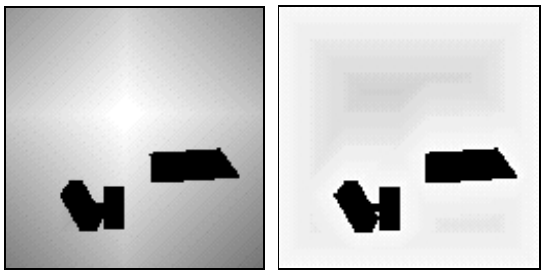
bitmap and then grow the obstacle boundary by r to form the obstacle regions, called *C-obstacles*, in the C-space for each leader. This computation only needs to be done once initially when obstacle configurations become known.

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. 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 number of dimensions for the composite C-space (C) is $2n$, where n 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 for 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. [7] 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

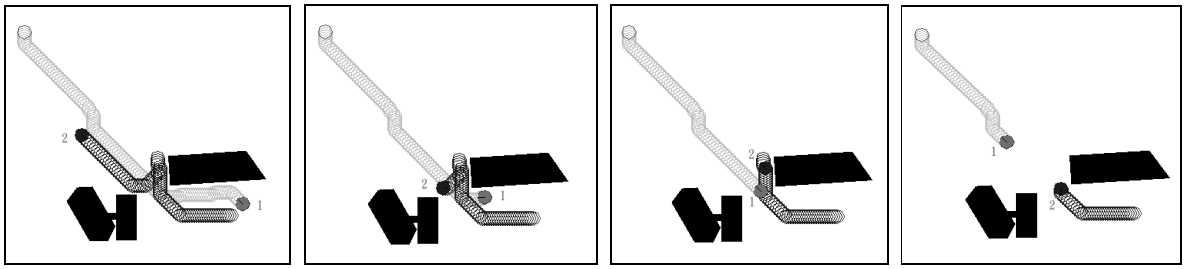


(a) (b)

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

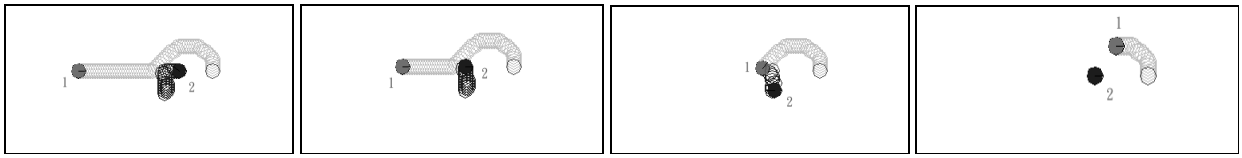
multiple leaders into smaller subproblems. Each of these subproblems considers one leader at a time under the constraints of other leaders' motions. The same approach has been successfully used in planning the motions of multiple robotic arms in an on-line manner.[11] Although this approach is not complete 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.

At the instant when the goal for a leader is specified, the planner tries to generate a path for the leader that does not cause any collisions with other group leaders as well as the static obstacles in the environment. The path of the ith group leader (denoted by τ^i, i = 1 to m) is known as a function of time t even if it is static. In our decoupled approach, we augment the C-space for the kth leader under consideration by the time dimension to form the so-called *Configuration Time Space* (*CT-space*), as illustrated in Figure 2. There are two types of forbidden regions in the CT-space representing obstacle regions that the leader should avoid entering. One (*static C-obstacles*, denoted by *SCB*) is due to the static environmental obstacles while the other (*dynamic C-obstacles*, 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 due to other moving leaders. When the ith 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 ith leader to infinity and this extended path is denoted by τ^{i*}. The objective of the path planner is to find a collision-free path τ^k for the kth 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 a virtual avatar must be within some reasonable limit; therefore, the slope at any point along a legal path in this CT-space must be positive (time is not reversible) and less than some user-specified value (maximal velocity).



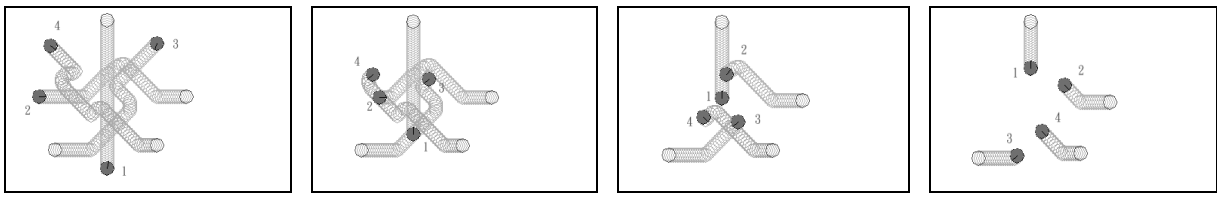
(a) (b) (c) (d)

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



(a) (b) (c) (d)

Figure 5. Example2: an example of coordinated motions requiring path extension



(a) (b) (c) (d)

Figure 6. Example 3: an example of coordinated crossing motions for four leaders

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 environmental obstacles. An example of the potential field is shown in Figure 3(a). The best-first 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 τ_k) must be greater than t_f^i (for all $i \neq k$).

3.3. Examples of coordinated motions

In Figures 4-6, we show three examples of coordinated motions generated by our planer. 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 aside the leaders indicate the order in which they are planned. The planning times for these examples are measured on a regular PC with 600MHz Duron 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. In Figure 5, we show an example where leader 2 can reach its goal earlier but it is not allowed to do so because its goal configuration is in the way of leader 1's motion. The path planner is required to ensure that no collisions could happen till all other leaders stop. That is, the time value of a goal configuration in the CT-space must be larger than the finish times of all other motions. The planning times for this example are 170ms and 1052ms for leader 1 and 2, respectively. Example 3 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 leader in this example is only 105ms.

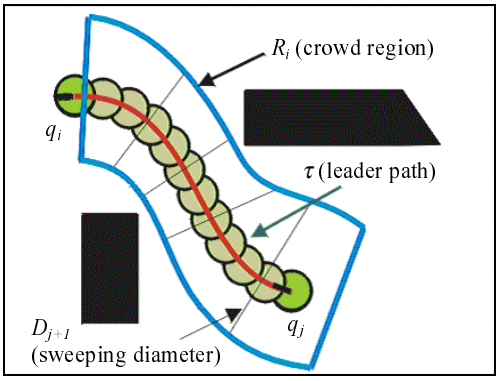


Figure 7. Crowd region estimation

4. Motion Planning for Multiple Crowds

In this section we will extend the path planner described in the previous section 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.

4.1. Problem description

When the motion of a leader has been determined, the motions for its followers can be generated either by planning or by reactive rules. In either case, for the crowd motions to be realistic, we can assume that the followers 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. 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.

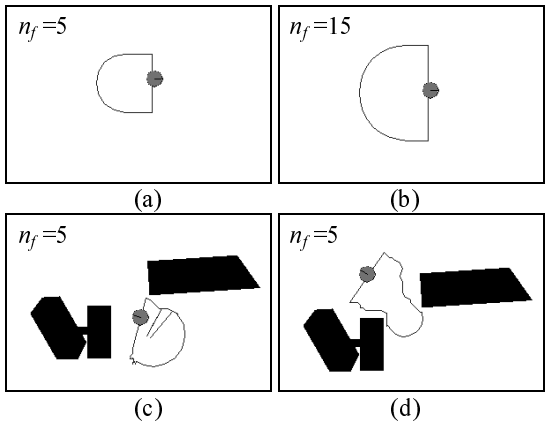


Figure 8. Examples of crowd regions: (a)(b) are for different number of followers, and (c)(d) are at different locations

4.2. 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 7. 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 8(a) and 8(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 adjusted according to the surrounding environment where the leader stands. To get an idea of how far the leader is from environmental obstacles, we compute a numerical L1 distance map (denoted by DM) as shown in Figure 3(b). Each cell in DM stores the minimal distance of this location 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 8(c) and 8(d), the crowd region changes its shape as the leader moves through a 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

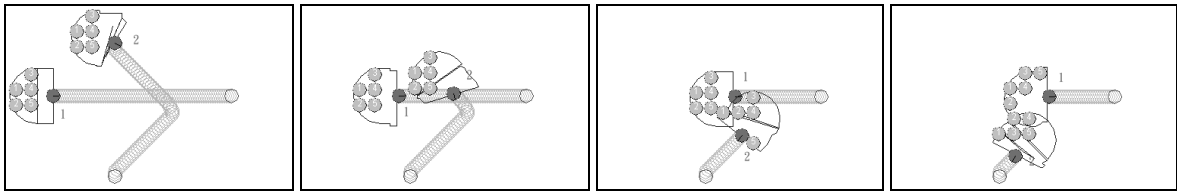


Figure 9. Example of motion plans for two virtual crowds consisting of 5 followers

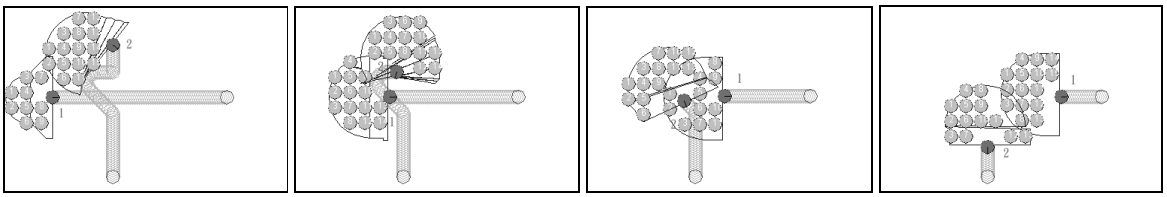


Figure 10. Another example of motion plans for two virtual crowds consisting of 15 followers

l_i the *longitudinal depth* of a crowd when the leader is at configuration q_i.

4.3. Planning crowd motions

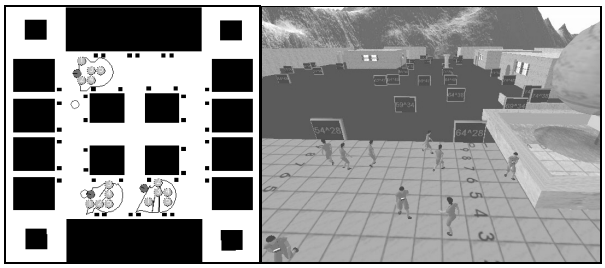
We first extend the decoupled path planner described in Section 3 to account for the motions of other crowds in addition to their leaders. The motions for other leaders are known and used as dynamic obstacles for the leader under consideration. Now the dynamic obstacles are extended from a single configuration 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 consider a list of configurations for each leader treated as a dynamic obstacle. 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? Unlike the previous problem, 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. Therefore, 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 form the current configuration in the CT-space.

4.4. 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 inter-avatar collision detection routine for each configuration from m-1 to $(m\text{-}1)\times(l^c)\times(l^o)$. 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 and time-consuming routine during the search.

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 enlarge 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 planning time for generating crowd motions.



(a) (b)

Figure 11. Graphical user interfaces for crowd simulation: (a) 2D control interface (b) 3D Active Worlds browser

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

4.5. Examples

In Figures 9 and 10, we show snapshots of crowd motions generated by our planner for 5 and 15 followers, respectively. The leader and followers in a crowd are depicted in green and blue, respectively. Two crowds 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. A straight-line path 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. In the case in Figure 9, since there are only 5 followers, the longitudinal depth of the crowd is short enough for it to safely cross the path segment not yet executed on the right. However, in the second case of 15 followers, the size of the crowd prohibits the leader to take such a risky move. Instead, the only feasible plan is to wait until the first crowd pass the common area and then continue leading its crowd downward. Note that instead of being waiting, the second crowd makes a smart move to the left so that it can pass the first crowd earlier.

5. Implementation and Experiments

We have implemented the crowd motion planner 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 11 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 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 9, the planning times for the first and second crowds are 60ms and 611ms, respectively. For the case in Figure 10, the planning time for the second crowd increases to 1322ms, which is still reasonable for our interactive application. The planning time is longer in this case partly because it takes a longer time (about 70% of planning time) to estimate the shape of a larger crowd.

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. In fact, the larger the number of followers, the more the saved time.

6. Conclusions

In conclusion, we have proposed a planning system capable of generating coordinated 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 routine. 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," in *International Journal of Robotics Research*, 16(6):759-774, Dec. 1997.

[3] J. Barraquand and J. Latombe, "Robot Motion Planning: A Distributed Representation Approach, " in *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. ISCIS 98, IOS Press*, pp. 326 – 333, Amsterdam, Netherlands, 1998.

[6] T. Capin, I. Pandzic, N. Magnenat Thalmann, D. Thalmann, "Avatars in Networked Virtual Environments", John Wiley & Sons, 1999.

[7] M. Erdmann and T. Lozano-Perez, "*On Multiple Moving Objects*," AI Memo No. 883, Artificial Intelligence Laboratory, MIT, 1986.

[8] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive Modeling: Knowledge, Reasoning, and Planning for Intelligent Characters," *Proceedings of ACM SIGGRAPH*, pp29-38, 1999.

[9] Y. Koga, K. Kondo, J. Kuffner and J.-C. Latombe, "Planning motions with intentions,", *Proceedings of ACM SIGGRAPH'94*, pp 395-408, 1994.

[10] 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.

[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," in *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," in *Proceedings of the Computer Animation '99 Conference*, Geneva, Switzerland, pp99-106, 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] I.S. Pandzic, T.S. Capin, E. Lee, N. Magnenat-Thalamann, D. Thalmann "Autonomous Actors in Networked Collaborative Virtual Environments", *Proceedings of MultiMedia Modeling '98*, pp. 138-145, IEEE Computer Society Press, 1998.

[16] C. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model, *Proceedings of ACM SIGGRAPH'87*, pp.25-34, 1987.

[17] X. Tu and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior, " *Proceedings of ACM SIGGRAPH'94*, 1994.