Hindawi
International Journal of Digital Multimedia Broadcasting
Volume 2020, Article ID 3634054, 13 pages
https://doi.org/10.1155/2020/3634054

Research Article

Crowd Motion Editing Based on Mesh Deformation

Yong Zhang, Xinyu Zhang, Tao Zhang, and Baocai Yin

Beijing Key Laboratory of Multimedia and Intelligent Software Technology, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

Correspondence should be addressed to Yong Zhang; zhangyong2010@bjut.edu.cn

Received 17 January 2020; Accepted 16 November 2020; Published 8 December 2020

Academic Editor: Floriano De Rango

Computer simulation is a significant technology on making great scenes of crowd in the film industry. However, current animation making process of crowd motion requires large manual operations which are time-consuming and inconvenient. To solve the above problem, this paper presents an editing method on the basis of mesh deformation that can rapidly and intuitively edit crowd movement trajectories from the perspective of time and space. The method is applied to directly generate and adjust the crowd movement as well as avoid the crash between crowd and obstacles. As for collisions within the crowd that come along with path modification problem, a time-based solution is put forward to avoid this situation by retaining relative positions of individuals. Moreover, an experiment based on a real venue was performed and the result indicates that the proposed method can not only simplify the editing operations but also improve the efficiency of crowd motion editing.

1. Introduction

Computer simulation of crowd movement refers to setting up simulation models depending on crowd motion features in the reality, which means imitating the movement process of a bunch of people in the virtual environment. This technique is so useful that has been widely applied in the field of architecture, transportation safety, computer games, animation, and so on. In terms of film and television production, it can save time and massive money in making collective scenes that involved a large group of people, for instance, the battle scenes. Under the circumstance, it is meaningful to develop the production method of crowd animation, which would improve the productivity and make promotion to the animation industry.

Up to now, the making process of crowd animation relies on quantity of manual operations and it is difficult to edit every individual manually in large-scale crowd motion, especially when one's appearance time needs to be precise or the speed of a certain group of people has to change over time. In addition, the influence on crowd formation and speed brought by editing is hard to eliminate. When adjusting the pace or routes of a small part of people in the group, the speed of the whole crowd often changes in a mutation which looks strange for walking people. Besides, collision matters are likely to occur after editing crowd motion and improper solutions may cause extensive crash among virtual characters. According to previous studies, crowd behavior simulation under different circumstances has been widely researched, such as simulating the crowd rushing to exits in a closed room [1] and establishing an interactive simulation model to imitate human-like behaviors among people as well as their influence to the environment [2]. Furthermore, many researchers paid attention to the calculation efficiency and large-scale crowd visualization. Wong et al. improved a model by combining clustering and spring force to govern individual movement in the virtual environment [3]. With the broad application of neural network, motion rules can be learned from real data so that human walking can be simulated in higher speed [4]. In terms of crowd formation, Hughes [5] considered crowd as flow continuum and continuum models have been set up to capture key features of pedestrian flows [6]. Nevertheless, the editing of crowd animation during the simulation process has been rarely studied. Therefore, this paper is aimed at proposing a method to effectively edit the crowd motion.

Inspired by the cage-based editing method by Kwon et al. [7], we adopt concave hull to encircle the crowd and adjust its motion as a whole, which was able to intuitively edit the

crowd by dragging characters. Moreover, triangular mesh is used to establish topological relation among characters in the virtual population in order to maintain their relative positions. The modification process relies on the mesh deformation technique proposed by Liu et al. [8]. In addition, according to the editorial way mentioned by Kim et al. [9], this paper divide crowd movement into a spatial part and temporal part to adjust crowd formation and solve collision problems.

There are three main contributions of this research:

(i) The proposed method can effectively edit the crowd motion both collectively and individually

(ii) The method is able to retain the formation of the crowd and the relative positions between characters. Moreover, the collision problems between characters/obstacles are handled

(iii) The method is suitable for simulating activities that needed fixed relationships among participants, such as the march-in ceremony and military review

2. Related Work

Crowd simulation technology mainly includes modeling and simulation of virtual people's behaviors, group animation, and visualization of virtual crowd. Many researchers have explored the manners of virtual people. The seminal contribution to the simulation of crowd's behaviors should be the work of Reynolds [10] in 1987. They thought flock behaviors were the dense interactions of the relatively simple manners of the individuals. In addition, a distributed model was set up to generate the simple individual behaviors and complex flock behaviors evolved from individual behaviors. Reeves [11] proposed a particle system which is a collection of minute particles, modeling irregular fuzzy object. Musse and Thalmann [12] presented a Vi-Crowd model, including a hierarchy of crowds, groups, and individuals. Each level of autonomy had its own behavioral rule, and the complex crowd behaviors were simulated by synthetic effects among all hierarchies, which took sociological aspects into consideration. On the basis of the above model, psychological effects were considered for crowd movement by Pelechano et al. [13]. Moreover, Chenney [14] described a method for representing and designing speed fields by using flow tiles, which realized the minimum impact from the mixture of social and psychological influences.

In the field of group animation, which is a branch of crowd simulation, Ji et al. [15] carried out researches on the algorithm of group movement simulation and proposed a method for group calisthenics based on group events. Takahashi et al. [16] presented a spectral-based approach to control the temporal and spatial distribution of individuals by mapping the group formation into the rotation interpolation of Laplace matrix eigenbases. Kovar et al. [17] suggested a novel method based on motion graph which was automatically constructed, and walks can be built on the graph to generate motion. Interactive manipulation of multicharacter animation was initially proposed by Kwon et al. [18]. They used a graph structure to model the spatiotemporal group behaviors of the crowd and employed a mesh-editing algorithm to manipulate the animation interactively. Ulicny et al. [19] put forward the concept of crowdbrush that users could brush in three-dimensional scenes with a brush metaphor, and then the system would generate a crowd in the brushed place. Furthermore, different brush colors form distinct crowd behaviors. Allain et al. [20] applied constraints to control the crowd. Different from other methods, they divided the constraints into macroconstraints and microconstraints. Macroconstraints were used to constrain the macroscopic characteristics of the crowd, for example, the speed field; microconstraints were used to constrain the crowd speed, location, and so on.

3. Crowd Motion Editing Based on Mesh Deformation

Crowd scenes are commonly used in animation, but group motion is complicated and difficult to modify. For specific scenes that required large crowd, such us parade and mass choreography, if we could edit people's movement in a straightforward way, what you see is what you get, it would be a useful method to simulate crowd animation. Therefore, this paper proposed a method that can intuitively edit group motion through mesh deformation, separating the crowd movement depending on spatial and temporal attributes. The method can be divided into three portions which are the establishment of representation model, the spatial track deformation, and the temporal track deformation.

The overview of the editing model is shown in Figure 1 that is organized by the crowd motion editing method and trajectory deformation model. In Figure 1(a), the lower left corner is the generation process of the spatial track that contributes to form the representation model above. The representation model demonstrates the trajectory situation of the crowd, which can be edited according to spatial and temporal attributes, connecting two methods on the right side of this part. The editing method applied on spatial trajectory calculates new routes of group motion through the trajectory deformation model presented on Figure 1(b), and so does the model apply on the temporal track editing method. Therefore, the trajectory deformation model maintains relative positions of individuals as well as plans new paths. Then, it returns the editing results to the crowd motion editing method and rebuilds the representation model.

3.1. Representation Model of Temporal and Spatial Track. The path of a virtual character in the crowd simulation is a sequence of the motion clips arranged in a chronological order. In this thesis, the A$*$ algorithm [21] is applied to calculate crowd tracks so that the motion clips can be formed by taking sample points on the tracks.

As it is displayed in Figure 2, every spatial track is composed of a two-dimensional set of points obtained by projecting the motion clips onto the ground. More specifically, consider $P^1, P^2, P^3, \cdots, P^N$ as the trajectories of the first person to the Nth one accordingly. The ith trace P^i is on behalf of a piecewise curve, and the collection of any point $p_j^i \in R^2$ on

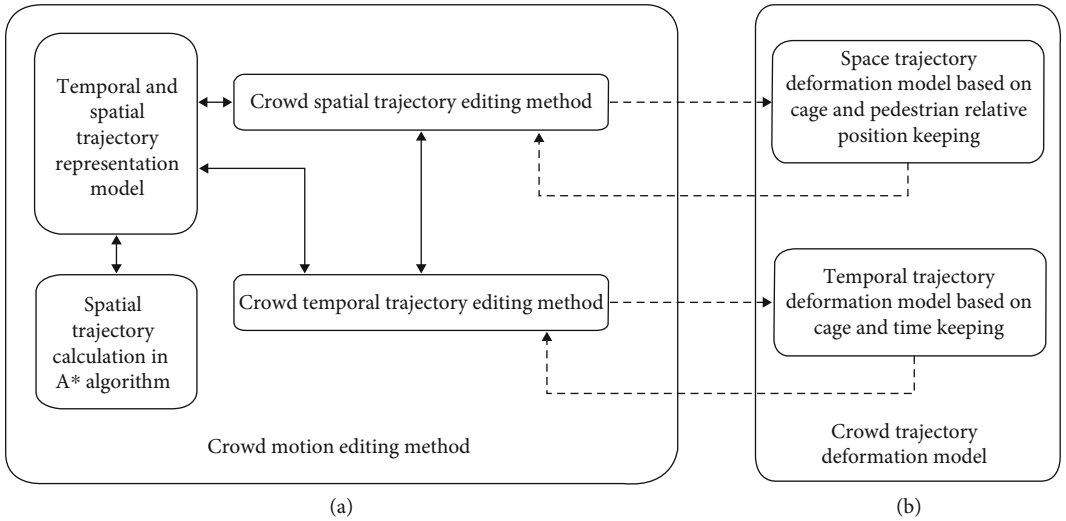

Figure 1: Architecture overview of the crowd motion editing model.

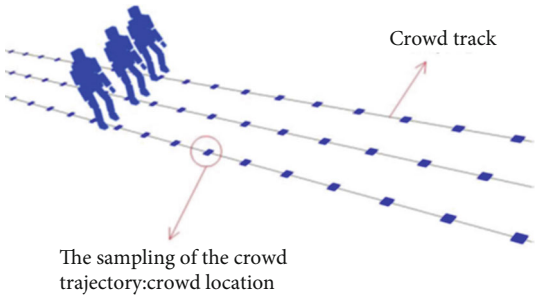

Figure 2: Crowd trajectories and sampling points.

this curve signifies a sequence of people's positions in time order. The direction of a certain point p_j^i on the track is represented by local coordinates: $(p_{j+1}^i - p_{j-1}^i)$ and $R(p_{j+1}^i - p_{j-1}^i)$, where R is a 2×2 rotation matrix of angle 90° [9].

In terms of the temporal attribute of the virtual population shown in Figure 3, regard $T^1, T^2, T^3, \cdots, T^N$ as the temporal tracks of the first N characters, respectively. Each track can be sampled and serialized into a set of one-dimensional points which show one-to-one correspondence to the points on the spatial track. Suppose t_j^i as a point on the time series, then $t_{j+1}^i - t_j^i$ is the interval from point p_j^i to p_{j+1}^i. Parameter t_j^i indicates the arrival time of the ith person in the jth place that is in accordance with point p_j^i. By adjusting the time sequence of the crowd, their speed displaying on a part of the space tracks can be changed. As a result, the crowd is able to reach a specific position in a given time.

3.2. Spatial Track Deformation Model Based on Concave Hull. The model is built on the basis of concave hull that encircles the selected crowd's space/time trajectories represented by Mean Value Coordinates (MVC) [22] with respect to the vertices in the hull (for details, refer to [22]). After generating the hull, Delaunay triangulation is performed to divide the area into numerous triangles and set up a triangular mesh. In consideration of keeping the relative positions among virtual population while editing, triangular meshes are used to establish the topological relationship. Therefore, the position of each individual is taken to form the vertices of the grid and mesh deformation with rigid characteristic [23] is applied to edit the crowd motion. After dragging a single vertex on the hull, the shape of the surrounding area is able to be modified through mesh deformation, changing the motion tracks of the crowd thereby.

When the user drags a point on the hull without fixing other points, as is shown in Figure 4, the deformation can be calculated by

$$\min_{v} \omega_{\mathrm{D}}^2 E_{\mathrm{D}} + \omega_{\mathrm{F}}^2 E_{\mathrm{F}} \quad \text{subject to } M\hat{v} = h, \tag{1}$$

where M is the constraint matrix to remain the dragging point unchanged during the deformation process. E_{D} represents the distortion energy which is used to measure the extent of the mesh deformation. Introduce as-rigid-as-possible deformation energy [23] as

$$E_{\mathrm{D}} = \sum_{t=1}^{T} \sum_{i=0}^{2} c_i^t \left\| \left(p\wedge_i^t - p\wedge_{i+1}^t\right) - R_t\left(p_i^t - p_{i+1}^t\right) \right\|^2, \tag{2}$$

where $\hat{p}_i^t$ and p_i^t denote the ith vertex of the tth triangle after and before the deformation in the triangular mesh, respectively. R_t represents the rotation estimation matrix from the deformed triangles to the triangles before deformation. E_{F} in the following function is aimed at adding the local effect of deformation:

$$E_{\mathrm{F}} = \|C(T\hat{v} - c)\|_2, \tag{3}$$

where C is the coefficient matrix. The closer the other points are to the dragging one, the smaller their coefficients are and vice versa. c is the position vector of the vertex before the deformation in the triangular mesh. $\hat{v}$ is the vertex of the

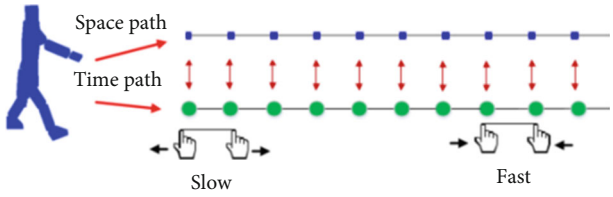

Figure 3: Spatial track and temporal track of each character.

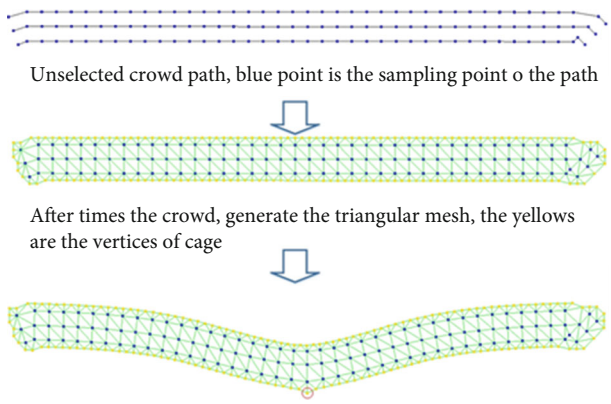

Figure 4: Dragging effect without fixed points.

concave hull after deformation. T is an $m * n$ matrix. m means the number of points on the tracks of the selected part with the addition of n—the vertexes of the concave hull. The first $m - n$ rows of T are the MVC [9] of the points on each track; the last n rows with n columns of the matrix form the $n * n$ unit matrix:

$$T = \begin{pmatrix} \lambda_1^1 & \lambda_2^1 & \cdots & \lambda_n^1 \\ \lambda_1^2 & \lambda_2^2 & \cdots & \lambda_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^k & \lambda_2^k & \cdots & \lambda_n^k \\ \hline 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \tag{4}$$

where k is the number of points on the tracks of the crowd.

As for the computation of coefficient matrix C, breadth-first search algorithm in the graph theory is used, namely, considering the mesh as a directed graph. The traversal begins from the dragging vertex and then goes through every layer of points around it. According to the layer combined with the distance from the start point, coefficients can be calculated. The formula of calculating the coefficient is shown below:

$$\text{coefficient}_i = \alpha d e^{c-k} + \beta c, \tag{5}$$

where coefficient_i is the parameter of the nonstarting node in the mesh; the coefficient of the starting node is 0. α and β are nonzero coefficients. c is the number of points in between the present vertex to the starting node. k is the threshold set in advance, and d is the distance covering from the current node to the starting one. It can be seen from Equation (5) that as c increases, d makes more influence on coefficient_i.

Based on the results of Equation (1), the closer the points are to the dragging vertex on the mesh, the greater the impact is; however, if the fixed point is far from the dragging vertex during the editing process, it may affect nothing to the points around the fixed point according to Equation (1). Nevertheless, often, it is the case that the mesh deformation is supposed to perform between the dragging point and the fixed point, as is presented in Figure 5. Therefore, Equation (6) is used to deal with the situation with the fixed point:

$$\begin{aligned} &\min_{v} \omega_{\mathrm{D}}^2 E_{\mathrm{D}} + \omega_{\mathrm{S}}^2 E_{\mathrm{S}} \\ &\text{subject to } M\hat{v} = h. \end{aligned} \tag{6}$$

In order to expand the influence of the dragging point, E_{S} is used to measure the distance variation among the vertexes of the concave hull, which is approximately presented by the change of the components on the X-axis and Y-axis:

$$E_{\mathrm{S}} = \|D\hat{v} - d\|_2, \tag{7}$$

where D is an $n * n$ matrix:

$$D = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & -1 \end{pmatrix} \tag{8}$$

After adjusting the tracks of the crowd by means of mesh deformation, the speed of some people in the crowd may become unnaturally fast and slow. This is because the distance between the sampling points on the character's track varies sharply; however, the time interval between them has not changed. Equation (9) is displayed to solve this problem:

$$\min_{\widehat{\Delta t'}} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\frac{\left\| p'_{i,j+1} - p'_{i,j} \right\|}{\Delta t'_{i,j}} - \frac{\left\| p_{i,j+1} - p_{i,j} \right\|}{\Delta t_{i,j}} \right)^2, \tag{9}$$

where $p_{i,j}$ and $p'_{i,j}$ represent the points on the tracks of the crowd before and after the deformation, respectively. Δt is the time interval between two points on the track before deformation.

The solving process of the target equation is demonstrated in Algorithm 1.

3.3. Temporal Track Deformation Model Based on Concave Hull. As users are able to drag points on the space track to manipulate crowd tracks, they can also operate the temporal track as well. When crowd tracks are discretely sampled,

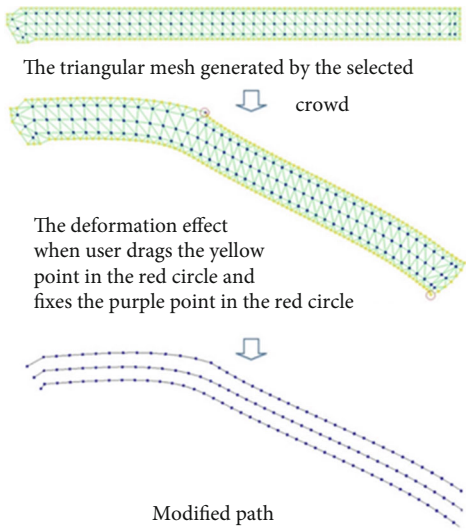

Figure 5: Dragging effect with a fixed point.

every sampling point on each track has a corresponding time node which indicates the moment that a character appears at this position. Therefore, it is not the only advantage for modifying the speed in a section of track after adjusting the time node; the time of occurrence at a certain place is also under control. When users drag the corresponding points on temporal tracks, the following equation is applied to adjust these points:

$$\min_{\hat{u}} \omega_{D'}^2 E_{D'} + \omega_{F'}^2 E_{F'} \quad \text{subject to } M\hat{u} = h \tag{10}$$

where $E_{D'}$ represents the degree of temporal deformation, which is similar to E_D. $E_{F'}$ controls the local variation traits of spatial deformation. $\hat{u}$ means the vertexes of the concave hull after deformation. The equation $M\hat{u} = h$ is used to ensure that the fixed points or the dragging points are not changed.

$$E_{D'} = \|D\hat{u} - d\|_2, \tag{11}$$

$$E_{F'} = \|F\hat{u} - p\|_2, \tag{12}$$

where matrix $D\hat{u}$ is the difference vector of two adjacent time nodes and d is the default time interval vector. $F\hat{u}$ and p denote the time node vector after and before the deformation separately.

4. Collision Avoidance Method Based on Space-Time Trajectory Deformation

During the process of crowd motion editing at present, the movement tracks are manually set. However, because of the huge amount of virtual characters involved with complicated routes, it is hard to notice the collisions among the crowd, let alone avoiding the crash by manual adjustment. Thus, collision detection algorithm and collision avoidance algorithm are indispensable, which make a great difference to the crowd animation quality. Therefore, we put up the measures of avoiding the collisions between characters/obstacles.

4.1. Collision Avoidance Method among People Based on Space Track Keeping. In view of collision avoidance among characters in the crowd, a method based on space track keeping is proposed—that is, to adjust the time tracks of related people to stagger the collision points. The strategy is shown in Figure 6.

4.1.1. Collision Detection. The collision problem of the ith character is detected by whether the minimum distances between the ith track and other tracks in the same section are less than the threshold.

d in Equation (13) displays the nearest distance between individual i and j:

$$d = \sqrt{\left(p_k^i + v_k^i t' - p_m^j - v_m^j t'\right)^2}. \tag{13}$$

In order to gain the relevant time information, take d^2's partial with respect to t' after getting d:

$$\frac{\partial d^2}{\partial t'} = 2\left(v_k^i - v_m^j\right)\left(p_k^i + v_k^i t' - p_m^j - v_m^j t'\right), \tag{14}$$

and then, make $\partial d^2/\partial t' = 0$ to solve t'. The result is presented in Equations (15) and (16):

$$t' = \frac{p_k^i - p_m^j}{v_k^i - v_m^j},\ v_k^i \neq v_m^j, \tag{15}$$

$$t' = 0,\ v_k^i = v_k^i, \tag{16}$$

$$t = \begin{cases} l_1, & t' + l_1 < l_1 \\ t', & l_1 < t' + l_1 < l_2, \\ l_2, & t' + l_1 > l_2, \end{cases} \tag{17}$$

where t is the time that two individuals get the shortest distance in different situations. In combination with t', d, and the speed mentioned in the above functions, the spatial position of collision can be presented, respectively, as $p_k^i + (t - t_k^i) * v_k^i$ and $p_m^j + (t - t_m^j) * v_m^j$.

The specific algorithm is shown in Algorithm 2.

4.1.2. Collision Avoidance. The space-time trajectory is the combination of the spatial track and the time track. It can be regarded as a polyline in a three-dimensional space, comprising the spatial position coordinates of the character and the timeline. When the collision occurred, the minimum distance of two segments of the broken lines in this three-dimensional space is less than the threshold. Therefore, enlarging the distance over the threshold is able to prevent the collision.

It is illustrated in Figure 7 that adjusting the time track can affect the character's three-dimensional trajectory to

Precondition: the user input the threshold.
(1) Initialize
(i) Calculate c_i^t in Equation (2), where c_i^t is the ratio of the perimeter of the tth triangle to the ith side of this triangle.
(ii) E1 =0. E2 =0. The number of iterations: iteration =0.
(iii) The vertex of the hull is equal to it before calculation: $\widehat{v} = v$.
(iv) Calculate EV_t of each triangle in the mesh: $EV_t = \begin{pmatrix} p_0^t - p_1^t \\ p_1^t - p_2^t \\ p_2^t - p_0^t \end{pmatrix}$.
(2) Terminal condition of the algorithm: abs (E1 – E2) <= threshold && iteration >=2
Calculation steps
(i) ***While*** abs (E1 – E2) > threshold || iteration <2:
(1) Calculate $\widehat{EV}_t$ according to $\widehat{v}$;
(2) Compute the rotation estimation matrix R_t in Equation (2) of each triangle in the mesh. Let $C_t = \begin{pmatrix} c_0^t & 0 & 0 \\ 0 & c_1^t & 0 \\ 0 & 0 & c_2^t \end{pmatrix}$,
$[s, v, d] = svd(EV\wedge_t{}' * C_t * EV_t)$, $R_t = d * s'$;
(3) According to the user's demand to solve the target Equation (1) or Equation (7)
(4) Save the value of previous target equation. For E1 = E2, the calculation results assigned to E1;
(5) Iteration = iteration + 1;
end while

Algorithm 1: Deformation solution process.

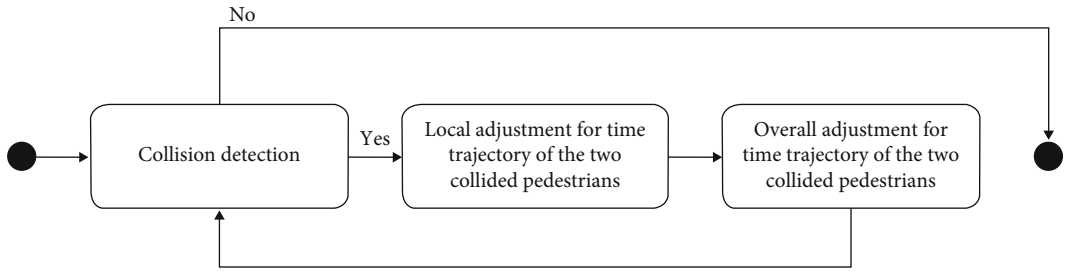

Figure 6: The collision avoidance strategy based on space track keeping.

(1) Judge whether the tracks of the ith and the jth character, which are (t_k^i, t_{k+1}^i) and (t_m^j, t_{m+1}^j) separately, have intersections.
(2) If there is no intersection, turn to Step 5. Or turn to Step 3.
(3) Gain the intersection (l_1, l_2) and use Equation (13) to find the nearest distance between two persons in (l_1, l_2).
(4) If the distance is less than the threshold, jump to Step 6.
(5) $k = k + 1$. If k is less than the total number of the track sections of the ith pedestrian, turn to Step 1, otherwise turn to Step 7.
(6) Record the characters who are going to collide and the corresponding position. Then jump to Step 8.
(7) $m = m + 1$. $k = 0$. If m is less than the total number of the track sections of the jth pedestrian, turn to Step 1.
(8) end

Algorithm 2: Collision detection.

change the minimum distance between two lines. The distance, therefore, becomes greater than the threshold with no need to change their spatial locations. This method is suitable for scenes with complex terrain and large crowd density, because it can avoid the collisions in the crowd without worrying about colliding obstacles in the environment.

In order to achieve the above purpose, the scaling factor S and translation coefficient T of time sequence are defined. When characters i and j have a crash, suppose that the crash moment is in the period (t_k^i, t_{k+1}^i) and (t_m^j, t_{m+1}^j) for two pedestrians separately. After the scaling translation with S and T, the time periods turn to $(t_k^i * a_i + b_i, t_{k+1}^i * a_i + b_i)$ and $(t_m^j * a_j + b_j, t_{m+1}^j * a_j + b_j)$, respectively. Therefore, the collision can be prevented if previous time periods are replaced with the changed one.

The scaling factor and the translation coefficient can be found according to Equation (19), while Equation (18)

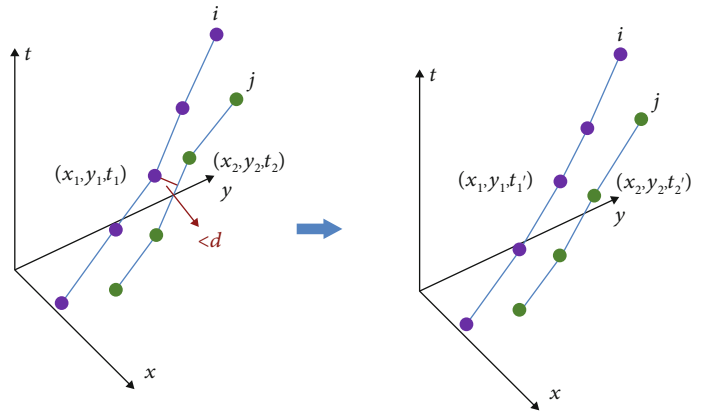

Figure 7: The time of collision avoidance.

measures the change of the time before and after the adjustment.

$$E_T = \left\|S_i T_k^i - t_k^i\right\|^2 + \left\|S_i T_{k+1}^i - t_k^i\right\|^2 + \left\|S_j T_m^j - t_m^j\right\|^2 + \left\|S_j T_{m+1}^j - t_{m+1}^j\right\|^2, \tag{18}$$

$$\begin{aligned} &f(S_i, S_j) = \min_{S_i, S_j} E_T dt = \text{distance}_{i,j}/\|v\| \\ &\text{subject to } g_1(S_i, S_j) = a_i \geq a, \\ &g_2(S_i, S_j) = a_j \geq a, \\ &g_3(S_i, S_j) = S_i T_k^i \geq 0, \\ &g_4(S_i, S_j) = S_i T_m^j \geq 0, g_5(S_i, S_j) = \left\|S_i T - S_j T\right\|^2 \geq \text{dt}^2 \quad dt = \text{distance}_{i,j}/\|v\|, \end{aligned} \tag{19}$$

where $S_i = \begin{bmatrix} a_i \\ b_i \end{bmatrix}^T$, $T_k^i = \begin{bmatrix} t_k^i \\ 1 \end{bmatrix}$, and $T = \begin{bmatrix} t - l_1 \\ 1 \end{bmatrix}$. dt is the time interval between character i and j, which is estimated as follows:

$$v = \min_{v_i, v_j} \left(\|v_i\|, \|v_j\|\right) dt = \text{distance}_{i,j}/\|v\|, \tag{20}$$

$$\text{distance}_{i,j} = D_{\text{threshold}} - \left\|p_k^i + \left(t - t_k^i\right) * v_k^i - p_m^j - \left(t - t_m^j\right) * v_m^j\right\|, \tag{21}$$

$$dt = \text{distance}_{i,j}/\|v\|, \tag{22}$$

where $D_{\text{threshold}}$ means the closest distance between two pedestrians. If the distance between two lines is less than $D_{\text{threshold}}$, the collision will happen.

Equation (19) is solved on the basis of the Kuhn-Tucker condition, that is, calculating the gradient of $f(S_i, S_j)$, $g_2(S_i, S_j)$, $g_3(S_i, S_j)$, $g_4(S_i, S_j)$, and $g_5(S_i, S_j)$, respectively, to find $\overline{S_i}$ and $\overline{S_j}$ satisfying Equation (23). The time nodes after adjustment are $\bar{t}_k^i = \overline{S_i} T_k^i$, $\bar{t}_{k+1}^i = \overline{S_i} T_{k+1}^i$, $\bar{t}_m^j = \overline{S_j} T_m^i$, $\bar{t}_{m+1}^j = \overline{S_j} T_{m+1}^j$.

$$\begin{aligned} &\nabla f\left(\overline{S_i}, \overline{S_j}\right) - \sum_{i=1}^{5} \omega_i \nabla g_i\left(\overline{S_i}, \overline{S_j}\right) = 0, \\ &\omega_i g_i\left(\overline{S_i}, \overline{S_j}\right) = 0, \quad i = 1, \cdots, 5, \\ &\omega_i \geq 0, \quad i = 1, \cdots, 5. \end{aligned} \tag{23}$$

After calculating the related time nodes, it is necessary to adjust the overall time series of the two characters in order to ensure that their speeds do not suddenly become very fast or slow. Considering that the change in speed is continuous, so the time people spent in per unit length is similar, expressed by E_{DT}:

$$E_{\text{DT}} = \sum_{i=1}^{m-1} \left(\frac{\Delta t_{i+1}}{l_{i+1}} - \frac{\Delta t_i}{l_i}\right)^2, \tag{24}$$

where l_i is the length of the ith segment on the pedestrian track and Δt_i is the time interval of the ith segment.

In view of the similarity it should have between the speed before and after, E_{V} is introduced to measure the degree of the speed change as shown in Equation (25). Nevertheless, the time node of the crowd is the only alteration during the whole process of collision avoidance, and therefore, the speed variation can be substituted by time information. Hence, E_{T} is defined to represent the modification on time as well as on speed for the replacement of E_{V}.

$$E_{\text{V}} = \sum_{i=0}^{m} \left(\bar{v}_i - v_i\right)^2, \tag{25}$$

$$E_{\text{T}} = \sum_{i=0}^{m} \left(\Delta \bar{t}_i - \Delta t_i\right)^2, \tag{26}$$

where $\Delta \bar{t}_i$ is the modified time interval. In the whole changing process, the adjustment of time node is hoped to be as

Comment: $d_{threshold}$ is the threshold that users input.
(1) Collision detection algorithm is used for any two virtual characters i and j in the crowd. If no collision happens, end the algorithm; otherwise record the number of the accident segment k and m as well as the closest distance $d_{\min}$;
(2) Solve Equation (20) to estimate the time interval dt;
(3) Use Equations (19) to adjust the kth section of character i and the mth section of character j;
(4) Use Equation (28) to adjust the whole time series of character i and j;
(5) Jump to Step 1.

Algorithm 3: Time trajectory deformation algorithm.

little as possible. As a result, E_{TP} is going to measure the degree of the change of time node as described in

$$E_{\mathrm{TP}} = \sum_{i=0}^{n} (\bar{t}_i - t_i)^2. \quad (27)$$

Associating with Equations (24), (26), and (27), the avoidance method can be demonstrated by an optimization model, as shown in

$$\min_{\bar{t}} \alpha^2 E_{\mathrm{DT}} + \beta^2 E_{\mathrm{T}} + \gamma^2 E_{\mathrm{TP}} \quad \text{subject to } MS * \bar{t} = mt, \quad (28)$$

where α, β, and γ are the weight coefficients of E_{DT}, E_{T}, and E_{TP}, respectively. The function of the matrix MS is to filter out the time nodes adjusted by Equation (18), while mt is a column vector consisting of the values of modified time nodes. The purpose of the constraint equation is to keep the time points that have been changed in the collision avoidance process away from changing again when the whole time track is later modified.

The specific algorithm of collision avoidance method is shown in Algorithm 3.

4.2. Obstacle Avoidance Method Based on Spatial Track Deformation Model. When the crowd pass through a narrow channel as displayed in Figure 8, it is easy to collide with obstacles. It is very tedious for animators to manually manipulate virtual characters avoiding obstacles, which consumes a lot of time. The reason of the crash is when the crowd turn at a narrow junction, the maximum width of the crowd is bigger than the width of the intersection, as shown in Figure 9.

Conditions for the crowd that smoothly pass obstacles can be expressed by the formula $D_{\text{crowd width}} > D_{\text{obstacle}}$, where $D_{\text{crowd width}}$ can be calculated by

$$D_{\text{crowd width}} = \max\left(\left\|p_{i,k} - p_{j,m}\right\| \bullet \sin\theta_{ij}\right), p_{i,k}, p_{j,m} \in G, \quad (29)$$

where $p_{i,k}$ and $p_{j,m}$ are the kth position of the ith character and the mth position of the jth character, respectively. G is a collection of characters' positions at the narrow junction. $\theta_{ij} = \theta_{ji}$ means the angle between the vector and the moving direction. In order to solve this collision problem, a measure integrating mesh deformation and cage-based method proposed by Kim et al. [7] is put forward to adjust the spatial trajectories of the crowd. This measure can restrict virtual characters to a narrow road without crossing obstacles by adjusting the contour of the mesh.

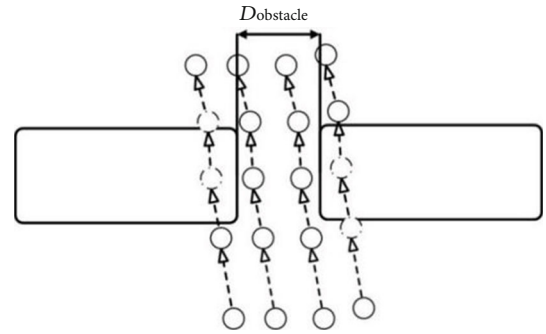

Figure 8: The crowd through the obstacles.

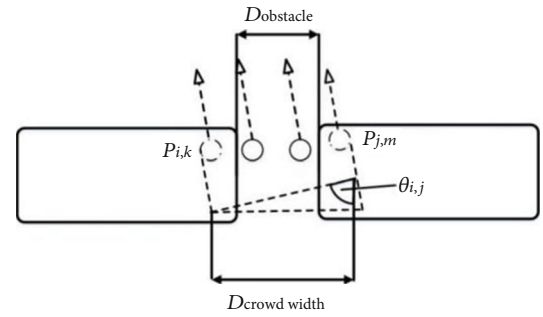

Figure 9: When the width of the population is greater than the width of the intersection.

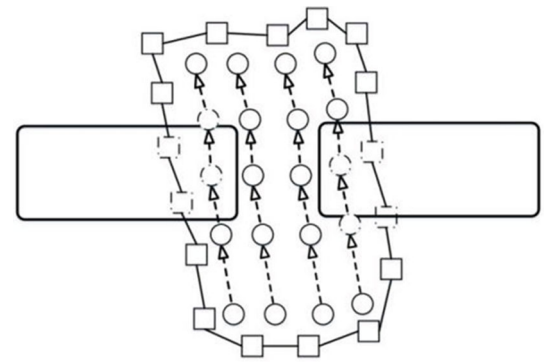

Figure 10: An enclosing shell of characters gets through obstacles.

As is shown in Figure 10, the contours with squares form an enclosing shell of the crowd. Since the shell is always on the periphery of the crowd, when the crowd is crossing the obstacles, the shell also gets through. In other words, if the

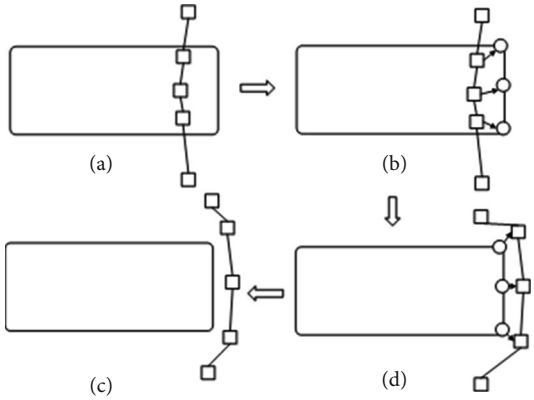

Figure 11: Adjustment of the bounding shell. (a) Intersections with the cage and an obstacle; (b) map the inside vertexes to the edge of the obstacle; (c) remove the inside points out of the obstacle depending on the mapping direction; (d) result of mesh deformation.

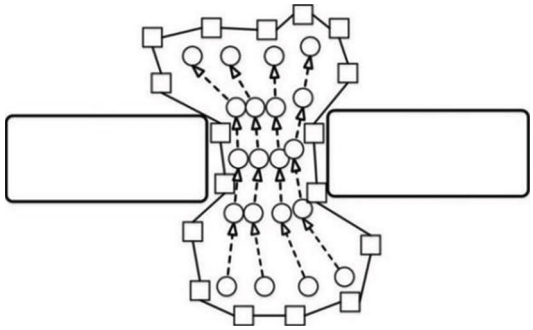

Figure 12: Adjustment of the vertices on the bounding shell.

Table 1: Hardware configuration of the experimental platform.

Configuration and parameters of the hardware
CPU Intel Core i3-2120 3.30 GHz
GPUNVIDIA GeForce 405
Motherboard Intel H61
RAM 4GB
Hard disk 250 GB SATA

shell has no intersection with barriers on the route, characters in the shell will not touch the barriers, too.

Thus, adjusting the squares on the contour line of the shell to move them out of the obstacles can effectively avoid the crash, so that the crowd will not get through the obstacles. The schematic diagram of adjusting the bounding shell is presented in Figure 11, where the squares are the vertices of the shell and the circular points are the mapping points of the vertices inside the obstacles.

As is presented in Figure 11, parts (a) to (c) describe the process of detecting whether the enclosing shell passes through the obstacles, calculating the mapping points, and obtaining movement directions, as well as moving the vertexes to the new positions. Part (d) represents the generation of mesh deformation by solving Equation (1) to get the end result. Figure 12 illustrates the overall situation when getting through the narrow road.

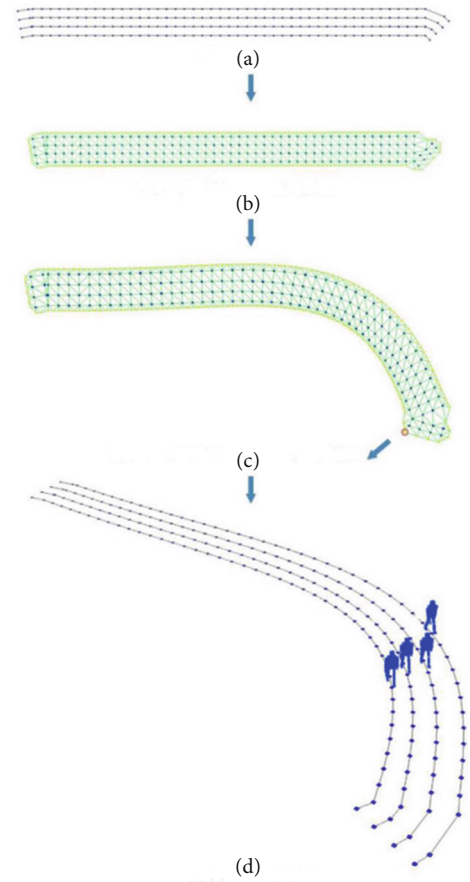

Figure 13: Effect of spatial track editing: (a) the original spatial tracks; (b) triangular meshes consist of spatial tracks and concave hull; (c) deformation effect after dragging the point in red circle along arrow direction; (d) editing result.

Mesh deformation is likely to change the trajectories of some characters in the crowd, which may lead to a sudden change in their speeds. Hence, Equation (9) is applied to adjust the speed, so that the movement becomes more realistic.

It is probably that the whole structure of original mesh will be influenced after deformation when a part of the mesh has been greatly narrowed down. As a result, the distance between each character is likely to become short and even brings track overlap, which easily causes the collision. Therefore, the collision avoidance method mentioned in Section 4.1 is used to solve this problem.

5. Experimental Results

To verify the performance of the proposed method, different experiments were designed to simulate the crowd motion according to distinct demands. This section demonstrates experiments on crowd motion editing and collision avoidance. Meanwhile, the practical use of the method has been tested by setting up an editing system for the badminton hall of Beijing University of Technology to imitate the event-holding situation. The experiments were programmed in C++, and OpenSceneGraph was used for rendering scenes.

Table 2: Crowd speed comparison before and after editing.

Character 1	Speed before editing						100					
	Speed after editing	89	88	87	87	87	88	89	90	91	91	91
	Speed after modification	100	100	100	99	100	99	100	99	100	99	99
Character 2	Speed before editing						100					
	Speed after editing	98	97	97	99	97	96	97	98	95	92	97
	Speed after modification	99	100	99	100	99	100	99	100	99	100	99

Furthermore, MATLAB was applied to compute the target equations and optimize the solutions. The hardware configuration of the experimental platform is shown in Table 1.

5.1. Crowd Motion Editing Results. According to Section 3, people's movement in the crowd is divided into spatial trajectory and temporal trajectory, while the position relationship between each character is recorded by the triangular mesh. Therefore, experiments in this part were conducted based on space and time attributes separately, which is in accordance with the description in Section 3.

5.1.1. Spatial Track Editing Results. In terms of editing the space tracks of the crowd, the experiments are focused on the mesh deformation capacity, the smooth transition problem of trajectory after editing, and the speed mutation. Figure 13 demonstrates the process of establishing, choosing, and editing the tracks as well as presents the final routes of characters. Equation (1) is used to calculate the editing result after mesh deformation, which is shown in part (c). It can be noticed that the edge of the crowd, that is, the concave hull, is in a smooth change during the editing.

In addition, the result reflects that partial deformation is caused because of Equation (1). There are two main factors deciding the deformation range brought by the above equation. The first one is the influence on the mesh given by editing. If the final position is far away from the original point, the whole grid will be under great influence, and therefore, the range of influence will be wide according to Equation (2). In order to make sure of the smoothness of the modification, it must be a gradual process that a large range of grid is involved. The second factor is the calculation of coefficient matrix C in Equation (3). The calculation result of Figure 13 is computed by Equation (4) that is set as $\alpha = 0.01$, $k = 300$, and $\beta = 0.02$. If C increase fast, the influence range will be small.

Table 2 demonstrates the comparison of two characters' speeds before and after editing. Because there are many segments on a trajectory, we only choose sections with large speed change. It is displayed in the table that after spatial editing on the crowd, their speeds have been influenced for the distance between each point is modified in some parts of the space tracks while the original time interval is retained. Equation (9) is applied to adjust the characters' temporal tracks to correct their speeds, and the result shows obvious effect after correction.

5.1.2. Temporal Track Editing Results. As for adjusting the time information, three characters in the crowd were chosen in this experiment. In Figure 14(a), we edit the temporal tracks of the bottom three characters to slow down their initial speeds and use Equation (9) to calculate the editing result. Parts (b) and (c) reflect the crowd formation before and after temporal track editing, respectively. It can be seen from the figure that the time modification causes variation on speed and further brings the formation change. In addition, it is shown in part (c) that the edited people remain in their relative positions during proceeding for their speeds are adjusted in an overall way.

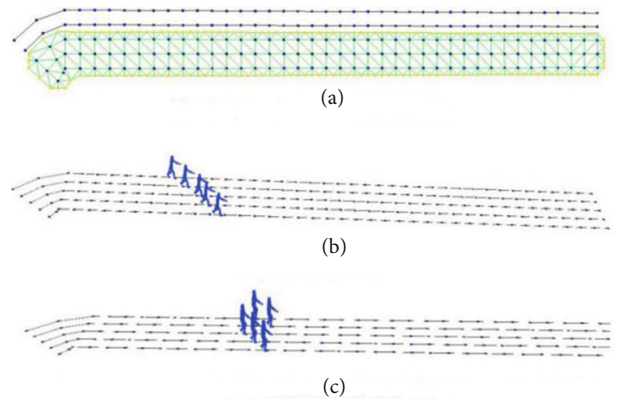

Figure 14: Effect of temporal track editing: (a) temporal track editing of three characters in the crowd: (b) the original formation of the crowd; (c) editing result.

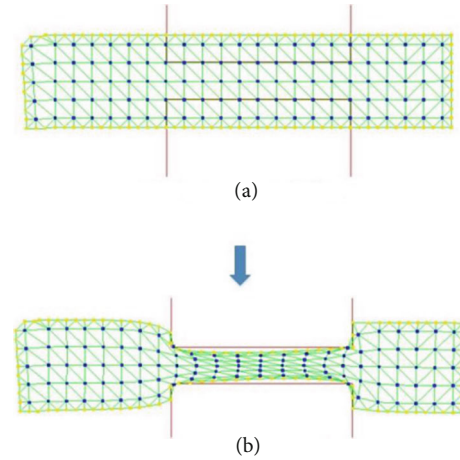

Figure 15: Collision avoidance between the crowd and obstacles: (a) the mesh generated from crowd getting through the obstacles; (b) mesh deformation application to avoid obstacles.

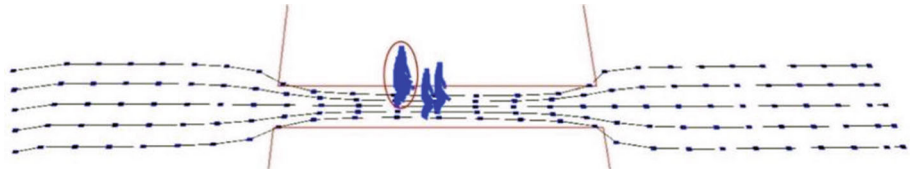

Figure 16: Collisions among people.

5.2. Collision Avoidance Results. In this section, we concentrate on collision detection as well as avoiding the crash that happens on characters in the virtual environment, which are the foundation of making crowd animation and has direct relation to the animation quality. In the experiment, a standard path was calculated by the A∗ algorithm from the starting point to the end. Then, other walks were generated by copying the standard path in accordance with the number of characters. To observe the effect of methods mentioned in Section 4, a part of the road was narrowed down to create an environment that is prone to accidents.

5.2.1. Collision Avoidance between People and Obstacles. Though the path avoided all of the obstacles during construction, it is likely for the crowd to crash on barriers when they pass a narrow alley. Hence, the mesh deformation method described in Section 4.2 is used to avoid characters getting through the obstacles.

As shown in Figure 15, part (a) is the original mesh of the crowd while part (b) presents the mesh after deformation under the constraint of the concave hull. It is illustrated in part (b) that the whole crowd with the concave hull (the outline consists of yellow points) exactly fills in or is narrower than the road. Because of the restriction from the hull, crowd tracks are bounded outside obstacles and achieve collision avoidance.

5.2.2. Collision Avoidance among People. On dealing with the crash between virtual people and the environment, collision within characters is easily involved. For the mesh is constructed from every spatial trajectory, deformation will cause track variation and also influences a character's speed, proceeding direction, and destination, resulting in crash in high possibility. The red circle in Figure 16 reflects the collision.

The measure mentioned in Section 4.1 is applied to this problem, and Figure 17 demonstrates the effect after modifying the spatiotemporal trajectories of characters.

For overall adjustment and optimization solution are needed for the crowd track after each modification, the convergence rate is low when a considerable number of people participate, and therefore, the calculation is time-consuming. In order to improve the computational efficiency, the crowd track was compressed before collision detection. After detecting and handling the collision problems, the results calculated on the basis of compressed tracks can be converted to that before compression. Equation (30) is used to transform the result after compression:

$$t_k^m = \frac{t_k^m}{\sum_{j=1}^{n} l_j^m} * \Delta T_i^m + t_{k-1}^m, \tag{30}$$

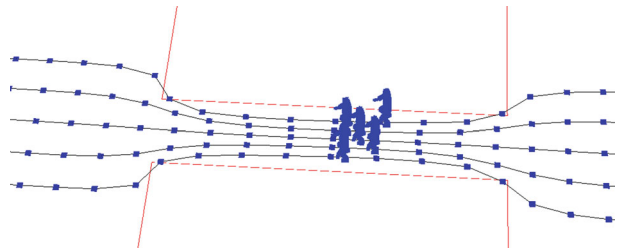

Figure 17: Collision avoidance among people.

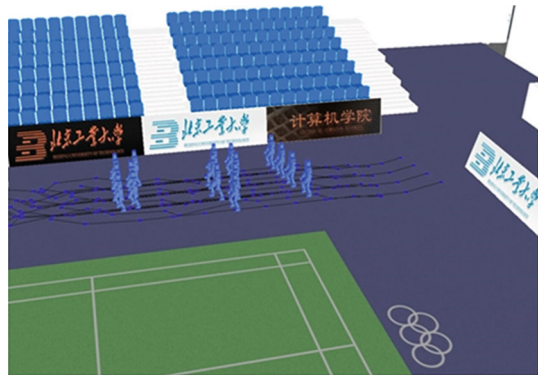

Figure 18: The editing result when characters pass the center of stadium.

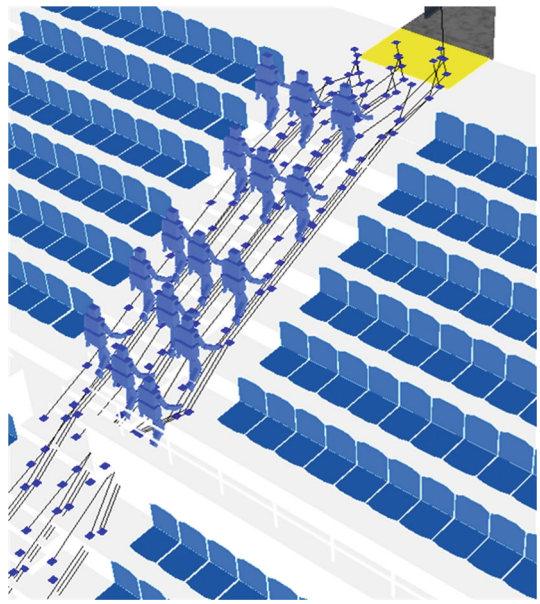

Figure 19: The editing result when characters move through the aisles.

where t_k^m and l_k^m present the kth time point and track segment of the mth character, respectively. Equation (30) informs that the result conversion is actually a process of time allocation in proportion.

5.3. An Application Case on Real Stadium. For practical use of the proposed method in Section 3 and Section 4, the badminton hall of Beijing University of Technology was mapped to a virtual scene to simulate crowd motion. The scene has nearly 7000 seats, and virtual characters are supposed to walk through the aisles naturally. As it can be seen in Figures 18 and 19, the model imitates the marching process of the crowd when they enter and leave the stadium, and the mesh deformation method with spatial and temporal track adjustment was applied to the movement editing.

6. Conclusions and Future Work

This paper put forward a crowd motion editing method that can be applied on crowd animation and simulation. The method is able to retain the relative positions of people, which is suitable for activities that needed fixed orientation relationships among participants, such as the march-in ceremony and military review. Meanwhile, this mesh deformation-based method directly edits the crowd by dragging the mouse, which is easier to operate and improves the simulation productivity.

Nevertheless, there is still room for promotion. Because of large amount of data involved in the editing process, future studies can focus on the enhancement of computational efficiency. In addition, the proposed collision avoidance method is time-based, and therefore, combining collision problems with spatial attributes is worth studying in the future.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request. Correspondence should be addressed to Yong Zhang: zhangyong2010@bjut.edu.cn.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62072015, U1811463, 61771058, and 61876012, in part by the Beijing Municipal Science and Technology Project under Grant Z171100004417023, and in part by Beijing Key Laboratory of Multimedia and Intelligent Software Technology, Beijing Artificial Intelligence Institute of Information Technology, Beijing University of Technology, Beijing, 100124, China.

References

[1] K. Yamashita and A. Umemura, "Lattice gas simulation of crowd behavior," in *Proceedings of 2003 International Symposium on Micromechatronics and Human Science*, pp. 343–348, Nagoya, Japan, Japan, October 2003.

[2] S. Kim, M. C. Lin, and D. Manocha, "Simulating crowd interactions in virtual environments (doctoral consortium)," in *2014 IEEE Virtual Reality*, pp. 135-136, Minneapolis, MN, USA, April 2014.

[3] K. Y. Wong, M. Thyvetil, A. Machaira, and C. Loscos, "System for simulating dynamic features of crowd behaviourvol. 115, p. 2005, ACM SIGGRAPH 2005 Posters.

[4] X. Wei, W. Lu, L. Zhu, and W. Xing, "Learning motion rules from real data: neural network for crowd simulation," *Neurocomputing*, vol. 310, pp. 125–134, 2018.

[5] R. L. Hughes, "THEFLOW OFHUMANCROWDS," *Annual Review of Fluid Mechanics*, vol. 35, no. 1, pp. 169–182, 2003.

[6] S. P. Hoogendoorn, F. L. M. van Wageningen-Kessels, W. Daamen, and D. C. Duives, "Continuum modelling of pedestrian flows: from microscopic principles to self-organised macroscopic phenomena," *Physica A: Statistical Mechanics and its Applications*, vol. 416, pp. 684–694, 2014.

[7] J. Kim, Y. Seol, T. Kwon, and J. Lee, "Interactive manipulation of large-scale crowd animation," *ACM Transaction on Graphics*, vol. 33, no. 4, pp. 1–10, 2014.

[8] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler, "A local/global approach to mesh parameterization," *Computer Graphics Forum*, vol. 27, no. 5, pp. 1495–1504, 2008.

[9] M. Kim, K. Hyun, J. Kim, and J. Lee, "Synchronized multi-character motion editing," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 558–560, 2009.

[10] C. W. Reynolds, "Flocks, herds and schools: a distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[11] W. T. Reeves, "Particle systems—a technique for modeling a class of fuzzy objects," *ACM Transaction on Graphics*, vol. 2, no. 2, pp. 91–108, 1983.

[12] 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.

[13] N. Pelechano, J. Allbeck, and N. Badler, "Controlling individual agents in high- density crowd simulation," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 99–108, San Diego, California, 2007.

[14] S. Chenney, "Flow tiles," in *ACM SIGGRAPH/Eurographics Proceedings of Symposium on Computer Animation*, pp. 233–242, Los Angeles, California, August 2004.

[15] Q. Ji, Z. Pan, L. Mei, S. Yang, and X. Li, "Virtual arrangement and rehearsal of group calisthenics," *Journal of Computer-Aided Design & Computer Graphics*, vol. 16, no. 9, pp. 1185–1190, 2004.

[16] S. Takahashi, K. Yoshida, T. Kwon, K. H. Lee, J. Lee, and S. Y. Shin, "Spectral-based group formation control," *Computer Graphics Forum*, vol. 28, no. 2, pp. 639–648, 2009.

[17] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," no. 51, pp. 1–10, 2008, ACM SIGGRAPH 2008 classes.

[18] T. Kwon, H. L. Kang, J. Lee, and S. Takahashi, "Group motion editing," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 15–19, 2008.

[19] B. Ulicny, P. D. Ciechomski, and D. Thalmann, "Crowdbrush: interactive authoring of real-time crowd scenes," in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 243–252, Los Angeles, California, August 2004.

[20] P. Allain, N. Courty, and T. Corpetti, "Optimal crowd editing," *Graphical Models*, vol. 76, no. 1, pp. 1–16, 2014.

[21] D. Sharma and S. K. Dubey, "Anytime A∗ algorithm-an extension to A∗ algorithm," *International Journal of Scientific & Engineering Research*, vol. 4, no. 1, pp. 1–4, 2013.

[22] M. S. Floater, "Mean value coordinates," *Computer Aided Geometric Design*, vol. 20, no. 1, pp. 19–27, 2003.

[23] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1134–1141, 2005.