Interactive Control of Large Crowd Navigation in Virtual Environment Using Vector Field

Xiaogang Jin, Jiayi Xu, Charlie C.L. Wang, *Member, IEEE*, Shengsheng Huang, and Jun Zhang

***Abstract*— Providing interactive control is a hot topic in the research of crowd navigation. In this paper, we propose a simple but effective way for authoring crowd scene. The movement of each pedestrian is composed of an autonomous part and a user specified one, the ratio between them can be interactively adjusted. The governed part is realized by Radial Basis Functions (RBF) based vector fields. With this governing tool, users can easily drive the flow of crowds by sketching velocities on anchor points in the scene. Our approach is fast enough to allow on-the-fly modification of vector fields.**

***Index Terms*— Crowd Animation, Navigation Control, Radial Basis Functions, Anisotropic RBF, Vector Field.**

I. Introduction

MASSIVE human crowds are ubiquitous in the real world, which leads to the necessity for simulating such scenarios in realistic interactive environment. In the last decade, crowd simulation techniques have been increasingly explored and many efforts have been made to modulate intuitive navigation control and real crowd behaviors. Existing navigation control techniques can be classified into bottom-up methods that generally manipulate interactions between independent agents, or top-down methods that in contrast rely on the user specified global path planning scheme. The scenarios generated by the agent based bottom-up frameworks that focused on the individual behaviors may result in less realistic crowd behaviors as the interactions between agents or the environment by the guiding rules are often in a local scope. When coupled with global navigation, this kind of methods offers realistic pedestrian motion planning. However, with increasing number of pedestrians, it becomes computationally expensive. Therefore, the bottom-up methods are not appropriate for the real-time crowd simulation with large number of people. Different from the agent based methods, the global behavior of crowds can be easily designed by incorporating the top-down methods. In this way, a road-map or a vector field is utilized to cover the entire virtual environment and handle the flow of crowds and long-term obstacle avoidance. Nevertheless, controlling crowd behaviors in a global setting without any wandering behaviors among individuals would not be adequate.

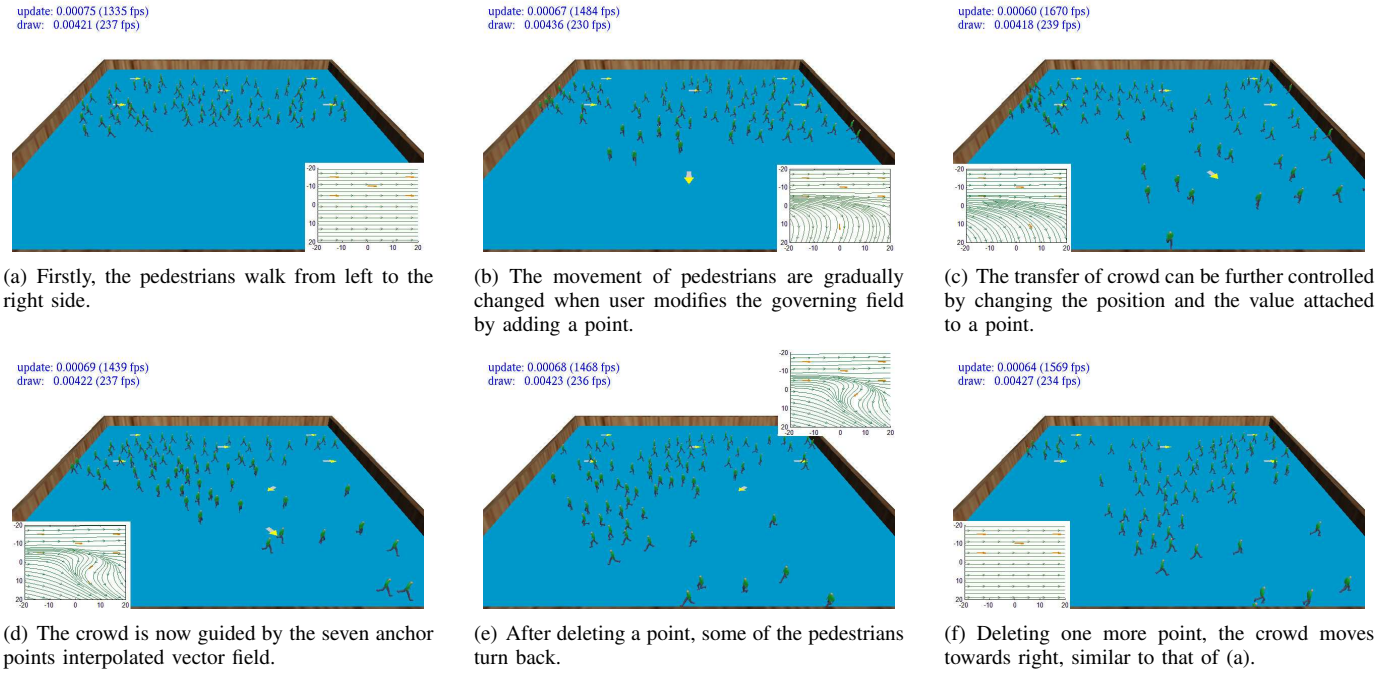
In the research of crowds navigation, interactive path design is often a desired function during simulation. However, most of the existing algorithms only adopt the initial-value based approaches and consider changing the movement when moving obstacles are encountered. In other words, the goals and trajectories are specified at the beginning of simulation. Obviously, this kind of control methods lacks flexibility and may not be helpful in creating large crowd animation in complex scenes. In the existing approaches such as [1], [3], the interactivity is not as intuitive and simple as ours.

Xiaogang Jin (Zhejiang University, China), Jiayi Xu (Zhejiang University and Chinese University of Hong Kong, China), Charlie C.L. Wang* (Corresponding author; Chinese Univeristy of Hong Kong, China; E-mail: cwang@mae.cuhk.edu.hk), Shengsheng Huang (Zhejiang University), and Jun Zhang (Shanghai Institute of Film Art, China).

Fig. 1. A medieval siege scenario simulated with our method.

For example, in the scenario shown in Fig. 1, several groups of pedestrians are simulated, each of which acts differently towards different goals. The behaviors of pedestrians should be adjusted from time to time. Here, we employ a field based method to provide an effective and efficient control tool of large crowd navigation. To make the behavior of pedestrians more real, autonomous components and the local interaction among agents is also added to each individual. Details will be given in the implementation section.

Main Results: In this paper, we propose a simple but effective method for interactively controlling the navigation of groups of agents. In our hybrid architecture, each individual is modeled as a particle with its own personality, whose actual behavior is determined by a globally governed movement component and an autonomous movement component. The globally governed component is derived from a user specified vector field, which is generated by the anisotropic RBF based vector interpolation. The autonomous one consists of a velocity proportional wandering and a weighted movement derived from the positions of anchor points. With this practical crowd navigation controlling



(a) Firstly, the pedestrians walk from left to the right side.

(b) The movement of pedestrians are gradually changed when user modifies the governing field by adding a point.

(c) The transfer of crowd can be further controlled by changing the position and the value attached to a point.

(d) The crowd is now guided by the seven anchor points interpolated vector field.

(e) After deleting a point, some of the pedestrians turn back.

(f) Deleting one more point, the crowd moves towards right, similar to that of (a).

Fig. 2. The interactive control of one vector field with arrows indicating the anchor points. The small sub-pictures show the flow field generated in each step.

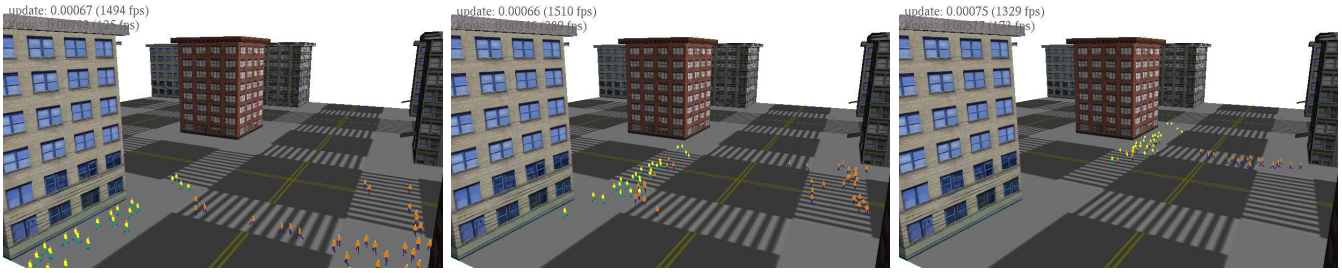


Fig. 3. Two groups of pedestrians walk in the city. As they move across the streets, several agents from different groups join together.

method, we can easily modify the behaviors of crowds on-the-fly without hurting the real-time performance. In the current implementation, we provide the following editing tools by mouse and keyboard interactions:

1) editing, adding or removing anchor vectors of the flow field;
2) adjusting the ratio of the governed and the autonomous movement of each agent.

Our approach has been implemented and tested on several scenarios, as shown in Fig. 1 and Fig. 3. Fig. 2 gives an example showing our interactive control tools. Furthermore, the vector field generated by our method is smooth and can be fast recalculated during the simulation, which are very good properties for interactive control of large crowd navigation in virtual environment. The movement of the crowd of virtual pedestrians is directed by the underlying continuous vector field without the need of explicit grid-based representation. Thus no resolution limitation is posed and a lot of memory space is saved.

A. *Previous work*

At present, the multiple motion planning solutions for crowds are either in the top-down manner or in the bottom-up way. In practice, top-down methods are usually employed together with other tuning algorithms to generate more plausible results (e.g., [1]), where the systems analyze and store particular scene information which is useful for path planning in the form of road maps, graphs or vector fields. Agents receive scene information, keep it in mind and react to it when they navigate to particular points or when certain event happens. More recent articles like [12] proposed to apply road-map based methods to dynamic virtual workspace in relatively simple environments with moderate number of entities. Bottom-up control often employs a rule-based scheme and possibly provides each reactive agent with individual and cognitive behaviors. Brogan et al [15] modeled the path planning by using pedestrian performance statistics that were obtained during a suite of experiments. Complicated crowd simulation is usually a hybrid. For example, a designated path planning algorithm combined with spacing rules based upon social and psychological autonomy is used to generate the

plausible crowd animations in [4]. However, none of these combined systems offers the function of interactive control of crowd navigation.

Interactive control of human figures has been deeply investigated in game development. However, only a few approaches in literature were devoted to the interactive control of crowd. Crowdbrush [1], a most significant progress in crowd control interaction design in recent years, features some fancy graphical interface elements for controlling crowds. However, their control is still limited to each individual by specifying the property or response-rule of agents, which becomes a very tedious work when controlling large crowds as the one shown in Fig. 1. The Vi-Crowd system [3] controlled the crowd motion by establishing a comprehensive hierarchical model of the agent behaviors according to autonomy level. Nevertheless, the control in Vi-Crowd is not intuitive – complicated scripting skills are needed. Different from them, the navigation control in our approach can be completed by a few sketches. Anderson et al. [14] provided a method to generate constrained group animation; however, as their constraints setting is agent-based it is impractical to be applied to large crowd navigation.

Many recent behavioral control approaches focused on global navigation with road-map or graph based methods. The work presented in [2] employed the navigation graphs as road-map to achieve a real-time performance. In [8], Pettré extended his autonomous navigation method by considering interaction between pedestrians with a predictive approach and achieved satisfying results. Recently, adaptive road-maps which are especially useful in complex and dynamic environments have been proposed in [9]. Navigation Graph [10] constructed from Voronoi diagrams also fits in real-time path planning of multiple virtual characters. In order to handle collision avoidance in congested scenes with moving obstacles, Berg *et al* [11] approached this question based on *Velocity Obstacles* concept. Each agent is navigated independently in a decentralized manner while maintaining a cognitive map of the entire scene. Again, no interactive control as ours is given in these approaches.

Other crowd control algorithms are motivated by the concept of fields, for example, vector field, force field and potential field to animate the overall performance of crowds. The flow tiles model developed by Chenney [13] ensures the smooth flow of agents without resorting to explicit collision detection. However, realism has been sacrificed to some extent due to the lack of interaction between them. Treuille [5] integrated global navigation with a resolution dependent dynamic potential field, solving the motion of crowd. However, due to the drawback of resolution dependent representation, users should take more care to fine tune the parameters. Otherwise, oscillation will be observed in the simulation. In our implementation, we tried to smooth the potential field but we still found the simulation result not very good as most of the pedestrians keep turning around before making the next step. More discussions will be given in the result section. Once again, none of them addresses the problem about how to efficiently and intuitively generate fields for the interactive control of crowd navigation. The work presented here extends our previous conference publication by providing ease control of large crowds of virtual humans.

II. Vector field construction by RBF

A vector field can be considered as a position-to-vector mapping in the problem domain. Thus, the problem for constructing a vector field can be stated as follows.

Problem 1 Find a (functional) representation of the desired vector field Γ, which describes a mapping from each position in the navigating position $\mathbf{p}$ to a vector $\mathbf{v}$ and satisfies

$$\Gamma(\mathbf{p}_i) = \mathbf{v}_i, \qquad i = 1, ..., n,$$

where $\mathbf{p}_i$ and $\mathbf{v}_i$ are the position of anchors and the user specified vectors on anchors respectively. The anchors satisfy that $\mathbf{p}_i \neq \mathbf{p}_j$ $(\forall i \neq j)$.

This is actually a scatter data interpolation problem. Here we will employ RBF interpolation to obtain the function $\Gamma(...)$ from anchors. In the domain of crowd simulation, a desired vector field can be either 2D or 3D. A two-dimensional field is employed in land navigation (e.g., automobile driving) and a three-dimensional field is used in the space navigation (e.g., bird flying). 2D fields are used as examples in this paper for illustration. Two-dimensional version of the above problem is defined below.

Problem 2 Given a set of distinct 2D points

$$X = \{\mathbf{p}_i\}_{i=1}^n \subset \Re^2$$

and a set of vectors defined on these points as $\{(x_i, y_i)\}_{i=1}^n \subset \Re^2$, we need to find a pair of interpolation functions $(\gamma^x, \gamma^y) : (\Re^2 \mapsto \Re, \Re^2 \mapsto \Re)$ such that

$$\begin{cases} \gamma^x(\mathbf{p}_i) = x_i, \\ \gamma^y(\mathbf{p}_i) = y_i. \end{cases} \qquad i = 1, ..., n.$$

From [6], we know that the Radial Basis Functions (RBFs) are good candidates for solving this scatter data interpolation problem. We adopt the thin-plate spline $\phi(r) = r^2 \log(r)$ as the radial basis function ϕ. Replacing ϕ in the general form, we have

$$\gamma(\mathbf{x}) = P(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i (\|\mathbf{x} - \mathbf{x}_i\|_2)^2 \log(\|\mathbf{x} - \mathbf{x}_i\|_2) \qquad (1)$$

where $P(\mathbf{x}) = c_1 + (c_2, c_3)^T \mathbf{x}$. Substituting the positions of anchors and the vectors defined on them into Eq.(1), we obtain two linear equation systems (one for x-coordinate and another for y-coordinate). Solving these equation systems, the values of λ_i, c_1, c_2 and c_3 are calculated so that the interpolation functions (γ^x, γ^y) can be determined. This 2D solver can be easily extended to a 3D one by replacing the basis function with the biharmonic $(\phi(r) = r)$ or triharmonic $(\phi(r) = r^3)$ spline.

Under most conditions, the vector function computed by above isotropic RBF is smooth. However, relatively small distances between scatter points can lead the linear system of Eq.(1) to become ill-conditioned. Thus a metric-regularization approach called anisotropic RBF [7] is employed to ease this problem. In short, the ill-conditioned system is replaced by

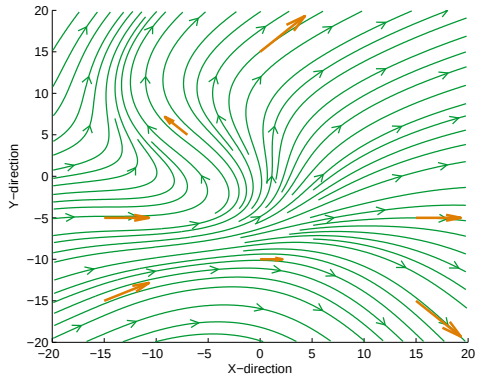


Fig. 4. A user defined vector field with arrows showing the anchor points.

a nearby, but better conditioned linear system, which is less sensitive to irregular points distribution.

This regularization approach involves building a local coordinate system by the product of a rotation matrix, and performs anisotropic scaling along the new cartesian axes on the interpolation data set. The basic idea is to change the Euclidean metric $\|\mathbf{x} - \mathbf{x}_i\|_2$ in Eq.(1) to a suitable metric T. Using T to measure the distance between data points, the distances between crowded points are increased reasonably in the new linear system so that the singularity is well prevented. In this way, the new basis functions are no longer radial with respect to the Euclidean norm, but considered radial regarding to the new metric T. After finding a non-singular 2×2 matrix M following [7], $T = M^T M$, then

$$\|\mathbf{x} - \mathbf{x}_i\|_T = \sqrt{(\mathbf{x} - \mathbf{x}_i)^T M^T M (\mathbf{x} - \mathbf{x}_i)} = \|M(\mathbf{x} - \mathbf{x}_i)\|_2$$

Thus, the new linear system is determined by

$$\gamma_i = P(M\mathbf{p}_i) + \sum_{j=1}^{n} \lambda_j (\|M(\mathbf{p}_i - \mathbf{p}_j)\|_2)^2 \log(\|M(\mathbf{p}_i - \mathbf{p}_j)\|_2) \tag{2}$$

In the cases of those already uniformly distributed data points, $T = I$ and the metric regularization has no effect on the solution.

In practice, more complicated manipulation of vector field stated above can be achieved by the following choices:

- using multiple vector fields to help the control of large groups of crowd navigation in complex scenes;
- changing some of the vector fields by adding, removing or editing any anchor point by mouse dragging and keyboard typing;
- extending the influence of 2D vector fields by incorporating time as the third component.

Different from the potential field in [5] which is defined on 2D grids with limited resolution, the vector field defined here has no limitation on resolution. Besides, we save a lot of memory space as we do not store every grid's value. The defined vector field will be regular as long as the distinct condition, $\mathbf{p}_i \neq \mathbf{p}_j$ $(\forall i \neq j)$, is strongly satisfied on anchor points. Fig. 4 gives an illustration about this continuous vector field. The arrows indicated anchor points defining the field.

III. Implementation Details

This section addresses the details about how to implement the interactive control of real-time crowd navigation. Each human in the simulation is processed as a particle with certain characteristics which are partly driven by the underground vector field derived in previous section. Four properties – mass m, radius r (used as the bounding cylinder for collision avoidance), velocity $\mathbf{v}$ and position $\mathbf{p}$ are stored for each particle during the simulation. The status variables are updated using explicit Euler's scheme.

The movement of each agent in the crowd simulation is composed of two components: the autonomous movement and the governed movement. For each agent's autonomous behavior, we adopt a behavior that consists of a conventional wandering and a weighted movement derived from all the anchor points, where the weights are inversely proportional to the distances between the agent's current position and the position of sample points – so that this movement somewhat reflects the nature of user specified crowd movement. The wandering behavior is realized by pointing a speed related random value at both x and y direction. At every time current, the 2D position of an agent is mapped to the user defined vector field, and the governed movement is calculated. The final movement of each agent is a synthesis of the autonomous movement and the governed movement. The ratio of synthesis can be adjusted interactively during the simulation to represent different personality of the crowd.

A more interesting nature provided by our method is that the movement of whole crowd can be easily adjusted on-the-fly in an interactive rate as we provide users with ease control of the guiding vector field during the simulation. More specifically, users can add more anchor points to the field, delete some of them, and modify the directions, values or positions of existing anchor points as long as the 2D positions of anchor points are different from each other. All of these functions can be easily completed. Adding more points can be done firstly by mouse clicking within the simulating scene, then dragging to determine their directions and magnitudes. For deleting operation, select one point with mouse, and use 'Delete' key on the keyboard. When changing the position of a certain point or the values attached to it, direction keys on the keyboard are used. As the deferred field value is smooth, the movement of pedestrians changes smoothly; no jerkiness is observed.

After editing, a new vector field is calculated to replace the old one. In contrast to grid-based representation that stores discrete field values, we only need to keep parameters λ_i, c_1, c_2 and c_3 in the memory as stated in **Problem 2**. At every simulation step, the governed portion of movement of an agent is calculated using the current position. To speed up the computation, GPGPU technique is employed. Firstly, we calculate the parameters and store them along with the position of anchor points in a texture for shader fetch. As everyone's governed movement is computed independently, it can be updated in parallel. We pack all pedestrians' positions in another rectangular texture, then use a fragment shader to calculate and output needed values to the third texture.

Fig. 5. A screenshot of the medieval city simulation.

Although reading from this third texture back to the main memory may be relatively slow, in our experiment results, this method is quite efficient when simulating hundreds of thousands of pedestrians.

The simulation of multiple groups of crowds can be simply implemented by specifying one vector field for each crowd. Therefore different crowd will have different governing vector field to guide its crowd behavior. This is very important for simulations as the ones shown in Figs. 1, 3 and 5.

The vector field derived by RBF-based interpolation only preserves the smoothness but does not prevent the self-intersection of flow lines. Furthermore, the autonomous movement has also been added onto each agent; there is no guarantee for leading to a collision-free crowd navigation. To prevent agents from running into each other, we enforce the minimum distances among them during the simulation. We implement collision and obstacle avoidance by using the function of OpenSteer. All the crowd members are stored in a database and organized into a tree. At every refreshment cycle, the position of each individual is changed, thus the database is updated accordingly. For every pedestrian to avoid obstacles, we first predict when the collision will happen with its current velocity and insert avoidance steering force when needed. While approaching pedestrians, position of the agent and a user specified radius are used to query the database; those neighbors within the defined sphere are returned. When the distance between two neighboring agents is less than a user defined threshold, a repelling force is employed to push them apart.

IV. Results

We have implemented our method as a C++ plug-in of OpenSteer. Simulations for Figs. 2, 3 and 6-9 ran on an Intel Core2 Duo 2.66GHZ CPU with Geforce 8600 GT graphic card and 2GB memory.

Fig. 2 gives an illustration about how to change the trajectory of crowd by user interaction. The crowd's main walking direction is guided by the anchor points interpolated vector

Algorithm 1 *SimulatorAdvancement*

```
for each time-step do
    if user modifies the samples on a vector field Γ then
        Reconstruct Γ by computing the function in Eq.(2);
    end if
    for each group of crowd do
        Compute the governed velocity vectors of agents;
        Compute the autonomous movement of agents;
        Update the locations of agents;
    end for
    Enforce the minimum distance between agents;
end for
```

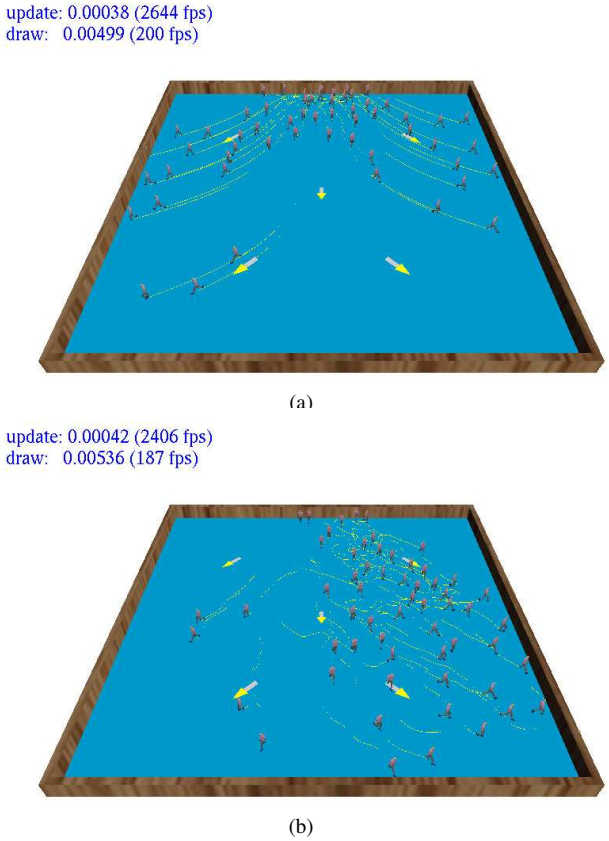


Fig. 6. Effect of the ratio between the movement governed by the vector field and the autonomous movement. The result without (a) vs. with (b) autonomous behavior.

field while still allow certain wandering activities. By our method, even if collision happens during the simulation, the whole crowd can still navigate following the user specified path eventually.

Fig. 6 shows the effect of different ratios between the governed behavior and the autonomous behavior on the trajectory of crowd simulation. It can be concluded that the one with more autonomous factors is closer to nature; however, we need to find a balance to avoid the movement of crowd to be far from what users designed on the vector field. Meanwhile, the speed of changing parameters and vector field needs to be controlled to avoid artifacts. Usually, we employ a linear interpolation between the one before modification and the

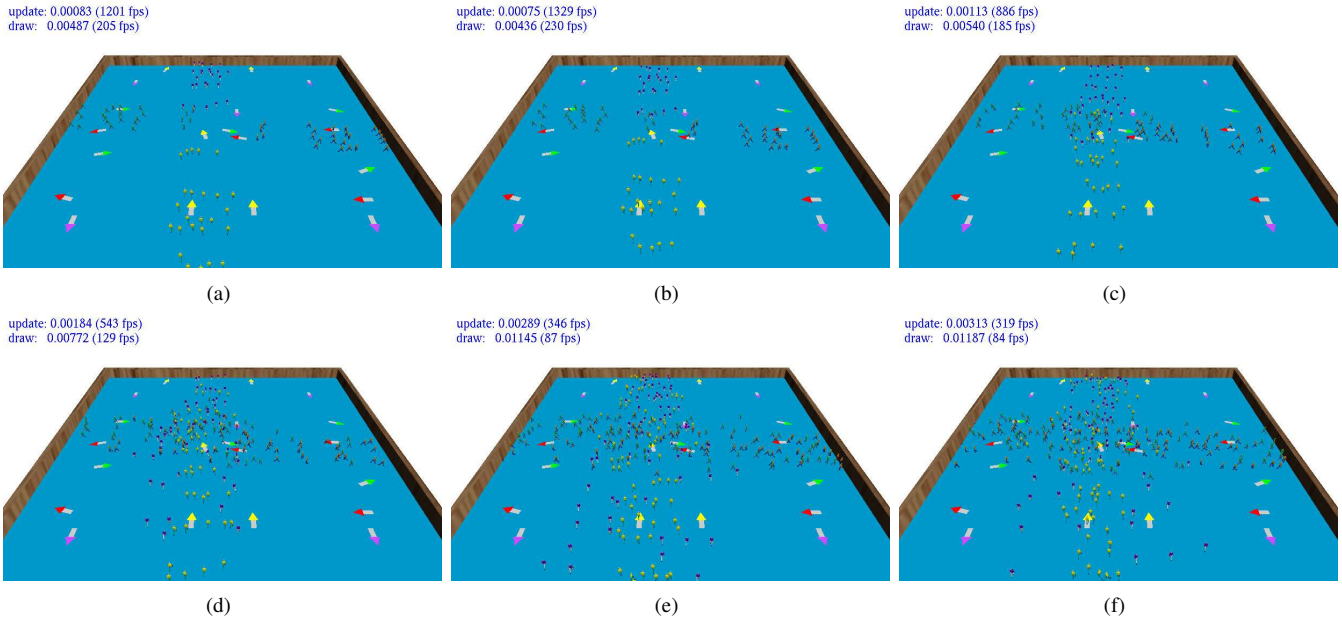


Fig. 7. A scene of four groups of pedestrians crossing each other. The four group of pedestrians and anchor points are rendered with similar colors.

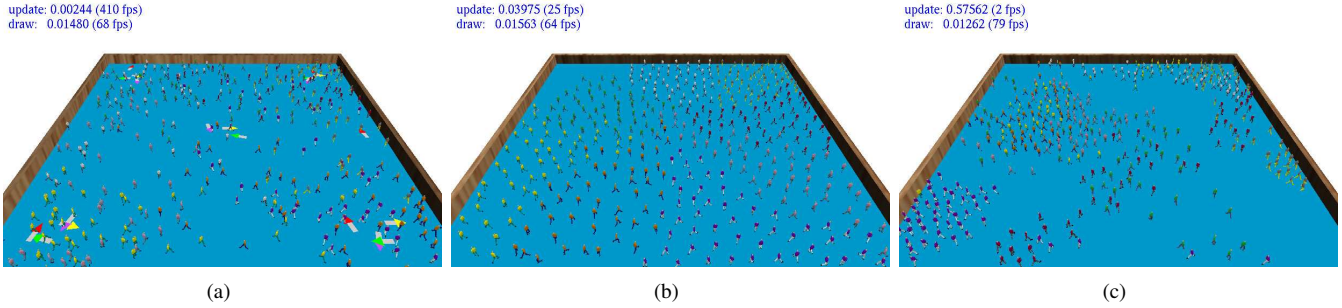


Fig. 8. Totally 400 pedestrians are driven by four vector fields with our method(a), vs. by $40 * 40$-grid potential field(b) and $100 * 100$-grid potential field (c) with method used by Treuille. As can be deduced from the pictures, Treuille's method is grid-resolution dependent.

modified one in a short period to improve the smoothness of change. Navigation of crowds in a medieval siege simulation (see the screenshots in Fig. 1 and Fig. 5) cannot be real without the autonomous behavior.

Often in complex virtual environment, different groups of crowds are simulated. The pedestrians walk along different paths until goals are finally reached. To deal with this situation, we employ different vector fields to govern their movement. As shown in Fig. 7, we simulate four groups of pedestrians entering the simulation workspace from four different directions, crossing from one side to another. In this example, we appoint five anchors points for each field. Similarly, we simulate four hundred pedestrians by our method (see Fig. 8(a)), the ratio of governed part is 50%, and compare it to the method proposed in [5]. The time performance of Treuille's method depends on how many grids are used to partition the virtual environment. Simulating with $40 * 40$ uniform grids (Fig. 8(b)), real-time performance is achieved. However, this level of space partition introduces artifacts in the final result as pedestrians may be uniformly spaced due to the coarse resolution of driving potential field. A better solution is to employ some high resolution workspaces. The simulation result of $100 * 100$ grids is illustrated in Fig. 8(c). As seen from Table I, the frame rate per second has dropped to about three. Moreover, it is difficult for novices like us to adjust the parameters; severe oscillations reduce the quality of result shown in the accompanied video demo by our implementation of [5].

TABLE I
UPDATING RATES OF TREUILLE'S METHOD

Grids Num	40*40	50*50	80*80	100*100	150*150
Performance	26 fps	18 fps	6 fps	3 fps	1 fps

* The number of agents tested is 500.

In order to prevent the rare unnatural vector field brought by putting anchor points too close to each other, our extension with Anisotropic RBF construction provides a better solution. In Fig. 9(a), around the middle area indicated by two crossing yellow vector, pedestrians pass through smoothly. In the Isotropic RBF integration result shown in Fig. 9(b), instead of walking downside, some of them walk towards the opposite direction or cycled around as indicated by orange cycle.

Experiments of update rates are shown in Tables II to IV. As shown in Table II, the calculation of vector field is quite fast. In Table III, the updating time of our CPU version is nearly linear proportional to the number of pedestrians simulated. This

TABLE II

COMPUTATION TIME OF ONE VECTOR FIELD

Anchors	50	100	150	200	300	500
Time	<1 ms	2 ms	5 ms	10 ms	28 ms	96 ms

TABLE III

UPDATING RATES OF OUR METHOD, CPU VERSION

	500 Agents	1000 Agents	1500 Agents
10 Anchors	145 fps	67 fps	35 fps
20 Anchors	131 fps	63 fps	33 fps
30 Anchors	124 fps	56 fps	30 fps
50 Anchors	105 fps	48 fps	28 fps
100 Anchors	82 fps	35 fps	22 fps

field governing method is appropriate for animating moderate number of crowds, while for large groups of crowds the GPU implementation is more efficient (see Table IV).

Our method is very easy to be implemented, and the simulation of crowds can be updated in an interactive rate because of its simplicity. The renderings of complex scenes as shown in Fig. 1 and Fig. 5 are performed as a post-process and can be time consuming. More specifically, we first export the simulation data, including crowd behaviors along with the chosen path into files. Next, the vivid soldier models and scenes generated by art designers are used to replace the ones stored in the previous step. We are planning to adopt some real-time rendering techniques in our framework. A good candidate is mixing LOD human model with image based impostor, geopostor or polypostor. When taking a close look at the scene, implementing 'instancing' on GPU would be a better choice. Futhermore, the current numerical solver of RBF in our system is implemented by the MATLAB API functions. In the next step, we will develop our new solver on GPU to achieve better time performance.

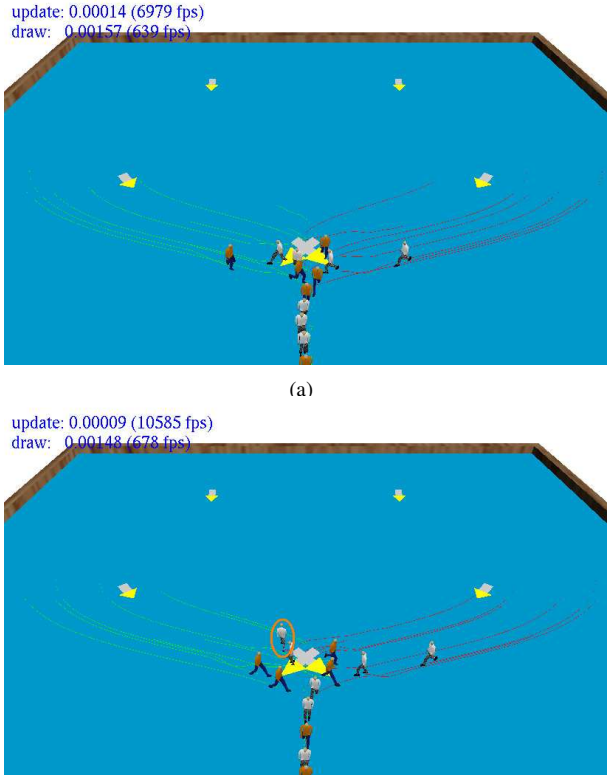


Fig. 9. The comparison of Anisotropic RBF (a) vs. Isotropic RBF method (b). With Anisotropic RBF method, pedestrians walk along user specified path. While in picture (b), several pedestrians are cycling around.

TABLE IV

UPDATING RATES OF OUR METHOD, GPU VERSION

	100,000 Agents	250,000 Agents	500,000 Agents
10 Anchors	60 fps	43 fps	20 fps
50 Anchors	60 fps	30 fps	16 fps
100 Anchors	60 fps	24 fps	13 fps

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (grant numbers 60573153, 60533080), the China 863 program (grant number 2006AA01Z314), and the China Key Technology R&D Program (grant number 2007BAH 11B03).

REFERENCES

[1] B. Ulicny and P.H. Ciechomski and D. Thalmann "Crowdbrush: Interactive Authoring of Real-time Crowd Scenes," *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer Animation,* pp. 243-252, 2004.

[2] J. Pettré and P.H. Ciechomski and J. Maïm and B. Yersin and J.P. Laumond and D. Thalmann, "Real-time Navigating Crowds: Scalable Simulation and Rendering," *Computer Animation and Virtual Worlds,* vol. 17, no. 3-4, pp. 445-455, 2006.

[3] S.R. Musse and D. Thalmann, "Hierarchical Model for Real Time Simulation of Virtual Human Crowds," *IEEE Transactions on Visualization and Computer Graphics,* vol. 7, no. 2, pp. 152-164, 2001.

[4] F. Lamarche and S. Donikian, "Crowd of Virtual Humans: a New Approach for Real Time Navigation in Complex and Structured Environments," *Computer Graphics Forum,* vol. 23, no. 3, pp. 509-518, 2004.

[5] A. Treuille and S. Cooper and Z. Popović, "Continuum Crowds," *ACM Transactions on Graphics,* vol. 25, no. 3, pp. 1160-1168, 2006.

[6] G. Turk and J.F. O'Brien "Shape Transformation Using Variational Implicit Functions," *Proceedings of SIGGRAPH 1999,* pp. 335-342, 1999.

[7] G. Casciola and L.B. Montefusco and S. Morigi, "The Regularizing Properties of Anisotropic Radial Basis Functions," *Applied Mathematics and Computation,* vol. 190, no. 2, pp. 1050-1062, 2007.

[8] J. Pettré and H. Grillon and D. Thalmann "Crowds of Moving Objects: Navigation Planning and Simulation," *2007 IEEE International Conference on Robotics and Automation,* 2007.

[9] A. Sud and R. Gayle and E. Anderson and S. Guy and M. Lin and D. Manocha "Real-time Navigation of Independent Agents Using Adaptive Roadmaps," *Proceedings of the ACM Symposium on Virtual Reality Software and Technology,* 2007.

[10] A. Sud and E. Anderson and S. Curtis and M. Lin and D. Manocha "Real-time Path Planning for Virtual Agents in Dynamic Environments s," *IEEE Virtual Reality Conference,* 2007.

[11] J. Berg and S. patil and J. Sewall and D. Manocha and M. Lin "Interactive Navigation of Individual Agents in Crowded Environments," *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games,* 2008.

[12] R. Gayle and A. Sud and M. Lin and D. Manocha "Reactive Deformation Roadmaps: Motion Planning of Multiple Robots in Dynamic Environments" *Proceeding of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2007.

[13] S. Chenney "Flow Tiles" *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation,* pp. 233-242, 2004.

[14] M. Anderson and E. McDaniel and S. Chenney "Constrained Animation of Flocks," *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Aymposium on Computer Animation,* pp. 286-297, 2003.

[15] D. C. Brogan and N. L. Johnson "Realistic Human Walking Paths," *Proceedings of the 16th International Conference on Computer Animation and Social Agents,* 2003.