Vis Comput
DOI 10.1007/s00371-013-0853-x

ORIGINAL ARTICLE

Inserting virtual pedestrians into pedestrian groups video with behavior consistency

Zhiguo Ren · Wenjing Gai · Fan Zhong · Julien Pettré · Qunsheng Peng

Abstract In this paper, we propose a novel approach to integrate virtual pedestrians into a scene of real pedestrian groups with behavior consistency, and this is achieved by dynamic path planning of virtual pedestrians. Rather than accounting for the local collision avoidance only, our approach is capable of finding an optimized path for each virtual pedestrian on his way based on the current global distribution of the real groups in the scene. The big challenge is due to the information of both position and velocity of real pedestrians in the video being unavailable; also the distribution of the groups in the scene may vary dynamically. We therefore need to detect and track real pedestrians on each frame of the video to acquire their distribution and motion information. We save this information by an efficient data structure, called environment grid. During the way of a virtual pedestrian, the respective agent frequently emits the detection rays through the environment cells to find the situation of the real pedestrians ahead of him and adjust the original path if necessary. Virtual pedestrians are merged into the video finally with the occlusion between virtual characters and the real pedestrians correctly presented. Experiment results on several scenarios demonstrate the effectiveness of the proposed approach.

Keywords Mixed reality · Agent-based simulation · Steering methods · Path planning

1 Introduction

Mixed reality has been widely studied in recent years. However, the seamless integration of virtual characters into a real scenes video remains as a difficult problem. Comparing to embedding static objects into pictures, three main challenges exist in on-line inserting virtual pedestrians into a video of pedestrian groups, i.e., acquiring the motion of real pedestrians, ensuring the behavior consistency between the virtual and real pedestrians and correctly presenting the occlusion between them without the 3D information of the real pedestrians.

Virtual pedestrians need to be located in the space consistent with the scene in the video. Video-based reconstruction approaches can be used to reconstruct the geometry of static objects, while the motion of real pedestrians should be acquired by methods of pedestrian detection and tracking. However, when scenes become crowded, it is difficult to efficiently obtain the precise motion information using these methods.

Behavior consistency, which means that the behavior of virtual pedestrians should be adapted to the environment in video, has a significant impact on the reality of integration. Although pedestrian simulation has been widely researched in fields such as virtual reality, computer animation, robotic etc., mixed reality has brought new challenges to it. As the motion of real pedestrians in the video is not available in advance, the virtual character should make predictions and

Electronic supplementary material The online version of this article (doi:10.1007/s00371-013-0853-x) contains supplementary material, which is available to authorized users.

Z. Ren (✉) · W. Gai · Q. Peng
State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
e-mail: naicer@126.com

F. Zhong
School of Computer Science and Technology, Shandong University, Jinan, China

J. Pettré
Mimetic team, INRIA Rennes, Rennes, France

Published online: 11 June 2013

take reactions properly. Moreover, in crowded scenes the accuracy of detecting and tracking individuals may decrease due to occlusion. The lack of 3D pedestrian group information may lead virtual pedestrians to squeeze through the crowd. No wonder, current pedestrian simulation methods which assume fixed and accurate motion inputs are not suitable to be applied to mixed reality.

When integrating virtual characters into a video, visibility needs to be computed when occlusion happens between the virtual and real pedestrians. Apart from setting the depth order of real and virtual pedestrians, the silhouette of the concerned real pedestrians need also be extracted from each frame of the video. However, the dynamic shape of real pedestrians and ubiquitous occlusion makes individual segmentation a big challenge. Current segmentation methods can hardly achieve a satisfactory tradeoff between the efficiency and precision in crowded scenes.

In this paper, we study techniques to integrate virtual characters into a video which contain pedestrian groups, with emphasis on the behavior consistency between the virtual and real pedestrians. Here, the group means some people walk together during a time period, leading to a locally crowded region in the scene, and members in the group may occlude each other in the view of the camera. Firstly, we propose an efficient dynamic path planning approach adapting to the motion of groups in real environment. Secondly, we present an approach to estimate the distribution of pedestrians despite the imprecise detection. Lastly, an on-line mixed reality system is implemented so that virtual characters can be inserted into real scenes after video processing at every frame. Experimental results show that virtual characters can be efficiently inserted into videos with pedestrian groups of medium-density.

2 Related work

Existing video analysis methods are used in this paper to detect, segment and track pedestrians in video. The motion information of real pedestrians extracted from the video is provided for virtual pedestrians to ensure that they can react appropriately in mixed reality simulation. Using silhouettes of the real pedestrians extracted through segmentation, the occlusion relationship between real and virtual pedestrians can be presented correctly.

Pedestrian detection methods based on classification [1–3] can robustly detect pedestrian location in complex environment. Wu et al. [1] adopted the human contour as cues to carry real-time detection on CPU. Sudowe et al. [2] reduced for the detection region by employing the constraints of the ground normal vector, height range of pedestrians etc.; to achieve accurate and efficient detection with GPU acceleration. Benenson et al. [3] increased the pedestrian detection rate to 100 fps by searching the feature space of different scale and using stixels. These efficient methods enable tracking-by-detection methods [4, 5] to be applied in real time video tracking. Pellegrini et al. [6] propose a model designed for walking people with short-term prediction in mind, which improving the tracking performance in busy scenarios. We adopt the method of Sudowe [2] to detect pedestrians, as well as the Extended Kalman Filter to track the bounding box of detected pedestrians.

Image segmentation methods [7–9] directly determine the location and silhouette of pedestrians from the image. The typical methods include background subtraction [7], methods based on graph theory [8] and methods based on level set [9]. Nevertheless, if there exist pedestrian groups in the scene, those methods can hardly distinguish individuals due to the complex occlusion relation among people in the group. In this paper, we take the group as a whole. Instead of segmenting individual pedestrians in the group, we ensure the correct occlusions between virtual pedestrians and the whole group by avoiding virtual pedestrians going through the group.

Pedestrian simulation, in general, can be divided into macroscopic simulation [10–13] and microscopic simulation [14–18]. Macroscopic simulation consider the crowd as a whole. This kind of methods can display macroscopic behavior of crowds; but the individuals motion result from the stream of people. They often result into unrealistic individual trajectories. Microscopic simulation, which often refers to agent-based simulation, however, focuses on the behavior of individuals allowing each virtual pedestrian to interact with environment independently. Compared to macroscopic simulation, microscopic simulation is more suitable for mixed reality environment applications.

Avoiding the collisions among dynamic obstacles, which particularly refers to pedestrians in this paper, is a core problem of microscopic simulation. Helbing et al. [14] adopted social force to represent interaction between pedestrians, the problem of collision avoidance is then converted to that of solving Newton dynamical equations. Reynolds [15] directed pedestrians to avoid collision by defining a series of rules. Velocity obstacle method [16, 17] is originated from robotics. An agent can find an appropriate velocity to avoid collision by solving an optimization problem in velocity field based on the relative position and velocity with respect to the nearby objects. Pettré et al. [18] improved the velocity-based method by analyzing the space–time relation of pedestrians and adjusting model parameter, leading to more natural simulation result. Kim et al. [19] introduce Bayesian RVO and learn the simulation parameters based on tracked data from video.

Existing microscopic simulation methods are not suitable to cope with the case of avoiding the potential collisions of the virtual pedestrians with the pedestrian groups in the

video. First, the methods solve for the optimized route of the current agent based on the information of the local region, while overlooking the potential collisions that may happen on its way. Agent getting closed to a group may fail to react timely to individuals in the group. Moreover, the framework of these methods is based on agent-to-agent interaction, which behaves well in sparse environment but may be less robust and efficient for crowd case. In particular, the extracted pedestrian information from a crowd video usually has noises, which may lead to oscillation problems, even deadlock if this information is adopted directly. In this paper, we will introduce a dynamic path planning method, allowing a virtual pedestrian to behavior properly against pedestrian groups in the video. The path of virtual pedestrians will be dynamically adjusted to avoid the crowded region based on the global distribution of real pedestrians in video.

In the field of mix reality, many approaches has been proposed to insert virtual object into real scenes [20]. Image-based methods put emphasis on the realistic illumination of the virtual object [21], while many other methods consider the physical interaction between virtual objects and the real scenes [22, 23]. Somasundaram et al. [24] proposed a system which can add virtual roles to video of indoor scenes. Since it needs to extract in advance the route of real pedestrians in video and then use the route information to avoid collision, the system is unable to run in real time. Zhang et al. [25] design an on-line mixed system to insert virtual pedestrians into a sparse environment with a velocity-based collision avoidance approach. Nevertheless, when the scene contain pedestrian groups, it will be very hard for the segmentation methods involved in video processing to accurately segment members in the group and the collision avoidance approach may not work as well. We will propose a mixed system which runs in real time and can deal with pedestrian groups in the video.

3 Overview

Similar to [25], our approach includes a pre-processing stage and an on-line integration stage. For the sake of simplicity, we use videos taken by a fixed camera, so the camera parameters need be computed only once. During pre-processing, we acquire the intrinsic parameters of camera through chessboard calibration, and get the homograph matrix by matching the coordinates of several marks in image and real ground. Then extrinsic parameters and the matrix transforming object from image coordinates to the unified world coordinates can be obtained. Also at this stage, certain frame background video will be collected for on-line segmentation. Moreover, the illumination will be estimated to guarantee the light consistency between the virtual pedestrians and the environment after rendering.

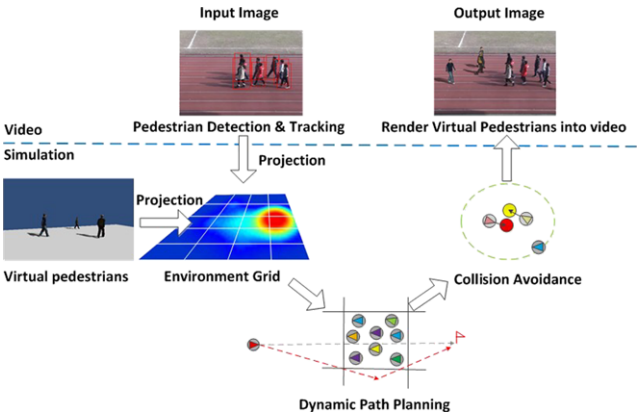

Fig. 1 Overview of on-line processing

The on-line integration stage as shown in Fig. 1, performs tasks of real pedestrian information acquisition, virtual pedestrian simulation as well as video integration. For each frame of video, the position and velocity of the real pedestrian in the world coordinate can be acquired from the video adopting the pedestrian detection and tracking methods. We divide the two dimensional space of the environment into uniform cells. During simulation, we first project the distribution information of the real and virtual pedestrians onto the environment cells, which we will describe in Sect. 4. Each cell records the information of the current pedestrians, including their density and the average velocity, within its local area. Next, based on density and velocity distribution, virtual pedestrians adjust their path using dynamic path planning algorithm, which will be presented in Sect. 5. Based on the adjusted path, their new velocity and position will be obtained by implementing a collision avoidance algorithm. Finally, we insert the location-updated virtual pedestrians into the real video.

Note that the optimized dynamic path planning of each virtual pedestrian is based on the information of the dynamic distribution of pedestrians on their way. The motion information of pedestrians in the environment can be divided into two types, i.e., microscopic and macroscopic, which is similar to [10]. Microscopic information includes the individual velocity $\mathbf{v}^{\text{cur}}$ and the position $\mathbf{p}$ of each pedestrian, while macroscopic information includes the crowd density ρ and the average flow velocity $\overline{\mathbf{v}}$, which reflects the congestion level of the region and the moving trends of the groups.

For description convenience, we use 'agent' to indicate virtual and real pedestrian, and an agent can be represented as a disc with certain attributes (such as velocity, position, etc.). During simulation, we store the distribution of agents on a sparse uniform grid, called environment grid. The grid points are located either at the center of the cell or at the midpoint of the cell edge, named central grid point or boundary grid point, respectively. Each grid point stores the information of a local region, including ρ and $\overline{\mathbf{v}}$. We use the splat-

ting method to project the discrete agent information on to the grid. The method is often used to record the distribution of pedestrians by macroscopic model.

By accumulating the values carried by nearby agents with weight, we can obtain the values of density ρ and velocity $\overline{\mathbf{v}}$ on the grid:

$$\begin{cases} \rho = \sum_i \varepsilon_i w(\mathbf{p}_i - \mathbf{p})\rho_i \\ \overline{v} = \dfrac{\sum_i \varepsilon_i w(\mathbf{p}_i - \mathbf{p}) v_i^{\text{cur}}}{\sum_i \varepsilon_i w(\mathbf{p}_i - \mathbf{p})} \end{cases} \tag{1}$$

where $w(\mathbf{p}_i - \mathbf{p})$ is used to ensure that the influence of an agent's $\mathbf{p}_i$ on grid point $\mathbf{p}$ decreases with the increasing distance, and we notated the influence radius of the weight as R^{weigth}. In addition, in Eq. (1), ε is used to adjust the individual impact of each agent on grid points. Accounting for the movement of the agent, we adopt the position of agent $\mathbf{p}_i$ after a short time τ for projection.

Each cell maintains a dynamic index table to record the indices of all agents within that cell. Agent can quickly find other agents nearby by query the index table, and get their $\mathbf{v}^{\text{cur}}$ and $\mathbf{p}$. During each time step in the simulation, the macroscopic information at each grid point is updated after updating the position and velocity of all agents. In order to improve efficiency, only those cells that contain agents need to get updated.

4 Video processing

4.1 Pedestrian detection and tracking

We adopt GroundHOG [2] to locate the individual 2D image of each real pedestrian, the 3D bounding box can then be estimated making use of the geometric constraints such as ground plane and the human body. Since the intrinsic and extrinsic camera parameters are already known after the preprocess, the mapping between image space and object space can be obtained performed easily. To track the 3D path of each real pedestrian, we adopt the Extended Kalman Filter (EKF) [26].

When virtual pedestrians are integrated into a scene video with a moving group, they may either occlude or be occluded by the real pedestrians in the scene. To generate a correctly composite image, the visibility of each virtual pedestrian should be computed. For this purpose, the depth order and an accurate alpha matte of each real pedestrian are required. The depth order can be estimated based on the footprint of each pedestrian. To get the alpha matte, we first use background subtraction to get a rough mask, and then apply morphological operations to remove noise and small holes.

The pedestrians in groups in the video are difficult to be detected robustly. Due to the occlusion, the detector may miss some objects. In Fig. 2, only the pedestrians represented by the yellow points close to the side of camera are detected. If only these real pedestrians are accounted, the virtual pedestrian may fail to make the right choice and squeeze into the crowd directly. In addition, his behavior of collision avoidance may also become instable due to the varied number of real pedestrians detected at adjacent frames. Even worse, the virtual pedestrians may penetrate with undetected real pedestrians. In this paper, we propose a new solution, which can deal with the above problem effectively for groups of medium-density.

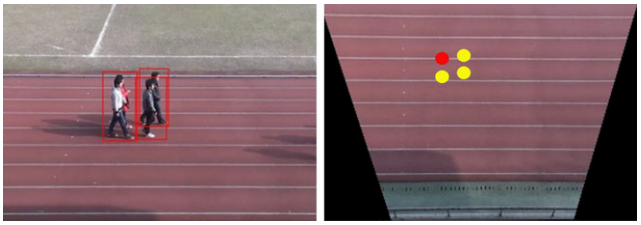

Fig. 2 Pedestrian detection

4.2 Computing the crowd density

To facilitate the path planning of a virtual pedestrian, the crowd density at each grid point along his way should be provided. To deal with the problem of unstable estimation of crowd density, we propose a two-pass solution. In the first pass, a density map is computed according to the number of all detected real pedestrians. In the second pass, the crowd density at some grid points are adjusted to cover the density loss due to the miss detected pedestrians. Our method is based on the following assumption: due to occlusion, the more crowded a region is, the more difficult to detect every pedestrian. Therefore, the second pass will focus on the regions which are found nearly crowded during the first pass. Specifically, we first sort the grid points according to their density value, and then select the most front N grid points in the list satisfying $\rho < \rho^{\text{thresh}}$ as the candidate grid points for further process. The density value at these grid points is likely to be underestimated due to the detection error. We then introduce Proxy pedestrian into the local region of each of these grid points to make up its density, which is shown in Fig. 3. Each Proxy pedestrian has its own density, velocity, and weighting function, which serve for density computation only. Note that the Proxy pedestrians do not actually participate into the crowd movement, they pose no impact on the process of collision avoidance. Assuming that each Proxy pedestrian has the same density and weighting function with the real pedestrians in the respective region, we will then determine its position, velocity and the influence radius of its weighting function.

To estimate the crowd density, each grid point records the real pedestrians distributed within its local region at every

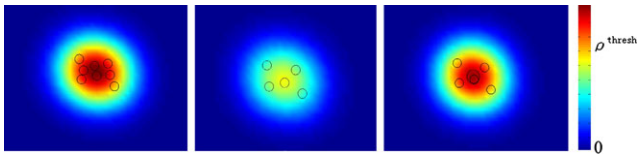

Fig. 3 Density map of real pedestrians. The *left image* is the density map of all real pedestrians, each represented by a *black circle*. The *central image* is the density map of the detected pedestrians. The *right image* is the density map after introducing a Proxy pedestrian

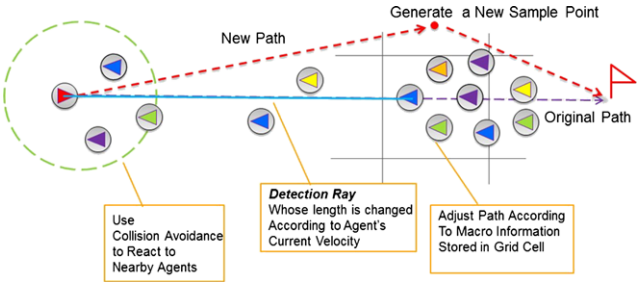

Fig. 4 Dynamic path adjustment

frame. The average position and velocity of these pedestrians is adopted as the position and velocity of the corresponding Proxy pedestrian. The Proxy pedestrians therefore automatically located near the center of the crowd area. According to the distribution of real pedestrians within the local region, we can determine the radius of the weighting function of the Proxy pedestrian. Denoted by $d^{\text{pedestrian}}$ the max distance between the Proxy pedestrian and real pedestrians, and d^{grid} the distance between the Proxy pedestrian and the grid point, then the radius of the weighting function can be determined by $R^{\text{weight}} = \max(d^{\text{pedestrian}}, d^{\text{grid}})$.

Based on the continuity of pedestrian movements, we assume that the number of pedestrians in a region stays stable within a short time period. For each grid point, we record the maximum number of detected pedestrians within the local region of that grid in a time period, and take it as the actual number of pedestrians in this period. In implementation, we maintain a queue at each grid point whose capacity is k, and then for each frame, the number of detected pedestrians within the respective local region is recorded and pushed into the queue, meantime the last record at the rear of the queue is popped. The entry with the maximum number, say $N^{\max}$, in the queue is regarded as the real number of pedestrians passing through the local region of that grid in the latest k frames.

When the number of detected pedestrians at a grid point is likely being underestimated, the introduction of a Proxy pedestrian will increase its density to a normal value. This is achieved by adjusting in Eq. (1). Let the number of pedestrians detected at the current frame be N^{cur}, we define:

$$\varepsilon = 1 + \lambda\left(N^{\max} - N^{\text{cur}}\right) \tag{2}$$

where λ is the parameter specified by user. On the other hand, the contribution of the Proxy pedestrian to the density value at that grid point will decrease following the increase of the radius of the weighting function. Let R^{default} be the default radius of weighting function for all real pedestrians, we rewrite Eq. (2) as

$$\varepsilon = 1 + \frac{\lambda R^{\text{weight}}}{R^{\text{default}}}\left(N^{\max} - N^{\text{cur}}\right) \tag{3}$$

5 Dynamic path planning

We extend the traditional agent-based simulation framework by developing a dynamic path planning module between global path planning and local collision avoidance module. This additional module locally adjusts the initial path of each virtual pedestrian according to the dynamic distribution of pedestrians in the scene, without recalculating the global path.

5.1 Notation

We use the following notations to represent the motion of pedestrians:

- $\mathbf{p}$—the current position
- $\mathbf{p}^{\text{goal}}$—the goal position
- $\mathbf{v}^{\text{cur}}$—the current velocity
- $\mathbf{v}^{\text{pref}}$—preferred velocity
- $v^{\max}$—maximum velocity
- r—radius of personal area

Here, $\mathbf{v}^{\text{pref}}$ refers to the pedestrian preferred velocity if there are no obstacles in the environment. Personal area refers to the minimum personal space of a pedestrian kept away from other pedestrians. In this paper, we use circle to approximate the personal area, and represent the personal area with radius r. During the simulation, each pedestrian can be regarded as a disc with the above properties. $\mathbf{v}^{\text{pref}}$, $v^{\max}$ and r are relevant to physiological and psychological factors of pedestrian, and can be generated by an additional physiological and psychological module, or specified by the user. In this paper, we assume that $\mathbf{v}^{\text{cur}}$, p and r for each pedestrian can be perceived by others, while the residual attributes are only visible to the pedestrians himself.

5.2 Path adjustment

As shown in Fig. 4, the agent makes decision taking into account macroscopic and microscopic information of the crowd distribution. Specifically, according to the information of crowd distributions to red in the grid on his way, each agent adjusts $\mathbf{v}^{\text{pref}}$ as well as the initial path at an early time.

Table 1 The type of grid points

Location of grid point	Type	Code
Central point (C)	Comfort point	00
	Congestion point	11
Boundary point (N, E, S, W)	Comfort point	00
	Congestion point	01
		10

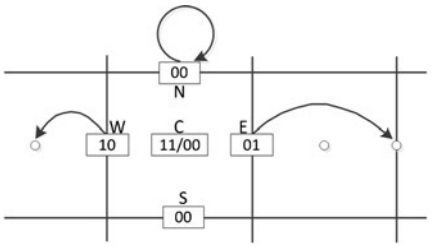

Fig. 5 The binary code of grid points

During the way, the agent determines dynamically his actual velocities $\mathbf{v}^{\text{cur}}$ based on $\mathbf{v}^{\text{pref}}$ and motions of his neighboring agents in the local region. Finally, the position of each agent is updated by adding $\mathbf{v}^{\text{cur}}$ into current position.

The ability of getting macroscopic information is essential for agents to interact with crowd. Each agent acquires the knowledge of crowd distribution by emitting a detection ray along the path to his destination. The range of ray l^{range} is determined according to the current velocity of each agent and the distance to his goal position:

$$l^{\text{range}} = \min\left(\frac{|\mathbf{v}^{\text{cur}}|}{v^{\max} l^{\text{default}}}, d\left(\mathbf{p}^{\text{goal}} - \mathbf{p}\right)\right) \tag{4}$$

Here, l^{default} is the parameter concerning the vision ability of each agent. It can be defined in the external module or simply specified by user. We use DDA algorithm to find the grid cells intersecting with the detection ray, and record the indices of the nearby grid points along the ray in a list which will be provided to the corresponding agent for reference. The initial path of each agent can then be adjusted according to the information stored in these grid points.

Grid points can be classified into two types: comfort points whose density values are smaller than ρ^{thresh}, and congestion points whose density values are larger than ρ^{thresh}. We use binary code to represent the type of a grid point as shown in Table 1. Figure 5 illustrates the way how to search for comfort point through the binary code stored at a boundary grid point along the specified direction. If the binary code of the grid point is 10, the central grid point of its adjacent cell will be the comfort point. On the other hand, the code 01 indicates a congestion point at the central grid of its adjacent cell and the search for comfort point should continue along other directions.

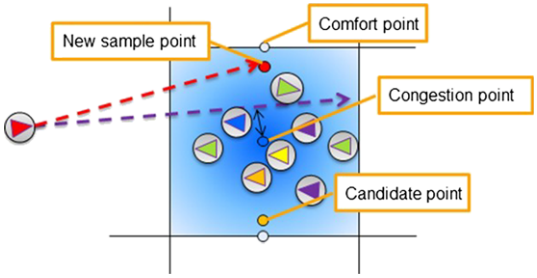

Fig. 6 Generating new sample point

In the implementation, the initial path of an agent is defined as a set of pre-computed sample points. During the way, this path may be adjusted according to the motion of the pedestrians ahead of the agent, both the density of the crowd and its moving tendency are considered. Once arriving at a congestion point, the agent makes decisions according to his current velocity and the average velocity of the crowd close to the agent. If the crowd moves along the same direction and its average velocity is faster than that of the agent, i.e., $\bar{\mathbf{v}} \cdot \mathbf{v}^{\text{pref}} > |\mathbf{v}^{\text{pref}}|$, and $\frac{\bar{\mathbf{v}} \cdot \mathbf{v}^{\text{pref}}}{|\bar{\mathbf{v}}| \cdot |\mathbf{v}^{\text{pref}}|} > 0.9$, then this congestion point can be ignored and the agent can follow the crowd without any path adjustment. Otherwise, the predetermined path should be adjusted by introducing some new sample point to replace the respective old points. The procedure of generating new sample points is as follows:

- Step 1—Search the neighboring grid points for two near comfort points along the directions perpendicular to the current moving direction of the agent.
- Step 2—Find two candidate sample points at which $\rho = \rho^{\text{thresh}}$ by interpolation between each near comfort point and congestion point as shown in Fig. 6.
- Step 3—Select one of the candidate sample points as the final choice by minimizing the following cost function:

  Where α, β, γ are weights for individual terms. In the first term θ is the angle between the adjusted path and the initial path. The last two terms serve to ensure the moving direction of agent and the moving tendency of the crowd consistent.

Agent will update his $\mathbf{v}^{\text{pref}}$ once the current path has been adjusted. Nevertheless, the agent still emits detection ray towards his original destination so that he can move along the direction towards the original goal again as soon as the ahead region becomes sparse. When searching for the comfort points, we specify a maximum number of queries to prevent the newly generated sample points far away from original path. If no comfort point is found, the neighboring congestion point with the smallest density values is chosen as the new sample point. In our implementation, we perform the path adjustment of each agent only once within a time interval, which greatly increases the efficiency.

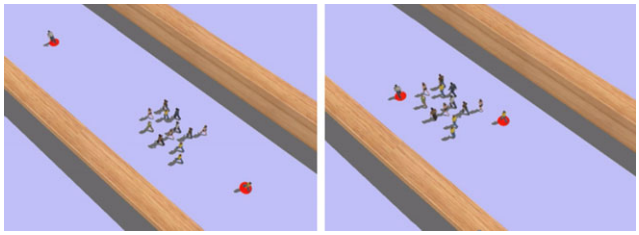

Fig. 7 Simulation in virtual environment

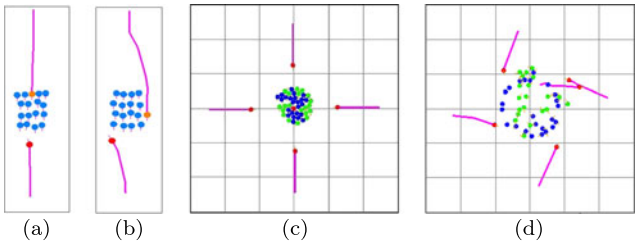

Fig. 8 Comparison displayed in 2D

To dodge the other agents nearby, a collision avoidance approach is employed. The current position and velocity of the neighbored agents can be acquired through the index list stored in the current cell of the agent. Then $\mathbf{v}^{\text{cur}}$ can be obtained according to this information and $\mathbf{v}^{\text{pref}}$. We choose RVO [17] method in our implementation, though any other collision avoidance approach can be used too.

6 Result

The proposed algorithm was implemented on a personal computer with Intel Core2 Due 3.00 GHz CPU, 4G memory and NVIDIA GeForce 9800GTX+ graphics card and our system runs at an average of 15 fps. We first conduct comparisons of our simulation result with that by traditional method, and then analysis the efficiency. Finally we demonstrate the result of integrating virtual pedestrians into videos.

6.1 Simulation

Figure 7 simulates the scene of a pedestrian, on the same road, there is a group. If a group moves along the direction opposite to the pedestrian, the pedestrian will often try to steer clear of the group. If the group moves along the same direction as the pedestrian, the pedestrian will choose to follow or exceed the group according to his relative speed to the group. Figure 8 makes a comparison between results simulated by our method and that by traditional method which use RVO as collision avoidance approach. In Fig. 8(a), the agent adopts only the local collision avoidance algorithm and eventually he squeezes into the group. Figure 8(b) shows our simulation result on the same case, the agent tries to steer clear of the group at an early time. Figures 8(c) and (d) show another scene, traditional methods [17] cannot detect the congested regions in advance. However, by adopting the method proposed in this paper the agent tries to avoid congested regions, and his trajectory is quite similar to a vortex in fluid simulation.

Next we analyze the performance of dynamic path planning algorithm. Because the path adjustment process is dispersed to a period of time, the overhead of our method is mainly on the environmental information maintenance. Let the pedestrians distribute randomly in the scene, we check the computational cost of information update under different conditions. As shown in Fig. 9, the performance varies depending on both the number of agents and the grid resolution. As our path adjustment is implemented at a low resolution grid, the overhead of integrating dynamic path planning module into simulation can be ignored.

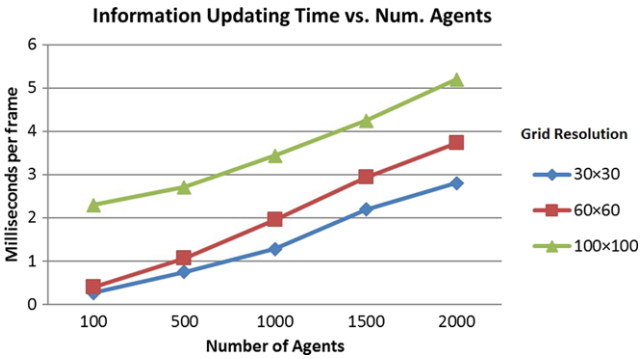

Fig. 9 Performance graph

6.2 Integrating virtual pedestrian into videos

Figure 10 shows a few frames of a video with a presence of a pedestrian group. Several virtual pedestrians are integrated into the video. In Fig. 10(a), the virtual pedestrian has detected the group and chooses to steer clear of it. During the way, he also tries to avoid collisions with other pedestrians nearby. In Fig. 10(b), two groups of pedestrians crisscross, the virtual pedestrian follows the group which moves along the same direction.

Some groups are small scale groups, the pedestrians within each group have social relevance with each other [27], such as friends, family members. This kind of group often consists of two–four members. Sometimes the member does not keep a closed form during walking, even though they are often recognized as a crowd in the space, and the pedestrian often chooses to steer clear of it. We can simulate such phenomenon by adjusting the ε in Eq. (1). A larger ε for members within the group will make the space more crowded, leading other pedestrian steer clear of it. This case is simulated by Fig. 11.

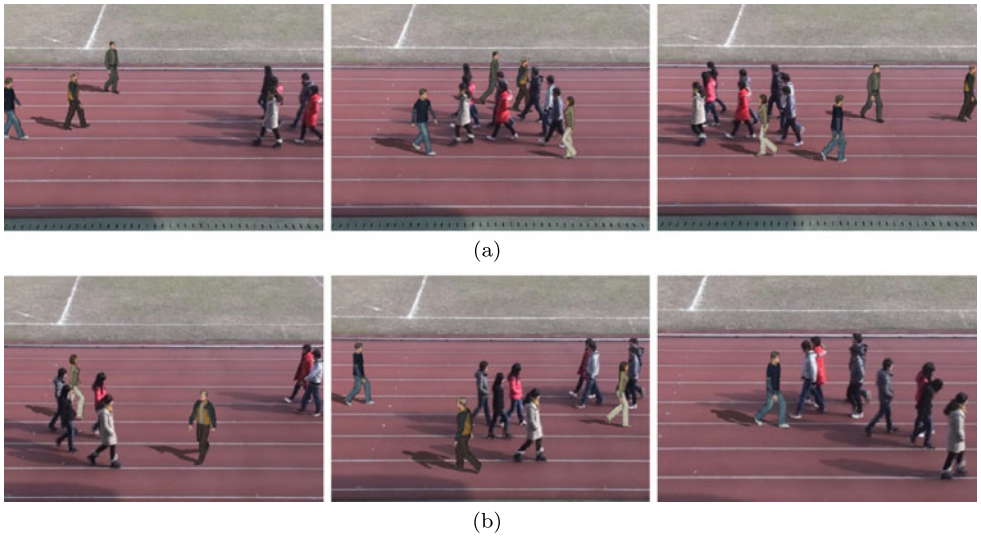

Fig. 10 Mixed videos

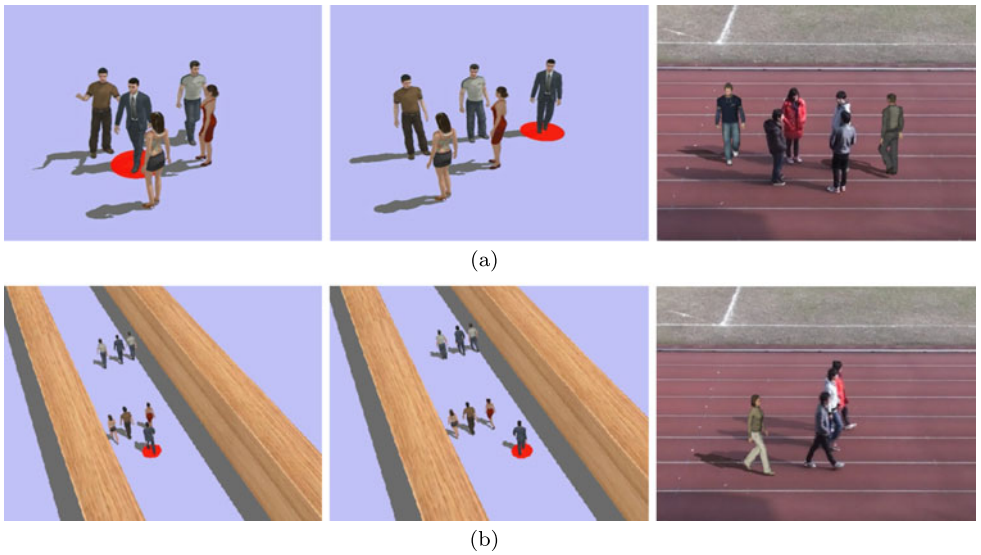

Fig. 11 Simulation of small group. In the *left column*, the pedestrian marked by *red disk* walks through the group if ε is small. In the *center column*, the virtual pedestrian chooses to steer clear of the group instead of passing through it when increasing ε. The *right column* shows examples in video with small group

Fig. 12 Compatible scenes

Figure 12 compares the compatible scenes of our system with previous work. The left image is extracted from the demo of Zhang et al. [25] indicating a scene with a sparse distribution of individuals, while the center image shows the scene containing a pedestrian group. In [25], accurate position of every individual should be available in order to perform the simulation, and this requirement can hardly be satisfied in case of a scene with groups since the current segmentation method cannot robustly segment each individual in the group. Therefore, the approach presented in [25] is not applicable to the scene as shown in the center image of Fig. 12, on the contrary, our approach, however, is capable of dealing with this kind of scenes and the result of our system is shown in the right image.

7 Conclusion and future work

We have proposed a mixed reality approach allowing virtual pedestrians to be integrated into a real scene video which

contains pedestrian groups with behavior consistency. This is achieved by dynamic path planning of virtual pedestrians. Our method tries to detect and track the real pedestrians on each frame of the video and stores their moving condition in a sparse grid. During the way of a virtual pedestrian, the respective agent frequently emits the detection rays through the environment cells to find both the macroscopic and the microscopic information about the real pedestrians ahead of him and adjust the original path if necessary. The adoption of the grid cell data structure accelerates the searching process and improves the efficiency of collision avoidance. Our dynamic path planning module is not dependent on any collision avoidance algorithm, thus can be integrated into existing simulation frameworks easily.

Our approach has some limitations. First, the accuracy of the computed density map depends on the performance of pedestrian detector. In addition, the current approach cannot segment the individuals in the group accurately. If virtual pedestrians should walk through the group, it cannot present the correct occlusion due to lack of silhouettes of some individuals in the group. More robust pedestrian detection and segmentation approaches are preferred. In the aspect of pedestrian simulation, our proposed path adjustment may fail when the concerned region get blocked, for example, a corridor is filled with people. Currently we just let virtual pedestrian stop and wait for a moment, a sophisticated method can be developed to make an appropriate decision.

Acknowledgements This paper was supported partly by National Basic Research Program of China under granted No. 2009CB320802 and National Natural Science Foundation of China under granted No. 61272302.

References

1. Wu, J., Geyer, C., Rehe, J.M.: Real-time human detection using contour cues. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation, pp. 860–867. IEEE, New York (2011)
2. Sudowe, P., Leibe, B.: Efficient use of geometric constraints for sliding-window object detection in video. In: Proceedings of International Conference on Computer Vision Systems, pp. 11–20. ACM, New York (2011)
3. Benenson, R., Mathias, M., Timofte, R., van Gool, L.: Pedestrian detection at 100 frames per second. In: Proceedings of Computer Vision and Pattern Recognition, pp. 2903–2910. IEEE, New York (2012)
4. Andriluka, M., Roth, S., Schiele, B.: People tracking-by-detection and people detection-by-tracking. In: Proceedings of Computer Vision and Pattern Recognition, pp. 1–8. IEEE, New York (2012)
5. Reichlin, F., Leibe, B., Koller-Meier, E., van Gool, L.: Online multiperson tracking-by-detection from a single, uncalibrated camera. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 1820–1833 (2011)
6. Pellegrini, S., Ess, A., Schindler, K., van Gool, L.: You'll never walk alone: modeling social behavior for multi-target tracking. In: Proceedings of 2009 IEEE 12th International Conference on Computer Vision, pp. 261–268. IEEE, New York (2009)
7. Piccardi, M.: Background subtraction techniques: a review. In: Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 3099–3104. IEEE, New York (2004)
8. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**, 888–905 (2000)
9. Malladi, R., Sethian, J.A., Vemuri, B.C.: Shape modeling with front propagation: a level set approach. IEEE Trans. Pattern Anal. Mach. Intell. **17**, 158–175 (1995)
10. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Trans. Graph. **25**, 1160–1168 (2006)
11. Jiang, H., Xu, W., Mao, T., Li, C., Xia, S., Wang, Z.: Continuum crowd simulation in complex environments. Comput. Graph. **34**, 537–544 (2010)
12. Henderson, L.F.: The statistics of crowd fluids. Nature **229**, 381–383 (1971)
13. Narain, R., Golas, A., Curtis, S., Lin, M.C.: Aggregate dynamics for dense crowd simulation. ACM Trans. Graph. **28**(122), 1–8 (2009)
14. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**, 4282–4286 (1995)
15. Reynolds, C.W.: Steering behaviors for autonomous characters. In: Proceedings of Game Developers Conference, pp. 763–782. Miller Freeman, San Francisco (1999)
16. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res. **17**, 760–772 (1998)
17. van den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal *n*-body collision avoidance. In: Proceedings of the 14th International Symposium on Robotics Research, vol. 70, pp. 3–7. Springer, Berlin (2009)
18. Pettré, J., Ondřej, J., Olivier, A.H., Cretual, A., Donikian, S.: Experiment-based modeling simulation and validation of interactions between virtual walkers. In: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 189–198. ACM, New York (2009)
19. Kim, S., Guy, S., Liu, W., Lin, M.: Predicting pedestrian trajectories using velocity-space reasoning. In: Proceedings of the 10th International Workshop on the Algorithmic Foundations of Robotics. Springer, Berlin (2012)
20. Azuma, R.: A survey of augmented reality. Presence, Teleoper. Virtual Environ. **6**, 355–385 (1997)
21. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: Proceeding of ACM Symposium on User Interface Software and Technology, pp. 559–568. ACM, New York (2011)
22. Abad, F., Camahort, E., Viv, R.: On the integration of synthetic objects with real-world scenes. In: Proceedings of EUROGRAPHICS. IEEE, New York (2002)
23. Kim, H., Sohn, K.: 3D reconstruction from stereo images for interactions between real and virtual objects. Signal Process. Image Commun. **22**, 61–75 (2005)
24. Somasundaram, A., Parent, R.: Inserting synthetic characters into live-action scenes of multiple people. In: Proceedings of the 16th International Conference on Computer Animation and Social Agents, pp. 137. IEEE, New York (2003)
25. Zhang, Y., Pettré, J., Ondřej, J., Qin, X., Peng, Q., Donikian, S.: Online inserting virtual characters into dynamic video scenes. Comput. Animat. Virtual Worlds **22**, 499–510 (2011)
26. Rosales, R., Sclaroff, S.: 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 123. IEEE, New York (1999)

27. Karamouzas, I., Overmars, M.: Simulating the local behaviour of small pedestrian groups. In: Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology, pp. 183–190. ACM, New York (2010)

Zhiguo Ren is a Ph.D. student in the State Key Lab of CAD&CG, Zhejiang University. He obtained his B.S. degree from Wuhan University of Technology in 2010. His research interests include crowd simulation, motion planning, and mixed reality.

Wenjing Gai is a Master's student in the State Key Lab of CAD&CG, Zhejiang University. She received her B.S. degree from Sichuan University in 2011. Her research interests include augmented reality and image processing.

Fan Zhong is a lecturer of School of Computer Science and Technology, Shandong University. He received his Ph.D. from Zhejiang University in 2010. His research interests include image and video editing, computer vision.

Julien Pettré is research scientist at INRIA in Rennes, France since 2006. He received M.S. degree in 2000 and obtained his Ph.D. in 2003 from the University of Toulouse III in France. His research topics are crowd simulation, motion planning, autonomous virtual humans, computer animation and virtual reality.

Qunsheng Peng is a Professor in the State Key Lab of CAD&CG at Zhejiang University. He graduated from Beijing Mechanical College in 1970 and received a Ph.D. from School of Computing Studies, University of East Anglia, in 1983. He is currently the chairman of the Professional Committee of CAD and Graphics, China Computer Federation. His research interests include realistic image synthesis, virtual reality, scientific visualization and biological computing, etc.