# Integrating Crowd Simulation for Pedestrian Tracking in a Multi-Camera System

Zhixing Jin, Bir Bhanu
Center for Research in Intelligent Systems
University of California, Riverside
California 92521

*Abstract*—**Multi-camera multi-target tracking is one of the most active research topics in computer vision. However, many challenges remain to achieve robust performance in real-world video networks. In this paper we extend the *state-of-the-art* single camera tracking method, with both detection and crowd simulation, to a multiple camera tracking approach that exploits crowd simulation and uses principal axis-based integration. The experiments are conducted on PETS 2009 data set and the performance is evaluated by multiple object tracking precision and accuracy (MOTP and MOTA) based on the position of each pedestrian on the ground plane. It is demonstrated that the information from crowd simulation can provide significant advantage for tracking multiple pedestrians through multiple cameras.**

## I. Introduction

Pedestrian tracking, especially in crowded scenarios, has been attractive to computer vision researchers for many years. For the purpose of applying pedestrian tracking in various areas of real life such as surveillance, security, and monitoring, hundreds of approaches have been proposed in the past several decades. While the approaches for single camera tracking have become more and more sophisticated to handle some extremely complicated situations, fusing the information from multiple cameras is another common technique to improve tracking performance. By integrating information from different views, one can greatly expand the area under surveillance by deploying multiple cameras focused on different areas, as well as improve the accuracy for tracking by using a series of cameras with overlapping field-of-views to reduce the influence from occlusion, etc [1].

In the proposed approach, we are focused on the situation where a set of overlapping cameras is used to track pedestrians by integrating the information from multiple views of the same area. A common way is to adopt homography-related methods to model the relationship of the information obtained from different cameras so that the actual position of each tracked pedestrian in the real-world can be easily estimated. However, in almost all the current multiple camera trackers, the estimation of real- world position is only related to its corresponding position on the frame from each view, but other relationships between the real-world position of each pedestrian are somehow ignored.

From another perspective, crowd simulation is a type of method utilizing the relationship between the real world position of each pedestrian, which is also a very popular topic in computer graphics, with special importance in the areas such as designing emergency evacuation routes. Basically, crowd simulation is designed for simulating the behavior of every individual in a crowd under the given constraints (*e.g.*, to avoid collisions). Currently the most popular crowd simulations are mainly focused on the simulation of walking (direction and velocity) of pedestrians given the starting and ending positions of each individual. Since the direction and velocity information can be easily acquired in a multiple camera tracking system, it is natural to consider integrating crowd simulation algorithms to provide additional information for accurate positioning in a tracking system.

In this paper, we propose an appropriate way to combine the multiple camera tracking system with a crowd simulation algorithm. In our integrated system, each camera has its own independent tracker based on the tracking-by-detection approach [2], and the simulator works, separately of the tracking system, based on the RVO2 library [3]. At each time step, the simulator generates a distribution of possible positions for each pedestrian in the scene based on their current positions and historical direction and velocity information. This distribution is adopted together with the information from trackers to estimate the current position of each pedestrian on the real ground plane, and further serves as the feedback to each tracker. The system diagram is illustrated in Figure 1.

The rest of the paper is organized as follows. Section II presents related work, including tracking approaches and crowd simulation methods. Section III describes the details of our proposed approach. Section IV demonstrates the experimental results and finally Section V concludes the paper.

## II. Related Work

The *state-of-the-art* tracking approaches usually take the advantages of combining classification and detection methods, and most of them perform in an online manner. Examples are: Online Ada-Boosting [4], Semi-Boosting [5] and Online Multiple Instance Learning [6] trackers. Generally speaking, this type of tracking trains a classifier for the appearance model from the first one or several frames. For the subsequent frames, it finds the position that maximizes the likelihood based on the evaluation from the classifier in a certain search window, and then bootstraps the classifier itself by using the information of the patch at the updated location. In addition to classification,

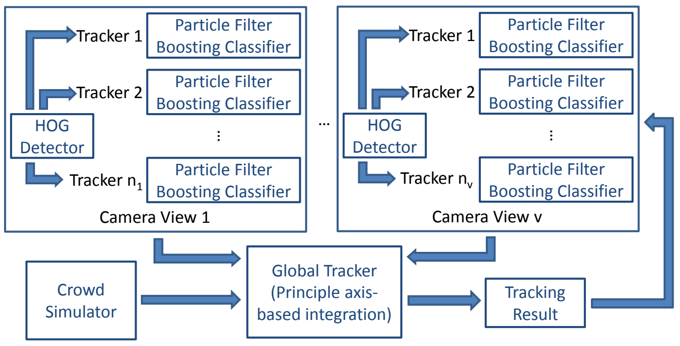
Tracker 1
Particle Filter Boosting Classifier
Tracker 2
Particle Filter Boosting Classifier
HOG Detector
Tracker $n_1$
Particle Filter Boosting Classifier
Camera View 1
Tracker 1
Particle Filter Boosting Classifier
Tracker 2
Particle Filter Boosting Classifier
HOG Detector
Tracker $n_v$
Particle Filter Boosting Classifier
Camera View v
Crowd Simulator
Global Tracker (Principle axis-based integration)
Tracking Result

Fig. 1. The system diagram. This example is a system with $v$ different views.

detection methods, especially human detectors are also integrated into the tracking system [7], [2]. Since human detector is generally more confident than tracker itself, the result from human detector is always used as an automatic initializer for the tracker and a strong corrective factor during the tracking process. The tracking part in the proposed approach used in this paper is based on the most recent tracking-by-detection method [2], which combines a particle filter, a human detector and an online boosting classifier.

When multiple cameras are used, the tracking system becomes more complicated and more research topics can be found. For example, the data correspondence between cameras is an important problem without a perfect solution till now. Many different methods, such as region-based, point-based and principal axis-based, have been reported in recent years [1], [8]. Since in this paper, we are focused on the improvement brought by the crowd simulation to multiple camera tracking, manual data correspondence is used to avoid the error from automatic data correspondence. However, the real position of each individual on the ground plane is still calculated based on the principal axis intersection.

For crowd simulation, there are many different models due to the complexity and uncertainty of human behaviors. For example, the social force model which is derived from physics and social-psychology and currently the most widely used [9]; the RVO2 (Reciprocal Velocity Obstacles) library which uses linear programming to find the optimal collision avoidance strategy [3]; the rule-based approaches combined with local collision avoidance is simple but effective [10]; and the continuum dynamics model from a macroscopic perspective is able to simulate extremely large and dense crowds [11]. Except the macroscopic techniques, most of the crowd simulation approaches are point (individual) based. For example, the social force model considers the crowd as a particle system and defines several different types of forces on each particle (individual) to obtain a solution to the whole system. In the proposed approach, the crowd simulation method adopted is the RVO2 library [3], because it only requires the information of the current position and the desired velocity of each pedestrian, which is quite easy to obtain. In addition, the calculation for the RVO2 model is relatively fast, allowing us to get the distribution of possible locations which is required at many times during the simulation at each time step.

## III. Technical Approach

The system can be divided into two major components. The first component consists of the *state-of-the-art* tracking-by-detection algorithm (frame tracker), which provides the capability to track pedestrians only based on the information from a single camera. The second component is the position calculator in the real world (global tracker). It integrates the information from the crowd simulator and the projection from each frame tracker, and estimates the real position for each pedestrian on the ground plane.

### A. Frame Tracker

The frame tracker in this approach is a sophisticated tracking-by-detection method combining a particle filter, a boosting classifier, and a human detector [2]. However, since our main purpose is to investigate the benefit of crowd simulation, we modify the original technique to eliminate possible errors brought by various steps, as well as speed up the single camera tracking process. For readers who are interested in the complete method, please refer to the paper [2].

The tracking is accomplished mainly based on a *bootstrap filter*. The state $\mathbf{x} = \{x, y, u, v\}$, consists of the information of position $(x, y)$ and velocity $(u, v)$ of each particle (on image frames). Since importance re-sampling is carried out at every time step and $w_{t-1}^i = 1/N$, the weight of each particle at every time step $w_t^i$ only depends on the likelihood of the current observation, $p(o_t|\mathbf{x}_t^i)$, which will be described at the end of this section.

The motion model adopted in this particle filter is a simple constant velocity motion model

$$(x, y)_t = (x, y)_{t-1} + (u, v)_{t-1} + \varepsilon_{(x,y)} \tag{1}$$
$$(u, v)_t = (u, v)_{t-1} + \varepsilon_{(u,v)} \tag{2}$$

where $\varepsilon_{(x,y)}$ and $\varepsilon_{(u,v)}$ are two independent noise functions following zero-mean normal distributions. The variances $\sigma_{(x,y)}^2$ and $\sigma_{(u,v)}^2$ are set initially proportional to the size of the patch and then decrease as the number of successfully tracked frames increases. For simplicity, we skip the *Iterative Likelihood Weighting* procedure adopted in the original tracker to deal with abrupt and fast camera motion. Furthermore, we initialize the tracker by manual annotations to ensure that each pedestrian has a corresponding tracker. The termination strategy of a tracker is the same as the original work, that is, each tracker stops if there are no associated detection results for a while.

The information provided by the human detector includes two parts, the normal detection results and the confidence map. For the normal detection results, it is required to associate them with each tracker. A function [2] calculates the matching scores between each detected patch and tracked patch

$$S(tr, d) = g(tr, d) \cdot \left( c_{tr}(d) + \alpha \cdot \sum_{p \in tr}^{N} p_{\mathcal{N}}(d - p) \right) \tag{3}$$

where $tr$ and $d$ denote the positions of the tracker and the detected patch, respectively. $p_{\mathcal{N}}(d - p)$ is the normal distribution based on the distance between the detection and particle, $c_{tr}(d)$ is the evaluation from the classifier on the detected patch, and $g(tr, d)$ is the gating function

$$\begin{aligned} g(tr,d) &= p(size_d|tr)p(pos_d|tr) \qquad (4)\\ &= \begin{cases} p_{\mathcal{N}}\left(\frac{size_{tr}-size_d}{size_{tr}}\right)\cdot p_{\mathcal{N}}(|d-tr|), & \text{if } |\mathbf{v}_{tr}| < \tau_v \\ p_{\mathcal{N}}\left(\frac{size_{tr}-size_d}{size_{tr}}\right)\cdot p_{\mathcal{N}}(dist(d,\mathbf{v}_{tr})), & \text{otherwise} \end{cases} \end{aligned}$$

where $\mathbf{v}_{tr}$ is the velocity of current tracker and $dist(d, \mathbf{v}_{tr})$ is the distance from the detected position to the velocity (distance between point and line). The distribution $p_{\mathcal{N}}(dist(d, \mathbf{v}_{tr}))$ has a shape similar to a 2D cone, which provides the constraint of possible positions of the tracker in the next frame when the current velocity exceeds certain threshold $\tau_v$.

After the pairwise matching scores have been computed for every pair between trackers and detected patches, the determination of association follows a greedy strategy. At every iteration, the algorithm selects the largest matching score $s^*(tr^*, d^*)$ in the current remaining pairs of trackers and detected patches. If $s^*$ is greater than a predefined threshold $\tau$, then this pair of tracker and detected patch $(tr^*, d^*)$ is marked as associated. In order to satisfy the constraint that one tracker (detected patch) can be associated to at most one detected patch (tracker) only, all pairs containing the associated tracker $tr^*$ or detected patch $d^*$ will be removed from the remaining pair set. This procedure continues until there is no pair has a matching score greater than the threshold $\tau$. It is noticeable that by using this greedy strategy, it is not guaranteed that we can get the global optimal solution (e.g., the maximum summation of matching scores) for this pairwise matching problem. However, the result obtained using greedy method is usually acceptable with a much lower computational cost.

Another one of the three important components combined in this tracker is the boosting classifier from [4]. For each tracker, there is an associated classifier. This classifier uses a boosting mode which consists of a series of weak classifiers. The classifier is initialized when the tracker is initialized, based on the information from the first frame. The positive sample is the patch at the current tracker location and the negative samples come from the nearby patches (background). At each frame, after the patch location is updated, this classifier also updates using the most recent information.

The observation model to weigh the particles is based on the output from both the detector and the classifier.

$$w_{tr,p} = \beta\cdot\mathcal{I}(tr)\cdot p_{\mathcal{N}}(p-d^*) + \gamma\cdot d_c(p)\cdot p_0(tr) + \eta\cdot c_{tr}(p) \quad (5)$$

The first two terms in Equation 5 are related to the output of the detector and the third term is obtained from the classifier. $\mathcal{I}(tr)$ is the indicator function which is equal to 1 if there is a detected patch $d^*$ associated to the tracker $tr$ and is equal to 0 otherwise. In the second term, $d_c(p)$ is the confidence produced by the detector at position $p$, which is scaled to $[0, 1]$. $p_0(tr)$ is called *interobject occlusion reasoning*, which is designed for the situation when the detection is failed because of the occlusion. Thus, it is defined as

$$p_0(tr) = \begin{cases} 1, & \text{if } \mathcal{I}(tr) = 1 \\ \max\limits_{tr':\mathcal{I}(tr')=1} p_{\mathcal{N}}(tr - tr'), & \text{else if } \exists\mathcal{I}(tr') = 1 \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

In [2], authors apply two different human detectors: Implicit Shape Model (ISM) [12] and Histogram of Oriented Gradient (HOG) [13]. But in our tracker, we only adopt the HOG detector, because the HOG detector is more generalized and has a more widely usage.

## *B. Crowd Simulator*

The crowd simulation algorithm adopted in the proposed approach is based on the RVO2 library [3]. This model solves an optimization problem based on the strategy named Optimal Reciprocal Collision Avoidance (ORCA). It is computational efficient and only requires the information about the current position and desired velocity of each pedestrian. Due to the page limitation, I will not describe the method in detail. For more details, the reader can refer to the original paper.

RVO2 introduces a concept, namely velocity obstacles, with the definition as

$$VO^T_{A|B} = \{v|\exists t \in [0,T] : v\cdot t \in D(p_B - p_A, r_A + r_B)\} \quad (7)$$

where $p_A$ and $p_B$ are the positions for two pedestrians and $r_A$, $r_B$ are their radii. $D(p, r)$ indicates a circle centered at $p$ with radius $r$. Basically, $VO^T_{A|B}$ defines the set of relative velocities of $A$ with respect to $B$ which will cause a collision in the future within a time period $[0, T]$. Therefore, to find a solution for the collision avoidance equals to find a set of velocities of pedestrians most adjacent to the desired velocities, but meanwhile none of them falls into the velocity obstacle set (that is why it is called "obstacle"). The global optimal solution can be efficiently computed using linear programming. The efficiency enables us to run the simulator multiple times at each frame to get a distribution rather than a single solution.

In a multiple camera tracking system, the position of each pedestrian in the real ground plane is easy to acquire using projection matrix. However, the desired velocity is not explicitly expressed since we have no idea about where a pedestrian will finally move to and how long will this movement takes place. That is, neither the direction nor the speed of the desired velocity can be obtained based on the current information. An alternative way we adopt is to use the historical information. We use an importance sampling strategy to estimate the desired velocity based on its derivatives (accelerations) from the last $m$ frames. Let's define the acceleration set of a pedestrian $k$, $A^t_k = \{a^{t-m}_k, a^{t-m+1}_k, \ldots, a^{t-1}_k\}$, and the corresponding weights for each acceleration

$$w^{t-i}_k = \frac{m-i+1}{\sum_{j=1}^m j}, \qquad i = 1, \ldots, m \quad (8)$$

The weight assigned to more recent acceleration will be larger according to Equation 8. Then following the importance sampling, a set of $n$ accelerations $A'_k = \{a'_{k,1}, a'_{k,2}, \ldots, a'_{k,n}\}$ ($n > m$) can be generated for the estimation of a distribution of desired velocity

$$v'_{k,i} = v_k + a'_{k,i} + \varepsilon_{a,k} \tag{9}$$

where $v'_{k_i}$ is the estimation of velocity corresponding to $a'_{k,i}$, and $v_k$ is the current velocity. $\varepsilon_{a,k}$ is a zero-mean normal distribution with variance $\sigma_{a,k} \propto \max_{i,j} ||a_k^{t-i} - a_k^{t-j}||$. In addition, to handle with the sudden stopping situation (i.e., the velocity suddenly changes to 0), we also randomly add some 0's ($n/5$) into the set of estimated velocities .

Finally, to reduce the computation, not all possible combinations of velocities of pedestrians are calculated using RVO2. Instead, for each pedestrian, we pick up the velocity with the same index (i.e., $i$) and form $n$ sets $C_i = \{v_{1,i}, v_{2,i}, \ldots\}$. So the total number of possible locations calculated for each pedestrian is $n$ in this case.

## C. Global Tracker

The function of the global tracker is to integrate the information from the frame trackers and the output of the crowd simulator to provide a final decision of the current position of each pedestrian in the real ground plane.

The first step we need to do is to recover the projection matrix (a $3\times3$ matrix) for each view. This is done by manually selecting four corresponding points from different views. Then the principal-axis based integration is adopted. A principal axis of a pedestrian is basically the line connecting the pedestrian's head to the feet. In our approach, the principal axis is simply defined as the vertical line in the middle of the patch. If the result obtained from the frame tracker of each view is accurate, then the projection of the principal axis back on the ground plane will intersect at a single point, which is exactly the position of the pedestrian on the ground plane. It is proven that the principal axis-based integration is very robust in fusing the information from different cameras [1], [8].

Therefore, in this proposed approach, the first part of the global tracker is to use the principal axis-based integration to combine the information from different views. For each particle in a frame tracker, we compute the principal axis of the patch associated to it and then project the principal axis back on to the ground plane. Then the intersection points between principal axes from different views are calculated. We also assign a weight for each intersection point

$$w^0_{tr1,p1,tr2,p2} = w_{tr1,p1} \cdot w_{tr2,p2} \tag{10}$$

where $w_{tr1,p1}$ and $w_{tr2,p2}$ are the weights of the particles $p1$ in tracker $tr1$ and $p2$ in tracker $tr2$, calculated by Equation 5. In this case, the intersection of higher weighted principal axes will get a higher weight. To speed up the computation of intersection, not all the possible combinations are tried, instead we experimentally choose $2 * N$ ($N$ is the number of particles in frame tracker) pairs particles between each two different views. The purpose is to decrease the calculation complexity.

The integration of the output provided by the crowd simulator is based on radial basis function. For a particular point $g$, its "simulation term" is defined as

$$sim(g) = \sum_{q \in Q_k} p_{\mathcal{N}}(g - Q_k) \tag{11}$$

where $Q_k$ is the set of possible locations of pedestrian $k$ (its size $|Q_k| = 2N$) and $p_{\mathcal{N}}(g - Q_k)$ is a zero-mean normal distribution. For each intersection point $g_i$, the final weight

$$w'_{g_i} = w^0_{g_i} + \delta sim(g_i) \tag{12}$$

And the final location of each pedestrian on the ground plane is calculated by a weighted sum

$$G = \frac{1}{Z} \sum_i w'_{g_i} g_i \tag{13}$$

where $Z$ is the normalization factor. After the location for each pedestrian on the ground plane is decided, we use the projection matrices $H_v$'s to project this position to different views. And the weights for each particle is reevaluated using a Gaussian filter.

## IV. Experimental Results

In this section, we describe in detail our experimental setting, as well as the results and related discussion. The programming of this algorithm is done in C++ with the widely distributed library OpenCV.

### A. Experimental Setting

Our experiment is conducted on the PETS 2009 dataset with medium density crowd (i.e., S2.L2). This part of the dataset originally has four views. However, according to the provider, View 4 suffers from frame rate instability, so we avoid using this view in our experiment. In addition, View 3 has a large tree on the right part of the field of view, which causes serious occlusions especially with medium density of crowd, and creates a lot of problem even during annotation. Therefore, the frame tracking is performed only based on View 1 and View 2 in our experiments, and we use View 1, 2, and 3 for testing. The image of the ground plane is obtained using Google Maps. The projection matrix for each view is calculated using the function provided in OpenCV, based on the information of a set of four static points.

The ground-truth is obtained by annotation. Each pedestrian is annotated for its bounding box every five frames, and the bounding boxes in between are calculated based on interpolation. With the annotation of bounding box, the principal axis is determined as the vertical line in the middle of the bounding box. The ground-truth of the position of each pedestrian on the ground plane is computed by intersecting the principal axes projected back from View 1 and View 2 (since we cannot guarantee that all the annotations of View 3 are correct because of the occlusion from the big tree).

The HOG detector adopted in our experiment also comes from OpenCV, with the smallest detection size as $48 \times 96$. Therefore, in order to keep the detection result acceptable, we

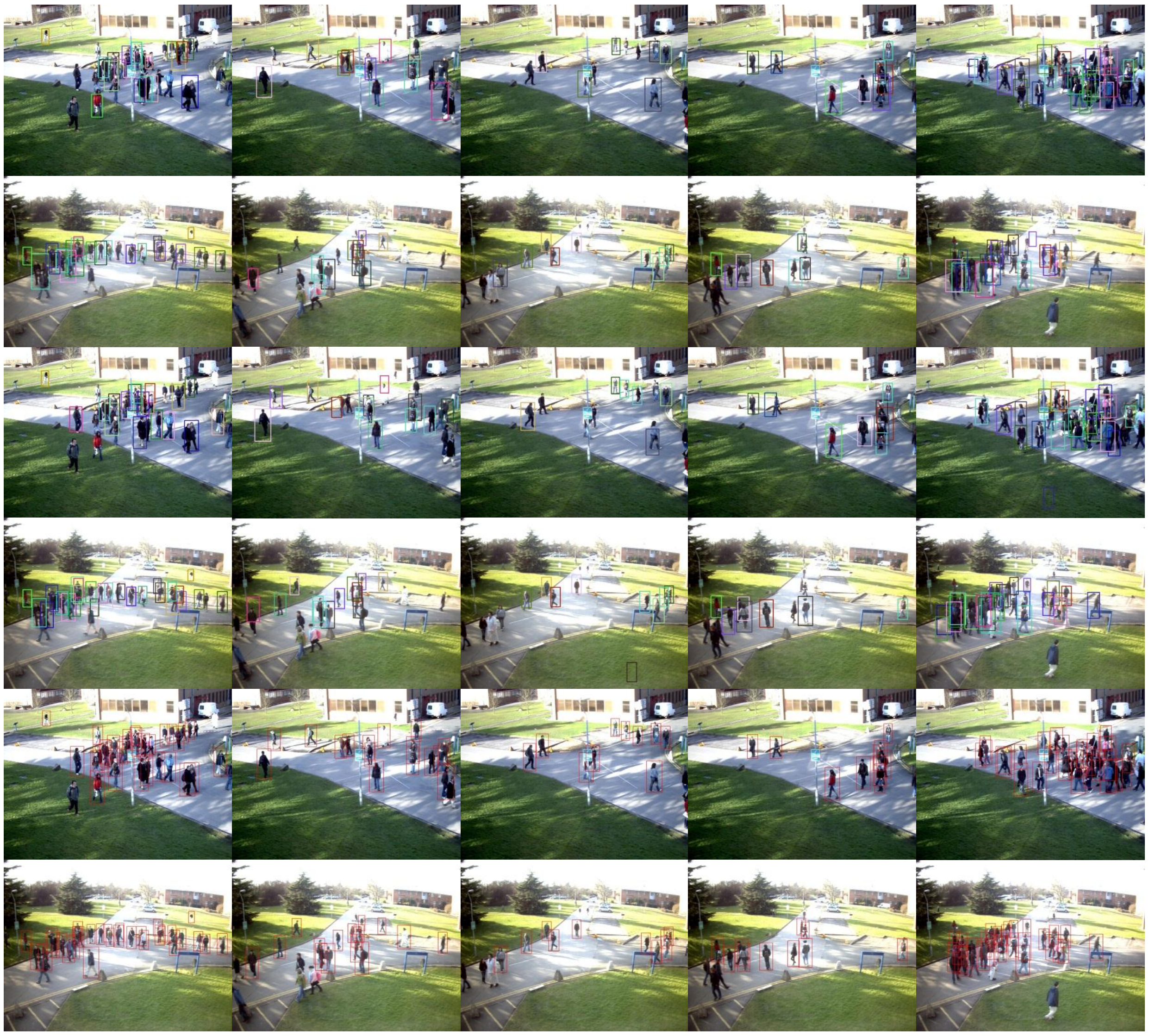

Fig. 2. Some sample images. The top two rows are the results from the tracker with simulation (the two rows are from View 1 and View 2 respectively). The middle two rows are the results without simulation. The bottom are the corresponding ground truth by annotation. Pedestrians who only appear in one view are not shown.

resize the original image frame to $1920 \times 1440$, which makes the smallest pedestrian in the with a similar size as $48 \times 96$.

The parameter setting used in the frame tracker follows the original work [2]. For example, we set $\beta : \gamma : \eta$ in the observation model Equation 5 to 20:2:1. The $\delta$ in Equation 12 is experimentally decided as $2\eta$ (We can also simply set $\delta = 0$ to avoid the influence from the simulator).

For the crowd simulator, RVO2 library, it has 10 parameters for each pedestrian. Among those 10 parameters, three of them keep changing during the runtime (current position, current velocity, and desired velocity) and they are different from individual to individual. The rest seven of them are: the time step of the simulation, the maximal number of neighbors each pedestrian can observe, the maximal speed of a pedestrian, the maximal observation distance of a pedestrian, the radius of a pedestrian, the minimal amount of time a pedestrian is safe with respect to other pedestrians, and the minimal amount of time a pedestrian is safe with respect to static obstacles. Except for the time step of the simulation, the rest of the parameters can actually differ across individuals, but in our experiment we simply make them the same to all individuals. These seven parameters are optimized based on the UCSD crowd dataset. We make the pedestrians in the UCSD and PETS datasets to have the same radius when projected to the ground plane so that the parameters trained on UCSD crowd dataset can be directly applied in the current experiment. However, the UCSD dataset only has one view, so it is not used in the current experiment.

## B. Results

Figure 2 is a qualitative illustration of tracking for View 1 and View 2, comparing the results from the multiple camera tracker with/without crowd simulation. For View 3, because the size of target patch is unknown, we only quantitatively

TABLE I
THE TRACKING ACCURACY WITH/WITHOUT CROWD SIMULATOR FOR THREE VIEWS. THE STATISTICS IS BASED ON ACTIVE TRACKERS.

| View | 1 | 2 | 3 |
|---|---|---|---|
| With simulator | 86.9% | 87.1% | 82.3% |
| Without simulator | 76.4% | 77.7% | 72.5% |

TABLE II
THE MOTP AND MOTA EVALUATION.

| | MOTP | MOTA |
|---|---|---|
| With simulator | 57.1% | 31.9% |
| Without simulator | 59.2% | 9.79% |

calculated the accuracy of tracking. Only the pedestrians appear in both views are evaluated.

For each view, we define the tracking as accurate if the projected point of the position for a pedestrian from the ground plane to the frame falls into a rectangle we called estimated feet area. This rectangle has its height equals to a quarter of the annotation height, its width equals to half of the annotation width, with its location vertically at the bottom and horizontally in the middle of the bounding box. The accuracy of tracking one pedestrian for each view is then calculated as

$$acc_v = \frac{\text{\# accurately tracked frames}}{\text{\# tracked frames}} \tag{14}$$

Table I shows the tracking accuracy with/without crowd simulator for three views.

On the ground plane, we use MOTP (multi-object tracking precision) and MOTA (multi-object tracking accuracy) [14] to measure the performance of the proposed approach, using a point-based distance. Let $r_p$ denote the radius of a pedestrian on the ground plane (which is the same as the radius parameter used in our crowd simulator), $p_{tr}$ and $p_{gt}$ denote the position from the multi-camera tracker and the ground-truth respectively, then the distance used in MOTP and MOTA is

$$s(tr, gt) = \max\left(0, \frac{||p_{tr} - p_{gt}||_2}{r_p} - 1\right) \tag{15}$$

The result of MOTP and MOTA with crowd simulator are 57.1% and 31.9%, respectively. When no information from crowd simulator is integrated (set $\delta$ in Equation 12 to 0), the MOTP and MOTA are 59.2% and 9.79%. The MOTP only evaluates the precision for *matched* objects, and fewer matches between observations and ground truths will possible increase its value, so the crowd simulation does not necessarily improve the performance using this metric. On the other hand, the result of MOTA is not extremely impressive since the scene is so complicated with too many pedestrians in it. However, according to these results as well as the tracking accuracies in each view, we can easily observe that the performance is significantly improved by integrating the crowd simulator.

## V. CONCLUSION

In this paper, we proposed a multi-camera tracking approach that combines *state-of-the-art* single camera tracking-by-detection method, RVO2 crowd simulation method and the principal axis-based integration between cameras. The purpose of this paper is to investigate the potential performance improvement of tracking when we integrate the crowd simulation information into traditional pure vision-based tracking approaches. The experiments are conducted on PETS 2009 dataset with medium density crowd. Tracking performance is evaluated for different views. Tracking precision and accuracy (MOTP and MOTA) are calculated based on the position of each pedestrian on the real ground plane. The experimental results show that the multi-camera tracking with crowd simulator integrated significantly outperforms the one without crowd simulator, which means that the utilization of the relationship between pedestrian positions is really important and helpful in tracking.

## ACKNOWLEDGMENT

This work was supported in part by NSF grant 0905671.

## REFERENCES

[1] W. Hu, M. Hu, X. Zhou, T. Tan, J. Lou, and S. Maybank, "Principal axis-based correspondence between multiple cameras for people tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 663–671, Apr. 2006.

[2] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1820–1833, sept. 2011.

[3] J. van den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *14th International Symposium on Robotics Research*, Sep. 2009.

[4] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, ser. CVPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 260–267.

[5] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proceedings of the 10th European Conference on Computer Vision: Part I*, ser. ECCV '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 234–247.

[6] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, Aug. 2011.

[7] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Computer Vision and Pattern Recognition, 2008*, june 2008, pp. 1 –8.

[8] W. Du and J. Piater, "Multi-camera people tracking by collaborative particle filters and principal axis-based integration," in *Proceedings of the 8th Asian conference on Computer vision - Volume Part I*, ser. ACCV'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 365–374.

[9] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, p. 487, 2000.

[10] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," in *ACM SIGGRAPH 2010 papers*, ser. SIGGRAPH '10. New York, NY, USA: ACM, 2010, pp. 123:1–123:9.

[11] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," in *ACM SIGGRAPH*, 2006, pp. 1160–1168.

[12] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *Int. J. Comput. Vision*, vol. 77, no. 1-3, pp. 259–289, May 2008.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 886–893.

[14] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *J. Image Video Process.*, vol. 2008, pp. 1:1–1:10, Jan. 2008.