Long Term Real Trajectory Reuse Through Region Goal Satisfaction

Junghyun Ahn[1], Stéphane Gobron[1], Quentin Silvestre[1], Horesh Ben Shitrit[1], Mirko Raca[1], Julien Pettré[2], Daniel Thalmann[3], Pascal Fua[1], and Ronan Boulic[1]

[1] EPFL, Switzerland
[2] INRIA-Rennes, France
[3] NTU, Singapore

Abstract. This paper is motivated by the objective of improving the realism of real-time simulated crowds by reducing short term collision avoidance through long term anticipation of pedestrian trajectories. For this aim, we choose to reuse outdoor pedestrian trajectories obtained with non-invasive means. This initial step is achieved by analyzing the recordings of multiple synchronized video cameras. In a second off-line stage, we fit as long as possible trajectory segments within predefined paths made of a succession of region goals. The concept of region goal is exploited to enforce the principle of "sufficient satisfaction": it allows the pedestrians to relax the prescribed trajectory to the traversal of successive region goals. However, even if a fitted trajectory is modified due to collision avoidance, we are still able to make long-term trajectory anticipation and distribute the collision avoidance shift over a long distance.

Keywords: Motion trajectories, Collision handling

1 Introduction

The present paper is motivated by two goals: first producing more plausible pedestrian crowds by reusing pedestrian trajectories captured with non invasive means, and second, relying on general principles of human behavior to minimize the computing cost when reusing as large as possible trajectory segments. We believe that a first step towards the ecological validity of the crowd motion is to capture pedestrian trajectories in an outdoor pedestrian area. Second, this material has to be obtained through a non-invasive means; for this reason we exploited multiple overlapping video cameras.

Instead of inferring a characterization of the captured pedestrian area, such as in term of a force field, we chose to completely decouple the measured trajectories from their initial context by searching how they could fit to a new spatial environment. Our objective is to assemble as fast as possible and adjust as little as possible large segments of real pedestrian trajectories to obtain a plausible variety of crowd motions. In particular we wish to reduce the occurrences of implausible short term collision avoidance by taking advantage of the

short term future of the pedestrian paths. The adjustment to the trajectories are designed to be minimal owing to the interaction of two general principles of human behavior identified in Psychology [29, 21, 25] and already partly exploited in crowd simulation [11, 3]. The principle of least efforts [29] states that people selects the action that require the least effort when given multiple choices to accomplish a task. It has been exploited in [11] to justify the choice of energy minimizing path (both movement and turning effort) when moving through a crowd. However Simon [21] pointed out that humans do not necessarily search for the optimal solution based on all the available solution because they lack the cognitive resources. They rather "satisfice", a term he coined by combining "satisfy" and "suffice", *i.e.* they reach good-enough solution through simplifying heuristics despite their occasional failing [25]. Within this frame of mind, we advocate for generalizing the concept of region goals introduced in [3] for steering an isolated pedestrian towards an oriented region goal. In the present case we intend to select real trajectories so that they fit in a succession of region goals and to adjust them through collision avoiding shifts that satisfy those region goals. One key advantage is the reduced cost of the adjustment step that allows to distribute the trajectory shift long before the potential collision occurrence.

The next section recalls the main background material mostly on crowd simulation. The human trajectory extraction is described in section three while section four presents the new approach for the reuse of long term trajectory segments and their on-line adjustment. Section five and six respectively shows some result and offer some directions for future work.

2 Related Works

The topic of crowd simulation has stimulated a large number of contributions. For this reason we focus in priority on those exploiting pre-existing real or synthesized trajectories or force fields. Brogan and Johnson [5] have built a walking path model from measurements from which they construct a heading chart ensuring trajectories with minimal radius of curvature towards a goal while avoiding static obstacles. By construction the chart is dedicated to the original environment and suited for the goals that were recorded. Chenney proposed to assemble flow tiles over the whole environment to guide pedestrians [6]; it has a clear interest for low cost background movement but it lacks the natural variety of human movements. Likewise [16] combines the attractive force field of a guiding trajectory with other standard force fields. These ideas are also developed in [18]. In [22] the proposed guiding flow is based on the concept of natural movement stating that humans have the tendency to follow their line of sight [9]. The work of Treuille *et al* offers an elegant solution based on a dynamic potential field for guiding groups of pedestrians with the same goals in large scale environments [23]. Reitsma and Pollard evaluate the suitability of an environment to a set of recorded movement organized as a motion graph [20]. In [14] an agent model is learned from recorded aerial view of group motion and is able to reproduce a wide range of behaviors; it is still limited by its computing cost. Similarly [7] exploits crowd

video footage but here to extract time series of velocity fields that are later used to advect people along a time varying flow. An alternate approach is proposed in [15] in which local relative movements are identified from video and stored in a database. Then the database is searched on-line to derive short term movements from the closest found pattern. The computing cost of this type of approach is still high. Yersin *et al* have proposed the concept of Crowd patch that can be assembled on the fly to allow the travel through a potentially infinite inhabited virtual city [28]. Periodic trajectories running through multiple patch constitute the core element of this contribution. Multiple contributions have focused on the simulation of groups [13, 12, 26] but this is beyond the scope of the present paper as we don't address the constitution and the controlled deformation of groups.

Our own approach relies on the one described in [19] for the stage of general trajectory planning producing a set of variant paths for large group of pedestrians between two regions in a virtual environment. We differ in the way the variant paths are exploited to produce the individual pedestrian trajectories. Our contribution is to fit the longest possible real trajectory segment within the path variant and to adjust them on the fly by taking advantage of the three zones of interest proposed in [27].

3 Extracting Human Trajectories from a Real Scene

Extracting human trajectories in a crowded scene is an active domain in the vision community. There exist several methods for reliable tracking in long sequences [2, 4]. Our multiple people tracking relies on pedestrians detections from multiple cameras [8], and multiple object tracking [2]. As the pedestrians detector requires multiple synchronized and calibrated cameras, we initialize our processing pipeline with calibrating the cameras using the Tsai calibration model [24]. Then, we subtract the background and the shadows from the video frames and feed only the binary foreground-background masks to the pedestrians detector. The pedestrians detector integrates the binary masks from all the cameras. The ground-plane is partitioned into grid cells. In each frame, the people detector estimates the probability of each grid cell to be occupied by a person [8]. Next, the tracking algorithm efficiently solves the detection association task by formulating it as a global optimization problem, which is solved using the K-Shortest Paths algorithm (KSP) [2]. Finally, we post-process the trajectories in order to obtain smooth and accurate trajectories. In the following subsections, we explain in great detail the different algorithms used in this work.

3.1 Background Subtraction and Shadows Removal

The background subtraction produces binary images, in which static parts are labeled differently than the dynamic ones. This technique is efficient, but it can be sensitive to differences in lighting conditions, colors of subjects similar to the background and shadows. For our system we used the EigenBackground algorithm [17], which models the background using eigenvalue decomposition of

several reference images. It is capable of dealing with global illumination changes, but does not remove pedestrians shadows as they are dynamic as well.

Elimination of shadows in the footage was made additionally difficult due to low quality of our video sequences and lack of distinct colors in the clothing of the pedestrians present. This meant that we could not use different statistical properties of the surface which are usually used to detect shadows such as texture information. Our method uses the fact that the shadows are *(i)* darker variations of the background color and *(ii)* have a distinct shape usually spread in the horizontal direction, which is the main difference from the human shapes which are primarily oriented in the vertical direction. After simple background subtraction each pixel in the foreground can be classified into one of 8 labels based on which angular direction has the largest number of foreground pixels (the *dominant orientation*, Fig 1(c)). By specifying the expected dominant orientation of the shadows, we can substitute the appropriate pixels with the original background color (Fig 1(d)) or remove them from the foreground map used by the tracking algorithm (Fig 1(e)). Since this approach is possible due to consistency in the shadow direction, it needs to be refined to handle dynamically changing shadows or shadows with dominant penumbra component.

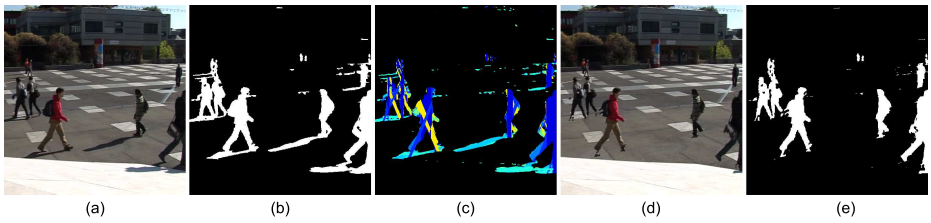

Fig. 1. Shadows removal process. (a) Original frame with shadows; (b) Foreground map after simple background subtraction; (c) Dominant orientations of each foreground pixel. Notice that the shadow areas have consistent classification compared to the rest of the human body; (d) Final video frame, with shadow pixels re-colored with background color; (e) Final foreground map

3.2 Multiple Pedestrian Detection and Tracking

We adopt the multiview people detector of [8]. In each frame, the detector integrates the binary background-foreground information from different cameras with respect to their calibration, and estimates the positions of the pedestrians on the ground plane. The ground plane is partitioned into uniform, non-overlapping grid cells, typically with size of 25 by 25 (cm). The detector provides an estimation of the probability of each grid cell to be occupied by a person. Hence, it produces a Probability Occupancy Map (POM) [8]. We use the publicly available implementation of the algorithm[4].

[4] POM: `http://cvlab.epfl.ch/software/pom`

Given the person locations estimated by POM, we use the K-Shortest Path (KSP) tracker [2]. The KSP formulates the tracking task as a global optimization problem on a Directed Acyclic Graph (DAG), yielding a convex objective function. Its computational complexity is $O(k(m + n \log n))$, where m, n, and k are the number of graph nodes, edges, and trajectories. The algorithm is suitable for tracking people in large areas for a long time period. However, even though the KSP has a space complexity of $O(n)$, this turns out to consume a lot of memory resources. Using the publicly available version of the algorithm[5], we could only process 2000 frames. Therefore, we introduced a pruning mechanism for reducing the consumed memory. Using the POM results, we kept only the detections that were above a certain threshold $Thr1 = 0.75$, in addition to their spatio-temporal neighborhood. We defined the neighborhood as all the cells that are proximate less than $Thr2 = 12$ cells to a cell with a detection. Using this very low threshold we managed to keep the tracking performance high, and still to prune more than half of the graph's edges. Thus, we have the ability to process more than 6000 frames, in one batch.

3.3 Trajectory rectification

As was mentioned in the previous subsection, positions of human trajectories have been extracted from 25 *cm* square grid cells. Because of this approximation, we couldn't avoid the problem of rough trajectories. Besides, other issues depicted in Fig 2 gave us the motivation to design rectification steps as follows:

1. Generate local and global confidences of all the frames and trajectories
   - **Local confidence**: defines reliability of a single frame in a trajectory
   - **Global confidence**: defines reliability of the entire trajectory
2. Remove low local confidence frames (issues #1 and #3 in Fig 2)
3. Remove low global confidence trajectories (issue #2)
4. Fill missed frames in each trajectory
5. **Smoothing** rough trajectories (issue #4)

Local confidence Two different formulations have been defined for local confidence. One from distance measure and the other one from speed at a given frame (see upper and lower equation of Fig 2). We defined two different distance measures d_1 and d_2, which represents distance to the border of capture area and distance between two trajectory positions at the same frame, respectively. The speed (m/s) is defined by subtracting the current position to the previous frame. In the upper equation, $C_d(d)$, constant k, which defines the slope of the *logistic function*, was set to 3. In the lower equation of Fig 2, $C_s(s)$, constants $S1$ and $S2$ are respectively the average human walking speed 1.4 and the width of the *gaussian function* defined as 24.5 (full width at half maximum 3.5). We pruned frames to $C_d(d_1) < 0.55$ and $C_s(s) < 0.6$.

[5] KSP: `http://cvlab.epfl.ch/software/ksp`

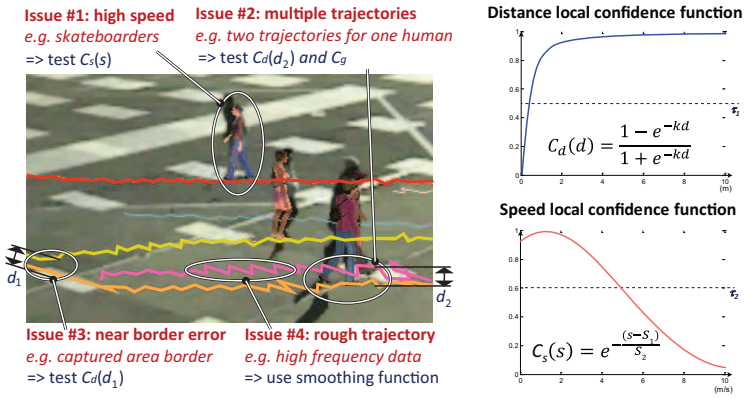

Fig. 2. Problems statement and their relation to local confidence equations. Bold red texts state problems of reconstruction and blue texts starting with "⇒" their solutions. On the right, shows two different local confidence equations with their pruning threshold (horizontal dashed line). Each equation's y-axis gives a value range [0,1], which represents the reliability of a reconstructed frame.

Global confidence The global confidence, $C_g = (\sum_{k \in Tr} \rho_k)/|Tr|$, is a general measure that defines how high were the probabilities of the detections. It is based on the sum of detection probabilities ρ_k at each location k which belong to the trajectory Tr. This sum is normalized by the length of the trajectory $|Tr|$. By construction, the KSP algorithm provides trajectories with global confidence $C_g > 0.5$. During the rectification process, we solve the "issue #2" stated in Fig 2. We first check if two trajectory positions at a given frame is $C_l(d_2) < 0.5$, and remove one of them by comparing C_g of both trajectories.

Smoothing Smooth filtering was exploited for the purpose of removing noisy effect on the trajectories. We applied different smoothing filters such as *Moving average*, *Savitzky-Golay*, and *Local Regression* with various span parameters from 0.1% to 5.0%. All filters were processed in the 2^{nd} polynomial degree. Among them, we found *Local Regression* with 0.5% had more similarity to the rough trajectories with low noisy effects.

4 Re-using Human Trajectories in a Virtual Scene

4.1 Generating path from given trajectories

Environment setup with navigation graph A typical trajectory capture session should result in a few hundreds trajectory segments with a wide variety of lengths. Each trajectory maintains the information of position, direction and speed of each frame. The first problem we want to address is to fit the longest possible trajectory segment within a set of region-to-region path variants precomputed according to the approach described in [19]. The path variants are

constructed from a navigation graph where nodes are (static) collision-free circular regions called vertices (Fig 3 left). Each path variant is defined by a set of successive vertices that link two potentially distant regions of the virtual environment (Fig 3 right). By construction, the path variants may share vertices in narrow parts of the scene, so it is also necessary to address dynamic collision detection as we show later.

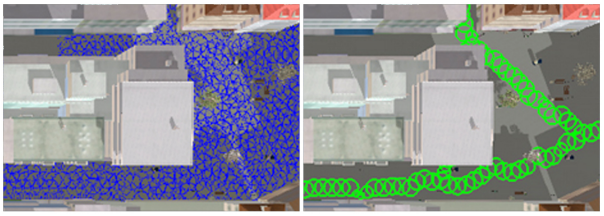

Fig. 3. The Navigation graph [19] illustrated in this city scene (top view) guarantees that no collision with static elements of the environment can occur within the sampled circular regions (left); path variants are automatically built to link distant regions through a list of vertices (right)

Preprocessing path trajectory The detailed process of determining a path trajectory is illustrated in Fig 4. The process is off-line and we have the guarantee that the match will succeed owing to a sufficient pool of short trajectory segments and to the relatively large size of the vertices. In order to ensure some variety of trajectory, we randomly initialize the candidate trajectory segment as follows: 1) random position of the starting position (P_1 in V_0) within the start vertex and 2) random orientation of the candidate segment start-to-end vector within the angular sector under which the target vertex (P_2 in V_{end}) is viewed. At the end of the matching process we apply a local smoothing, by bezier curve approximation, in the region linking successive segments.

4.2 Real-time collision handling

The advantage of reusing known trajectories is the possibility to estimate collision long time ahead. Based on future trajectory information, we formulated a real-time collision avoidance algorithm illustrated in Fig 5. For the collision avoidance model we first define two circular shape areas as follows. The first circle represents a **colliding area** (inner circle) which represents average breast width of human, and second circle represents a **shift influencing area** (dashed circle) which represents an area where people feel threat of collision. We admit that the shape of shift influencing area is rather an elliptic or more complex shape [15] [10], however, in this paper we simplify our approach for real-time simulation, and to compensate this area simplification, we gave more weights to the collision threat coming from forward walking direction.

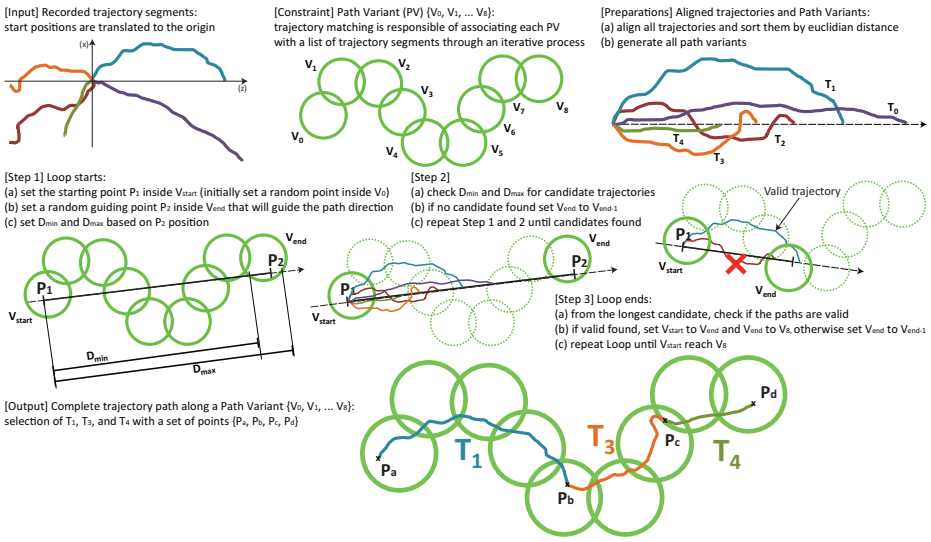

Fig. 4. A fully detailed process of trajectory path generation on a path variant.

The proposed collision avoidance method has mainly two steps: generating 1) direction and 2) magnitude of trajectory shift. The detail of getting this instantaneous avoiding direction is as follows. At a given time t_0, a virtual character H_i checks its collision by looking up trajectory paths ahead. In the present implementation we sub-sampled future positions every N_f frames. We denote t_c as the time of a destined collision detected at time t_0, and t_f as the time after the collision such that t_f-t_c is equal to t_c-t_0. When H_i detects a destined collision at t_c, it analyzes the relative positions of all the other characters (H_{ij}) in the current View Frustum (VF). Here the relative positions $H_{ij}(t_0)$, $H_{ij}(t_c)$, and $H_{ij}(t_f)$ are the positions of H_{ij} in H_i local coordinate. For each character H_{ij} two line segments $\mathbf{l}_{ij}(t_c)$ and $\mathbf{l}_{ij}(t_f)$ are built (see Fig 5). Given these line segments, we are able to compute the **shift influence vectors** $\mathbf{v}^s_{ij}(t_c)$ as in Eq 1. First, we define a vector $\mathbf{v}^l_{ij}(t_c)$, which represents the minimum distance vector from $\mathbf{l}_{ij}(t_c)$ to H_i. The vector $\mathbf{v}^s_{ij}(t_c)$ is generated by re-scaling $\mathbf{v}^l_{ij}(t_c)$ with the radius of the shift influencing area R_s. The vector $\mathbf{v}^s_{ij}(t_f)$ is calculated in a similar way by replacing t_c by t_f.

$$\mathbf{v}^s_{ij}(t_c) = \frac{R_s - |\mathbf{v}^l_{ij}(t_c)|}{|\mathbf{v}^l_{ij}(t_c)|}\mathbf{v}^l_{ij}(t_c) \tag{1}$$

By accumulating all the shift influence vectors ($\sum_{j \in VF}(w_c\mathbf{v}^s_{ij}(t_c)+w_f\mathbf{v}^s_{ij}(t_f))$), we finally get a **trajectory shift vector** $\mathbf{v}_i(t_0)$, which guides H_i to avoid collision at time t_0. The weighting factors w_c and w_f are defined as 2/3 and 1/3.

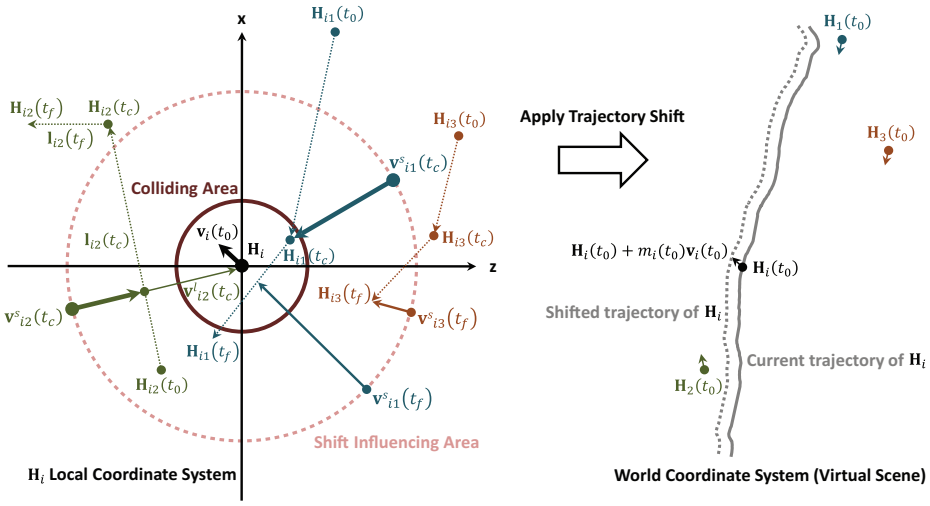

Fig. 5. The collision avoidance mechanism of character H_i; Left: the trajectory shift vector $\mathbf{v}_i(t_0)$; Right: shifting trajectory at run time.

Additional shift influence vectors Besides these shift influence vectors, other factors were also considered for calculating trajectory shift vector. Each trajectory path may not surpass the limit of the current path variant's vertex. We check the magnitude of the sum of the shifting vectors accumulated from previous simulation loops, and apply an additional shift influence vector heading toward the original trajectory. Moreover, in case of ideal linear frontal collision without any other intruder could fail to avoid each other. To solve this problem, we give a weak left or right shift influence on forward walking direction. The trajectory shift vector $\mathbf{v}_i$ is *normalized* after adding all shift influence vectors.

Magnitude of trajectory shift vector For a realistic collision avoiding movement, we formulated a magnitude m_i, which gives an instantaneous distance to deviate path. For each simulation loop, if a character H_i detect a collision threat at time t_c, it analyzes the scaling factor $m_i(t_0)$ (see Eq 2), which will be multiplied by the given normalized vector $\mathbf{v}_i(t_0)$ for shifting H_i's trajectory. Eq 2 computes the instantaneous collision avoidance speed for the current time step. The durations $\Delta t_s(t_0)$ and Δt_v are respectively the current inverse of display rate at time t_0, and the inverse of captured video rate (constant $0.04sec.$). The distances R_c and $min(|\mathbf{v}^l_{ij}(t)|)$ are respectively the radius of collision area and the minimum distance among the different j. Finally, the parameter $f_c(t_0)$ is the number of frames left before collision at time t_0. A maximum magnitude has been also defined to reduce jerky motions.

$$m_i(t_0) = \frac{\Delta t_s(t_0)(R_c - min(|\mathbf{v}^l_{ij}(t_c)|))}{\Delta t_v f_c(t_0)} \tag{2}$$

5 Results

From the video capture, we obtained 30,000 frames recorded at 25-fps. Among them, 10,000 frames have been selected as rough trajectories and 297 trajectories were generated by the reconstruction and rectification process. By the smoothing process, we tried to apply optimal filtering method and parameter, so that we could keep more originality of trajectory with less noisy effects.

Fig 6 illustrates the results of our approach. The city scene consists of 357 path variants over the entire walkable area. For each path variant, in average, around 135 successive *navigation graph vertices* are connected. A vertex is shared by a number of path variants which makes possible to generate lots of variant paths. In our result, one trajectory path was generated for each path variant. The size of preprocessed 357 trajectory paths was 161 MB in ASCII format.

Fig. 6. Top left: reconstructed and rectified real trajectories from video capture; Top right: real trajectory scene with free viewpoint; Bottom left: preprocessed trajectory paths; Bottom right: trajectory reused scene with collision avoidance (500 characters).

An experiment of simulation speed was conducted by varying number of characters. We compared to a previous work [27] and the results shows as in Table 1. The computing cost of our approach was slightly higher than the previous work. However, from the movie at `http://iig.epfl.ch/movies`, we could see that real trajectory can produce more plausible animation when a character avoids collision or follows its animation path. We also noticed that characters following real trajectories looks more realistic with various speed and movement along a path. In case of 1,000 characters simulation, about twice of computation is needed than motion simplification method [1]. However, the collision avoidance

was not considered. Our collision avoidance algorithm didn't work perfectly in a high density crowd scene. We think this problem can be solved by considering the speed variation on trajectory shifting.

# of characters	250	500	1000	2000
Yersin *et al* [27]	60.0	27.7	14.7	7.7
Our approach	60.0	25.6	13.6	6.6

Table 1. A display rate (fps) comparison between two different methods from the same camera view point (environment: NVidia GTX 460 1GB).

6 Conclusion

The methods presented in this paper cover the whole pipeline for automating non-invasive pedestrian trajectories processing and reuse in arbitrary scenes. Contributions encompass both the real trajectories recovery from multiple video cameras, for which large speedups have been obtained, and the trajectory reuse for long-term prediction and avoidance of collisions. We will proceed to extensive benchmarking to evaluate whether the simplified circular shape of the shift influence area is detrimental for the plausibility of the resulting crowd motion. More comparisons with prior approaches would be beneficial although none have adopted our reuse strategy for collision avoidance. Future work will also focus on the exploitation of this know-how in Augmented Reality to demonstrate the efficient editing of the original captured crowd (*e.g.* adding lifelike pedestrians or removing existing pedestrians).

Acknowledgements

This research has been funded by the Swiss National Science Foundation Synergia project AerialCrowds.

References

1. Ahn, J., Oh, S., Wohn, K.: Optimized motion simplification for crowd animation: Research articles. Comput. Animat. Virtual Worlds 17, 155–165 (July 2006)
2. Berclaz, J., Fleuret, F., Türetken, E., Fua, P.: Multiple object tracking using k-shortest paths optimization. PAMI (February 2011)
3. Boulic, R.: Relaxed steering towards oriented region goals. In: Motion in Games '08, LNCS, vol. 5277, chap. 18, pp. 176–187. Springer Berlin Heidelberg (2008)
4. Breitenstein, M.D., Reichlin, F., Leibe, B., Koller-Meier, E., Gool, L.V.: Online multiperson tracking-by-detection from a single, uncalibrated camera. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 1820–1833 (2011)

5. Brogan, D.C., Johnson, N.L.: Realistic human walking paths. In: CASA '03. pp. 94–101 (2003)
6. Chenney, S.: Flow tiles. In: SCA '04. pp. 233–242 (2004)
7. Courty, N., Corpetti, T.: Crowd motion capture. Computer Animation and Virtual Worlds 18, 361–370 (September 2007)
8. Fleuret, F., Berclaz, J., Lengagne, R., Fua, P.: Multi-camera people tracking with a probabilistic occupancy map. PAMI 30(2), 267–282 (February 2008)
9. Hillier, B., Penn, A., Hanson, J., Grajewski, T., Xu, J.: Natural movement: or, configuration and attraction in urban pedestrian movement. Environment and Planning B: Planning and Design 20(1), 29–66 (January 1993)
10. Kapadia, M., Singh, S., Hewlett, W., Faloutsos, P.: Egocentric affordance fields in pedestrian steering. In: I3D '09. pp. 215–223. ACM (2009)
11. Karamouzas, I., Heil, P., van Beek, P., Overmars, M.H.: A predictive collision avoidance model for pedestrian simulation. In: MIG '09. pp. 41–52. Springer (2009)
12. Karamouzas, I., Overmars, M.: Simulating the local behaviour of small pedestrian groups. In: VRST '10. pp. 183–190. ACM (2010)
13. Kwon, T., Lee, K.H., Lee, J., Takahashi, S.: Group motion editing. ACM Transactions on Graphics 27, 80:1–80:8 (August 2008)
14. Lee, K.H., Choi, M.G., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: SCA '07. pp. 109–118 (2007)
15. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Computer Graphics Forum 26(3), 655–664 (2007)
16. Metoyer, R.A., Hodgins, J.K.: Reactive pedestrian path following from examples. In: CASA '03. IEEE Computer Society (2003)
17. Oliver, N., Rosario, B., Pentland, A.: A bayesian computer vision system for modeling human interactions. PAMI 22(8), 831–843 (2000)
18. Park, M.J.: Guiding flows for controlling crowds. Vis. Comput. 26, 1383–1391 (November 2010)
19. Pettré, J.: Populate your game scene. In: Motion in Games '08, LNCS, vol. 5277, chap. 18, pp. 33–42. Springer Berlin Heidelberg (2008)
20. Reitsma, P.S.A., Pollard, N.S.: Evaluating motion graphs for character animation. ACM Trans. Graph. 26 (October 2007)
21. Simon, H.A.: Rational choice and the structure of the environment. Psychological Review 63(2), 129–138 (1956)
22. Stylianou, S., Fyrillas, M.M., Chrysanthou, Y.: Scalable pedestrian simulation for virtual cities. In: VRST '04. pp. 65–72. ACM (2004)
23. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Trans. Graph. 25, 1160–1168 (July 2006)
24. Tsai, R.: A versatile cameras calibration technique for high accuracy 3d machine vision mtrology using off-the-shelf tv cameras and lenses. JRA 3(4), 323–344 (1987)
25. Tversky, A., Kahneman, D.: Judgment under uncertainty: Heuristics and biases. Science 185(4157), 1124–1131 (1974)
26. Van Den Akker, M., Geraerts, R., Hoogeveen, H., Prins, C.: Path planning for groups using column generation. In: Motion in Games '10. pp. 94–105 (2010)
27. Yersin, B., Maïm, J., Morini, F., Thalmann, D.: Real-time crowd motion planning: Scalable avoidance and group behavior. Vis. Comput. 24, 859–870 (Sept 2008)
28. Yersin, B., Maïm, J., Pettré, J., Thalmann, D.: Crowd patches: populating large-scale virtual environments for real-time applications. In: I3D. pp. 207–214 (2009)
29. Zipf, G.: Human Behaviour and the Principle of Least-Effort. Addison-Wesley, Cambridge, MA (1949)