Leveraging Long-Term Predictions and Online-Learning in Agent-based Multiple Person Tracking

Wenxi Liu, Antoni B. Chan, Rynson W. H. Lau, *Senior Member, IEEE,*
and Dinesh Manocha, *Fellow, IEEE*

***Abstract*—We present a multiple-person tracking algorithm, based on combining particle filters and RVO, an agent-based crowd model that infers collision-free velocities so as to predict pedestrian's motion. In addition to position and velocity, our tracking algorithm can estimate the internal goals (desired destination or desired velocity) of the tracked pedestrian in an online manner, thus removing the need to specify this information beforehand. Furthermore, we leverage the longer-term predictions of RVO by deriving a higher-order particle filter, which aggregates multiple predictions from different prior time steps. This yields a tracker that can recover from short-term occlusions and spurious noise in the appearance model. Experimental results show that our tracking algorithm is suitable for predicting pedestrians' behaviors online without needing scene priors or hand-annotated goal information, and improves tracking in real-world crowded scenes under low frame rates.**

***Index Terms*—Pedestrian tracking, video surveillance, particle filter, pedestrian motion model.**

I. Introduction

Pedestrian tracking from videos of indoor and outdoor scenes remains a challenging task despite recent advances. It is especially difficult in crowded real-world scenarios because of practical problems such as inconsistent illumination conditions, partial occlusions and dynamically changing visual appearances of the pedestrians. For applications such as video surveillance, the video sequence usually has relatively low resolution or low frame rate, and may thus fail to provide sufficient information for the appearance-based tracking methods to work effectively.

One method for improving the robustness of pedestrian tracking w.r.t. these confounding factors is to use an accurate motion model to predict the next location of each pedestrian in the scene. In the real-world, each individual person plans its path and determines its trajectory based on both its own internal goal (e.g., its final destination), and the locations of any surrounding people (e.g., to avoid collisions, or to remain in a group). Many recent methods [30, 38] have proposed online tracking using agent-based crowd simulators, which treat each person as a separate entity that independently plans its own path.

Despite their successes, agent-based online trackers [30, 38] have several shortcomings. First, they assume that the goal position or destination information of each pedestrian in the scene is known in advance, either hand annotated or estimated off-line using training data. Second, these tracking methods only use single-step prediction, which is a short-term prediction of the pedestrian's next location based on its current position. As a result, these trackers are not fully leveraging the capabilities of agent-based motion models to predict *longer-term* trajectories, e.g., steering around obstacles. Third, existing motion models often overlook subtle, but crucial, factors in crowd interactions, including anticipatory motion and smooth trajectories [15]; this avoidance is *reciprocal*, as pedestrians mutually adjust their movements to avoid anticipated collisions. One of our goals is to use the long-term predictive capacities of such motion models to improve pedestrian tracking.

In this paper, we address these shortcomings by proposing a novel multiple-person tracking algorithm, based on a higher-ordered particle filter and an agent-based motion model using reciprocal velocity obstacles (RVO). RVO [35], a multi-agent collision avoidance scheme, has been used for multi-robot motion planning and generating trajectories of synthetic agents in dense crowds. In this paper, we combine the RVO motion model with particle filter tracking, resulting in a flexible multi-modal tracker that can estimate both the physical properties (position and velocity) and individual internal goals (desired destination or desired velocity). This estimation is performed in an online manner without acquiring scene prior or hand-annotated goal information. Leveraging the longer-term predictions of RVO, we derive a higher-ordered particle filter (HPF) based on a higher-order Markov assumption. The HPF constructs a posterior that accumulates multiple predictions of the current location based on multiple prior time steps. Because these predictions from the past ignore subsequent observations, the HPF tracker is better able to

recover from spurious failures of the appearance model and short-term occlusions that are common in multi-pedestrian tracking.

We evaluate the efficacy of our proposed tracking algorithm on low frame rate video of real scenes, and show that our approach can estimate internal goals online, and improve trajectory prediction and pedestrian tracking over prior methods. In summary, the major contributions of this paper are three-fold:

- We propose a novel tracking method that integrates agent-based crowd models and the particle filter, which can reason about pedestrians' motion and interaction and adaptively estimate internal goals.
- We propose a novel higher-order particle filter, based on a higher-order Markov model, to fully leverage the longer-term predictions of RVO.
- We demonstrate that our RVO-based online tracking algorithm improves tracking in low frame rate crowd video, compared to other motion models, while not requiring scene priors or hand-annotated goal information.

II. Related works

Multi-object tracking has been an active research topic in recent years. It is a challenging problem due to the changing appearances caused by factors including pose, illumination, and occlusion. [40] gives a general introduction to object tracking. Despite recent advances in tracking [4, 16], appearance-based approaches struggle to handle large numbers of targets in crowded scenes.

Data association: Recent data association approaches for multi-object tracking [2, 6, 25] usually associate detected responses into longer tracks, which can be formulated as a global optimization problem, e.g., by computing optimal network flow [10, 41] or using linear programming [8, 19]. In addition, the mutual relation among targets is used to refine the data association for multiple targets [2, 22, 37]. Data association methods usually require detection responses from the past and the future frames, while ours is an online tracking method.

Particle filters: Particle filters (PF) are often used for online multi-object tracking [5, 9, 18, 22, 27, 31, 36]. To handle the high-dimensionality in the joint state space when tracking multiple targets, [22] proposes a PF that uses a Markov random field (MRF) to penalize the proximity of particles from different objects. [18] describes a multi-target tracking framework based on pseudo-independent log-linear PFs and an algorithm to discriminatively train filter parameters. [9] uses a detection-based PF framework with a greedy strategy to associate the pedestrians' detection responses from consecutive frames. In contrast to these approaches, we focus on using PF to adaptively estimate the agents' internal goals (e.g., desired location), as well as the position and velocity. To enhance the performance of the traditional PF, we leverage crowd motion models to provide more accurate and longer-term state transition.

Higher-order models: Previous works on "higher-order" Bayesian models related to tracking include: [13], which embeds multiple previous states into a large vector for state transition; [29], which is conditioned on multiple previous observations; [28], which formulates the state transition as a unimodal Gaussian distribution with the mean as a weighted sum of predictions from previous particles. Refer to Sec. IV-B for detailed comparisons with our HPF.

Crowd motion models: Crowd tracking methods use complex motion models to improve tracking large crowds [1, 23, 32, 34, 39]. E.g., [1] uses a floor-fields to generate the probability of targets' moving to their nearby regions in extremely crowded scenes, while [39] learns the non-linear motion patterns to assist association of tracklets from detection responses. Related to our work are trackers using agent-based motion models. E.g., [3] discretizes the velocity space of a pedestrian into multiple areas and models the probability of choosing velocities for tracking, while [30] formulates pedestrians' movements as an energy optimization problem that factors in navigation and collision avoidance. [38] obtains insights from the social force approach, by using both group and destination priors in the pedestrian motion model. Pedestrian motion is also studied in other fields, e.g. pedestrian dynamics, graphics and robotics [15, 17]. Recently, RVO [35] has also been applied to tracking [20].

With these previous approaches [3, 20, 30, 38], agent-based motion model is simply used to predict one step ahead for the state transitions, regardless of its longer-term predictive capabilities. In contrast, our focus is to leverage the capabilities of agent-based motion models, by using longer-term predictions and online estimation of the agent's internal goals without scene priors.

III. Tracking with particle filters and RVO

In this section, we first present the RVO crowd model, and integrate it into a multiple-target tracking algorithm using the particle filter.

A. Reciprocal velocity obstacle (RVO)

The RVO motion model [35] predicts agents' positions and velocities in a 2D ground space, given the information of all agents at the current time step. The predicted agents' positions and velocities are such that they will not lead to collision among the agents. Each agent is first simplified as a disc in 2D space. The formulation of RVO then consists of several parameters, which are used to manipulate each agent's movement, including desired velocity and the radius of agent's disc.

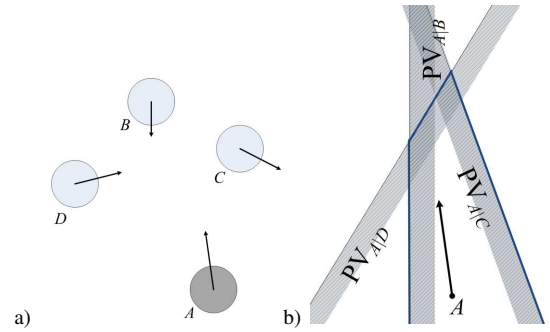


Fig. 1: The RVO multi-agent simulation. (a) shows a scenario of four agents in a 2D space, with velocities indicated by arrows. (b) shows the corresponding velocity space. The shaded side of each plane indicates the set of permitted velocities for agent A that avoid collision with each of other agents. The region with bolded lines denotes the set of velocities of agent A that lead to no collisions.

RVO is built on the concept of velocity obstacle (VO) [14]. Let $D(\mathbf{p}, r)$ denote an open disc of radius r centered at a 2D position $\mathbf{p}$:

$$D(\mathbf{p}, r) = \{\mathbf{q} \mid\mid \mathbf{q} - \mathbf{p} \mid\mid < r\}. \quad (1)$$

Given a time horizon τ, we have:

$$VO^{\tau}_{A|B} = \{\mathbf{v} \mid \exists t \in [0, \tau], t\mathbf{v} \in D(\mathbf{p}_B - \mathbf{p}_A, r_A + r_B)\}. \quad (2)$$

Here $VO^{\tau}_{A|B}$ refers to the set of velocities that will lead to the collision of agents A and B. Hence, if agent A chooses a velocity that is outside of $VO^{\tau}_{A|B}$, A is guaranteed not to collide with B for at least τ seconds, where τ is the planning horizon. RVO sets up linear constraints in the velocity space and ensures that A's new velocity will be outside VO^{τ}. Let $\mathbf{u}$ be the smallest change required in the relative velocity of A towards B in order to avoid collision. Let $\mathbf{v}^{opt}_A$ and $\mathbf{v}^{opt}_B$ be the velocities of agents A and B, respectively, at the current time step. $\mathbf{u}$ can be geometrically interpreted as the vector going from the current relative velocity to the closest point on the VO boundary,

$$\mathbf{u} = (\underset{\mathbf{v} \in VO^{\tau}_{A|B}}{\operatorname{argmin}} \mid\mid \mathbf{v} - (\mathbf{v}^{opt}_A - \mathbf{v}^{opt}_B) \mid\mid) - (\mathbf{v}^{opt}_A - \mathbf{v}^{opt}_B). \quad (3)$$

If both agents A and B are implicitly sharing the responsibility to avoid collision, each needs to change their velocity by at least $\frac{1}{2}\mathbf{u}$, expecting the other agent to take care of the other half. Therefore, the set of velocities permitted for agent A towards agent B is:

$$PV^{\tau}_{A|B} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}^{opt}_A + \frac{1}{2}\mathbf{u})) \cdot \mathbf{n} \geq 0\}, \quad (4)$$

where $\mathbf{n}$ is the normal of the closest point on $VO^{\tau}_{A|B}$ to maximize the allowed velocities. In a multi-agent planning situation, this set of permitted velocities for agent A is computed as,

$$PV^{\tau}_{A} = D(\mathbf{0}, v^{max}_A) \cap \bigcap_{B \neq A} PV^{\tau}_{A|B}, \quad (5)$$

where v^{max}_A is the velocity with the maximum speed that agent A can take. Thus, A's collision-free velocity and position at the next time step can be computed as,

$$\mathbf{v}_{t+1} = \underset{\mathbf{v} \in PV}{\operatorname{argmin}} \mid\mid \mathbf{v} - \mathbf{v}_{desire}, \mid\mid \quad (6)$$

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_{t+1} \triangle t, \quad (7)$$

where $\triangle t$ is the time interval between steps and $\mathbf{v}_{desire}$ is the desired velocity. This leads to a convex optimization solution. Figure 1 illustrates an example of RVO. More details can be found in [35].

B. Tracking with particle filters

In our tracking approach, we use an independent particle filter for each person, similar to [18], to track the state of the person. Each particle filter only estimates the state of a single person, but it can access the previous state estimates of the other pedestrians to infer the velocity using RVO. Hence the particle filters are loosely coupled throughout the tracking and simulation.

The state representation of a person at time t contains the person's position, velocity, and desired velocity, i.e., $\mathbf{x}_t = [\mathbf{p}^T_t, \mathbf{v}^T_t, \mathbf{v}^T_{desire}]^T$. While $\mathbf{p}_t$ and $\mathbf{v}_t$ determine the physical properties of the person, $\mathbf{v}_{desire}$ represents the intrinsic goal of the person. In contrast to previous tracking methods using crowd models [30, 38], our method automatically performs *online estimation* of the intrinsic properties (desired location or desired velocity) during tracking.

In the standard online Bayesian tracking framework, the propagation of state $\mathbf{x}_t$ at time t depends only on the previous state (first-order Markov assumption). The goal of the online Bayesian filter is to estimate the posterior distribution of state $\mathbf{x}_t$, given the current observation sequence $\mathbf{y}_{1:t}$,

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \alpha p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \quad (8)$$

where $\mathbf{y}_t$ is the observation at time t, $\mathbf{y}_{1:t}$ is the set of all observations through time t, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the state transition distribution, $p(\mathbf{y}_t|\mathbf{x}_t)$ is the observation likelihood distribution, and α is the normalization constant. As the integral does not have a closed-form solution in general, the particle filter [12] approximates the posterior using a set of weighted samples $\mathcal{X}_t = \{(w^{[m]}_t, \mathbf{x}^{[m]}_t)\}_{m=1:M}$, where each $\mathbf{x}^{[m]}_t$ is an instantiation of the process state, known as a particle, and $w^{[m]}_t$ is the corresponding weight. Under this representation, the Monte Carlo approximation can be given as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \alpha\, p(\mathbf{y}_t|\mathbf{x}_t) \sum_{m=1}^{M} w^{[m]}_{t-1} p(\mathbf{x}_t|\mathbf{x}^{[m]}_{t-1}), \quad (9)$$

where M is the number of particles.

For the transition density, we model the propagation of the velocity and position as the RVO prediction with additive Gaussian noise, and use a simple diffusion

process for the desired velocity. Thus, the transition density is

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}(\begin{bmatrix} \mathbf{p}_t + \mathcal{R}(\mathbf{X}_t)\Delta t \\ \mathcal{R}(\mathbf{X}_t) \\ \mathbf{v}_{desire} \end{bmatrix}, \Gamma), \tag{10}$$

where we represent the process of computing the predicted velocity using RVO in Eq. 6 as $\mathcal{R}(\mathbf{X}_t)$. $\mathbf{X}_t$ is the set of posterior means for all agents at time t, and Γ is a diagonal covariance matrix of the noise.

IV. Tracking with higher-order particle filters

Due to the modeling of person-to-person reciprocal interactions, RVO can reliably predict the crowd configuration over longer times. However, the particle filter from Section III-B uses a first-order Markov assumption, and hence the crowd model only predicts ahead one time step. Aiming to take better advantage of the longer-term predictive capabilities of RVO, in this section, we derive a higher-order particle filter (HPF) that uses multiple predictions, starting from past states.

A. Higher-order particle filter

In the HPF, we assume that the state process evolves according to a Kth-order Markov model, with transition distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1:t-K})$, which is a mixture of individual transitions from each previous state $\mathbf{x}_{t-j}$,

$$p(\mathbf{x}_t|\mathbf{x}_{t-1:t-K}) = \sum_{j=1}^{K} \pi_j p_j(\mathbf{x}_t|\mathbf{x}_{t-j}), \tag{11}$$

where π_j are mixture weights, $\sum_j \pi_j = 1$, and $p_j(\mathbf{x}_t|\mathbf{x}_{t-j})$ is the j-step ahead transition distribution, i.e., the prediction from previous state $\mathbf{x}_{t-j}$ to time t. For the Kth-order model, the predictive distribution is

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) &= \int p(\mathbf{x}_t|\mathbf{x}_{t-1:t-K}) p(\mathbf{x}_{t-1:t-K}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1:t-K} \\ &= \int \left[\sum_{j=1}^{K} \pi_j p_j(\mathbf{x}_t|\mathbf{x}_{t-j}) \right] p(\mathbf{x}_{t-1:t-K}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1:t-K} \\ &= \sum_{j=1}^{K} \pi_j \int p_j(\mathbf{x}_t|\mathbf{x}_{t-j}) p(\mathbf{x}_{t-j}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-j}, \end{aligned} \tag{12}$$

where Eq. 12 follows from swapping the summation and integral, and marginalizing out the state variables $\mathbf{x}_{t-k}$, for $k \neq j$. Note that the posterior of the jth previous state $p(\mathbf{x}_{t-j}|\mathbf{y}_{1:t-1})$ depends on some "future" observations $\{\mathbf{y}_{t-j+1}, \cdots, \mathbf{y}_{t-1}\}$. As we want to leverage the longer-term predictive capabilities of our crowd model, we assume that these "future" observations are unseen, and hence $p(\mathbf{x}_{t-j}|\mathbf{y}_{1:t-1}) \approx p(\mathbf{x}_{t-j}|\mathbf{y}_{1:t-j})$. The motivation for the approximation is two-fold: 1) it makes HPF completely online, by avoiding backtracking needed for smoothing PFs; 2) it reduces the effects of bad observations (e.g. if occlusion occurs at t-1, predictions from t-2 can jump over it). Substituting it into Eq. 12 yields an approximate predictive distribution as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \approx \sum_{j=1}^{K} \pi_j p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}), \tag{13}$$

which is a weighted sum of individual j-step ahead predictive distributions from the state posterior at time $t-j$,

$$p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}) = \int p_j(\mathbf{x}_t|\mathbf{x}_{t-j}) p(\mathbf{x}_{t-j}|\mathbf{y}_{1:t-j}) d\mathbf{x}_{t-j}. \tag{14}$$

The j-step ahead predictive distribution is approximated with particles from time $t-j$,

$$p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}) \approx \sum_m p_j(\mathbf{x}_t|\mathbf{x}_{t-j}^{[m]}) p(\mathbf{x}_{t-j}^{[m]}|\mathbf{y}_{1:t-j}). \tag{15}$$

Associated with the j-step ahead predictor is a corresponding posterior distribution, which is further conditioned on the current observation $\mathbf{y}_t$,

$$p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}, \mathbf{y}_t) = \frac{p(\mathbf{y}_t|\mathbf{x}_t) p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j})}{p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j})} \tag{16}$$

$$\approx \tfrac{1}{\alpha_j} p(\mathbf{y}_t|\mathbf{x}_t) \sum_m p_j(\mathbf{x}_t|\mathbf{x}_{t-j}^{[m]}) p(\mathbf{x}_{t-j}^{[m]}|\mathbf{y}_{1:t-j}), \tag{17}$$

where $\alpha_j = p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j})$ is the likelihood of observing $\mathbf{y}_t$ using the j-step ahead predictor,

$$p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j}) = \int p(\mathbf{y}_t|\mathbf{x}_t) p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}) d\mathbf{x}_t \tag{18}$$

$$\approx \sum_m p(\mathbf{y}_t|\mathbf{x}_t^{[m]}) p_j(\mathbf{x}_t^{[m]}|\mathbf{y}_{1:t-j}). \tag{19}$$

The posterior distribution can now be approximated as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{\int p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) d\mathbf{x}_t} \tag{20}$$

$$\approx \frac{p(\mathbf{y}_t|\mathbf{x}_t) \sum_j \pi_j p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j})}{\int p(\mathbf{y}_t|\mathbf{x}_t) \sum_j \pi_j p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}) d\mathbf{x}_t} \tag{21}$$

$$= \frac{\sum_j \pi_j p(\mathbf{y}_t|\mathbf{x}_t) p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j})}{\sum_j \pi_j \int p(\mathbf{y}_t|\mathbf{x}_t) p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}) d\mathbf{x}_t} \tag{22}$$

$$= \frac{\sum_j \pi_j \frac{p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j})}{p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j})} p(\mathbf{y}_t|\mathbf{x}_t) p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j})}{\sum_j \pi_j p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j})} \tag{23}$$

$$= \frac{\sum_j \pi_j p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j}) p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}, \mathbf{y}_t)}{\sum_j \pi_j p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j})} \tag{24}$$

Hence, the approximate posterior distribution is a weighted sum of the j-step ahead posteriors,

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_j^K \lambda_j p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}, \mathbf{y}_t), \tag{25}$$

where the weight for each individual posterior is

$$\lambda_j = \frac{\pi_j p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j})}{\sum_j \pi_j p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j})}. \tag{26}$$

Finally, substituting in the particle filter

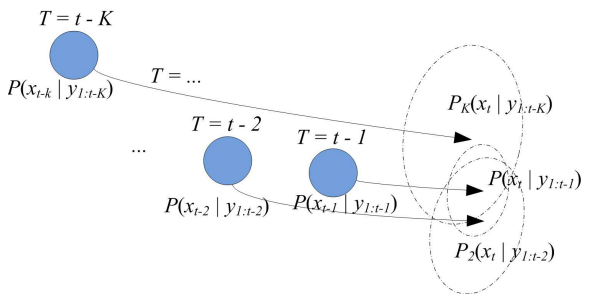


Fig. 2: The basic principle of the higher-order particle filter (HPF). Each circle represents the posterior of the previous state at time steps $t-1$, $t-2$, ..., $t-K$. Each dashed ellipse is the predictive distribution evolved from the posterior of the previous state. The resulting posterior of the current time step is the weighted sum of these predictive distributions (Eq. 25).

approximation,

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto \sum_j^K \lambda_j p(\mathbf{y}_t|\mathbf{x}_t) \sum_m p_j(\mathbf{x}_t|\mathbf{x}_{t-j}^{[m]}) p(\mathbf{x}_{t-j}^{[m]}|\mathbf{y}_{1:t-j}), \tag{27}$$

$$p_j(\mathbf{y}_t|\mathbf{y}_{1:t-j}) = \int p(\mathbf{y}_t|\mathbf{x}_t) p_j(\mathbf{x}_t|\mathbf{y}_{1:t-j}) d\mathbf{x}_t \tag{28}$$

$$\approx \sum_m p(\mathbf{y}_t|\mathbf{x}_t^{[m]}) p_j(\mathbf{x}_t^{[m]}|\mathbf{y}_{1:t-j}). \tag{29}$$

In summary, the posterior of HPF is a weighted sum of standard particle filters, where each filter computes a j-step ahead prediction from time $t-j$ to t. The weights are proportional to the likelihood of the current observation for each filter, and hence, some modes of the posterior will be discounted if they are not well explained by the current observation. The overview of HPF is illustrated in Figure 2 and the pseudo-code is in Algorithm 1.

Note that HPF is computationally efficient, as we may reuse the previous particles (i.e., $\mathbf{x}_t^{[j,m]}$ in Algorithm 1) and the j-step ahead predictions can be computed recursively from the previous predictions.

B. Comparison with previous higher-order models

Different from previous methods, our HPF formulation assumes the state transition a mixture of individual transitions from multiple prior time steps. Felsberg et al. [13] embed multiple previous states into a vector, resulting in a transition density that is a product of individual transitions terms, while ours is a mixture distribution. Park et al. [29] use an appearance model conditioned on multiple previous observations, while we use a first-order appearance model. Both [13, 29] are not based on PFs. Our work is closely related to [28], which formulates a higher-order PF where the transition model is a unimodal Gaussian distribution with the mean as a weighted sum of predictions from previous particles. In [28], the influence of previous states depends on their trained transition weights rather than the observation likelihood, and each particle depends only on its parent particles. In contrast, our higher-order PF models the state transition as a multimodal mixture distribution and the importance weights rely on both observations and transition weights (Eq. 26). Specifically, previous states with high likelihood in predicting the current observation impose more effects on the posterior, which is useful for tracking through occlusion. Moreover, all particles of the previous states are used to form the posterior, which increases diversity of particles (Algorithm 1).

Algorithm 1 The Higher-Order Particle Filter.

$\mathcal{X}_t = \overline{\mathcal{X}}_t = \emptyset$.
for all time step $t \in \{1...T\}$ **do**
  for all $j \in \{1, ..., K\}$ **do**
    {* calculate individual posterior from time $t-j$ *}
    for all $m \in \{1...M\}$ **do**
      $\mathbf{v}_t = \mathcal{R}(\mathbf{X}_{t-j})$. {* RVO prediction *}
      $\overline{\mathbf{x}}_t = [(\mathbf{p}_{t-j} + \mathbf{v}_t \triangle t)^T, \mathbf{v}_t^T, \mathbf{v}_{desire}^T]^T$.
      $p(\mathbf{x}_t|\mathbf{x}_{t-j}) \sim \mathcal{N}(\overline{\mathbf{x}}_t, \Gamma)$.
      $\mathbf{x}_t^{[j,m]} \sim p(\mathbf{x}_t|\mathbf{x}_{t-j}^{[m]})$. {* update particle *}
      $w_t^{[j,m]} = w_{t-j}^{[m]} p(\mathbf{y}_t|\mathbf{x}_t^{[j,m]})$. {* update weight *}
    end for
  end for
  {* calculate posterior *}
  for all $j \in \{1, ..., K\}$ **do**
    $\lambda_j = \text{Normalize}(\pi_j \sum_{m=1}^M w_t^{[j,m]})$.
    $w_t^{[j,1:M]} = \text{Normalize}(w_t^{[j,1:M]})$.
    for all $m \in \{1...M\}$ **do**
      $w_t^{[j,m]} = \lambda_j w_t^{[j,m]}$.
      $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t \cup \{(w_t^{[j,m]}, \mathbf{x}_t^{[j,m]})\}$.
    end for
  end for
  {* resample and select most weighted M particles *}
  $\mathcal{X}_t = \text{Resample}(\overline{\mathcal{X}}_t, M)$.
end for
NOTE: $\mathcal{R}(\mathbf{X}_{t-j})$ predicts the velocity at time t from the state $\mathbf{x}_{t-j}$ by recursively applying a single step of RVO j times.

V. Experiments

We evaluate our tracking algorithm in two experiments: 1) prediction of pedestrians' motion from ground-truth trajectories; 2) multi-pedestrian tracking in video with low fps.

A. Dataset and models

We use the same crowd datasets used in [30, 38] to study real-world interactions among pedestrians and evaluate motion models. We use the *Hotel* sequences from [30] and the *Zara01*, *Zara02* and *Student* sequences from [24]. All sequences are captured at 2.5 fps and annotated at 0.4s intervals. *Hotel* is captured from an

aerial view, while *Zara01* and *Zara02* are both side views, and *Student* is captured from an oblique view. All of them include adequate interactions of pedestrians (e.g. collision avoidance).

In our framework, we transform pedestrians feet positions in the image to coordinates in the ground space, as is the case in many recent tracking papers [2, 30, 38]; pedestrians motion can be estimated and described using pedestrian motion model such as RVO and LTA for the sake of accuracy. Tracking approaches using ground-space information become popular due to advances in estimating the transformation between the world space and the image space, e.g. Fig.7 in [30] uses a moving camera at eye-level. Here the geometric transformation is enough for static camera. Our datasets follow [30, 38], which are all general surveillance videos. Except Hotel, these datasets are captured with oblique views.

In addition to RVO, we also consider three other agent-based motion models, LTA [30], ATTR and ATTRG [38]. LTA and ATTR predict the pedestrians' velocities by optimizing energy functions, which contain terms for collision avoidance, attraction towards the goals and desired speed. ATTRG adds a pedestrian grouping prior to ATTR. The source codes are acquired from the authors of [30, 38]. We also consider the baseline constant velocity model (LIN).

In order to make a direct comparison between motion models in the same framework, we combine each of these motion models with the identical particle filter (PF), described in Sec. III-B, by replacing RVO with others. The state representation of the PF includes the agent's physical properties (position and velocity) and internal goals (desired velocity), as in Sec. III-B. The desired velocity is initially set to the initial velocity and evolves independently as the tracker runs. These online adaptive models are denoted as LTA+, ATTR+, ATTRG+, and RVO+. We also test a version of the PF where the desired velocity does not evolve along with position and velocity, and is fixed as the initial velocity (denoted as LIN, LTA, ATTR, ATTRG, and RVO).

In addition to PF, we also test RVO+ with our proposed HPF (Sec. IV). We compare against the higher-order particle filter from [28] (denoted as pHPF), which models the transition density as Gaussian where the mean is the weighted sum of predictions from previous particles. We use both RVO and LIN with pHPF. The higher-order models of [13, 29] are not based on PFs, so we do not consider them in experiments.

Parameter selection: As in [30, 38], the parameters of LTA, ATTR and RVO are trained using the genetic algorithm on an independent dataset *ETH*. The hyper-parameters of HPF (K, π_j) and pHPF (weights from each past particle) are also trained on *ETH* from [30]. For HPF, the resulting settings are $K = 2^1$, $\pi_j = \{0.91, 0.09\}^2$. Outside of this training, no scene prior or destination information is given, since we assume that the tracker can estimate these online.

TABLE I: Mean prediction errors of different motion models. The bold numbers are the best in each column.

	Zara01			*Zara02*			*Student*			
	L=5	L=15	L=30	L=5	L=15	L=30	L=5	L=15	L=30	avg.
LIN	0.35	0.69	0.78	0.38	0.79	0.89	0.39	0.84	1.00	0.68
LTA	0.34	0.64	0.72	0.36	0.70	0.80	0.42	0.84	0.98	0.64
LTA+	0.37	0.73	0.82	0.37	0.77	0.89	0.42	0.91	1.08	0.71
ATTR	0.43	0.85	0.94	0.41	0.80	0.92	0.55	1.06	1.22	0.80
ATTR+	0.35	0.69	0.77	0.36	0.72	0.82	0.42	0.84	0.98	0.66
ATTRG	0.42	0.80	0.88	0.41	0.81	0.93	0.55	1.06	1.22	0.79
ATTRG+	0.36	0.67	0.74	0.36	0.70	0.80	0.46	0.88	1.02	0.67
RVO	0.34	0.66	0.74	0.34	0.72	0.84	0.39	0.83	0.98	0.65
RVO+	0.31	0.61	0.69	0.34	0.69	0.80	0.39	0.82	0.97	0.62
pHPF (LIN)	0.41	0.79	0.88	0.43	0.86	1.00	0.46	0.95	1.11	0.77
pHPF(RVO+)	0.37	0.68	0.76	0.38	0.76	0.88	0.44	0.90	1.05	0.69
HPF (RVO+)	**0.25**	**0.52**	**0.60**	**0.26**	**0.61**	**0.69**	**0.31**	**0.72**	**0.86**	**0.54**

B. Online estimation and motion prediction

In the first experiment, we evaluate the motion models mentioned in Sec. V-A and online estimation of desired velocities on a ground-truth prediction problem. For each dataset, the following two-phase experiment starts at every 16th time-step of the sequence. (*Learning*) the PF iterates using the ground-truth positions as observations for 10 consecutive time steps; (*Prediction*) after 10 time steps, the underlying motion model predicts (without further observations) the person's trajectory for at most 30 time steps. We evaluate the accuracy of the motion models' predictions using mean errors, i.e., distances from ground-truth, averaged over tracks in each dataset.

Table I reports the errors of predicting $L = \{5, 15, 30\}$ time steps on each dataset, as well as the average error over all datasets. Looking at the prediction performance without online estimation of desired velocities, LTA and RVO have similar error (0.64/0.65). However, due to the absence of the goals, ATTR and ATTRG do not perform as well. With the desired velocities adapted, the prediction errors decrease for most motion models (ATTR+, ATTRG+, and RVO+). This suggests that these crowd motion models rely on online estimating goals to produce accurate predictions. Since the behavior prediction of LTA relies on the velocity at the previous time step, the desired velocity estimation (LTA+) does not improve prediction. Among motion models, RVO+ has the lowest prediction error (0.62). Finally, using

[1]Since we test on low fps (2.5fps) data, compared with normal fps (25fps), the locations and velocities of pedestrians may change significantly between 3 frames. From our experiments, $K = 2$ turns out to be a reasonable longer-term duration to predict ahead.

[2]The ratio between π_1 and π_2 suggests that the prediction from 2-steps ahead should have more than 10x higher likelihood in order to override the prediction from 1-step ahead (Eq. 26)

Fig. 3: Prediction on *Zara01* and *Student*. The markers indicate the points used in the learning phase, while the solid line is the subsequent prediction. The predicted trajectories of RVO+ and HPF are adapted through the learning phase, and are closest to the ground-truth.

the HPF with RVO+ improves the prediction over the standard PF (RVO+) for all prediction lengths, with the average error reduced to 0.54 (about $\sim$15% lower). Figure 3 shows two examples comparing RVO+ with related models.

C. Multiple-person tracking

We have evaluated our framework for tracking pedestrians in videos with low frame rate.

Setup. The observation likelihood is modeled as $p(\mathbf{y}_t|\mathbf{x}_t) \propto \mathrm{HSV}(\mathbf{p}_t, \mathbf{y}_t)\mathrm{Detect}(\mathbf{p}_t, \mathbf{y}_t)$. The first term, $\mathrm{HSV}(\mathbf{p}_t, \mathbf{y}_t)$, measures the color histogram similarity between the observed template and the appearance template centered at location $\mathbf{p}_t$ in frame $\mathbf{y}_t$, i.e., $\mathrm{HSV}(\mathbf{p}_t, \mathbf{y}_t) \propto \exp(-B(\mathbf{p}_t, \mathbf{y}_t)^2/2\sigma^2)$, where $B(\mathbf{p}_t, \mathbf{y}_t)$ measures the Bhattacharyya distance between the observed template and the appearance template and σ is the variance paramter. The second term, $\mathrm{Detect}(\mathbf{p}_t, \mathbf{y}_t)$, is the detection score at $\mathbf{p}_t$ of $\mathbf{y}_t$ from a HOG-based head-shoulders detector [11][3].

We use the same experimental protocol described in [38] for 2.5 fps video. The tracker starts at every 16th frame and runs for at most 24 subsequent frames, as long as the ground truth data exists for those frames in the scene. The tracker is initialized with the ground-truth positions, and receives no additional future or ground-truth information after initialization. No scene-specific prior or destination information are used since these are estimated online using our framework.

We evaluate tracking results using successful tracks (ST) and ID switches (IDS), as defined in [38]. A successful track is recorded if the tracker stays within 0.5 meter from its own ground-truth after $N = \{16, 24\}$ steps[4]. A track that is more than 0.5 meter away from its ground-truth position is regarded as lost, while a track that is within 0.5 meter from its own ground-truth but closer to another person in the ground-truth is considered an ID switch (IDS). The number of ST and IDS are recorded for each dataset. We also report the total numbers using the "short" interval ($N = 8$ for *Hotel* and $N = 16$ for *Zara01*, *Zara02*, and *Student*) and "long" ($N = 16$ for *Hotel* and $N = 24$ for others). In additional to the metrics used in [38], we have also adopted four more metrics from [21] to measure our tracking performance: the true positive rate (TP), the ratio of false negatives (FN), the multiple object tracking precision (MOTP), and the multiple object tracking accuracy (MOTA). For MOTP, we measure the distance between the ground truth and the prediction by the ratio of overlapping area between their tracking boxes. For the detailed definitions of TP, MOTP and MOTA, please refer to [21].

Furthermore, we have integrated LTA+, ATTR+, and ATTRG+ with HPF to compare with HPF (RVO+). Similar to the combination of RVO+ and HPF, we replace the state transition component with these methods.

Tracking results. The tracking results are presented in Table II. Compared to other motion models, RVO+ has the highest numbers of successful tracks in all the sequences. In addition, estimating the desired velocity increases the overall number of successful tracks for all motion models. When learning the desired velocity, the proportion of "long" successful tracks increases compared to the proportion of "short" STs (e.g., for ATTR/ATTR+, the short STs increase by 14%, whereas the long STs increase by 30%). These results suggest that online learning is useful in terms of long-term tracking. Online-learned desired velocities can prevent the trackers from drifting away. As shown in Figure 5, the desired velocity remains stable as the tracked person steers away to avoid collision. Note that on *Zara02* and *Student*, RVO+ has large number of IDSs. This is because the desired velocity increases the variation of the particle distribution to prevent losing track, but as a consequence reduces the tracking precision in crowded scenes. The number of IDSs are reduced using our HPF model. Figure 4 shows two tracking examples on *Hotel*

[3]We use a similar training/test sets as [38] for cross validation. Each dataset is divided into two halves with the detector trained on one half and tested on the other.

[4]We use $N = \{8, 16\}$ for *Hotel* since it contains shorter trajectories.

TABLE II: Comparison on the number of successful tracks (larger the better) and ID switches (smaller the better).

	Zara01				*Zara02*				*Student*				*Hotel*				Total			
	ST		IDS		ST		IDS		ST		IDS		ST		IDS		ST		IDS	
	N=16	N=24	N=16	N=24	N=16	N=24	N=16	N=24	N=16	N=24	N=16	N=24	N=8	N=16	N=8	N=16	short	long	short	long
LIN	149	57	7	2	343	178	24	13	414	197	62	21	457	149	13	6	1363	581	106	42
LTA	146	54	9	1	315	159	16	4	338	162	50	22	457	153	10	3	1256	528	85	30
LTA+	154	60	6	0	331	185	20	9	391	185	59	22	465	148	14	2	1341	578	99	33
ATTR	129	38	9	1	307	170	16	3	276	107	34	12	462	144	12	4	1174	459	71	20
ATTR+	149	67	7	2	339	185	23	7	394	199	49	24	457	145	8	3	1339	596	87	36
ATTRG	130	39	9	1	300	162	14	3	284	109	36	14	463	144	9	4	1177	454	68	22
ATTRG+	155	60	9	3	316	180	14	13	381	174	50	24	467	145	12	2	1319	559	85	42
RVO	171	75	4	2	365	194	22	7	412	202	50	25	451	147	7	3	1399	618	83	37
RVO+	173	76	5	2	383	209	28	15	419	208	60	25	474	162	5	4	1449	655	98	46
pHPF(LIN)	124	40	9	0	293	158	21	8	348	158	54	13	432	144	12	10	1197	500	96	31
pHPF(RVO+)	141	48	4	1	313	175	23	8	344	175	50	26	457	155	5	0	1255	553	82	35
HPF(LTA+)	171	67	6	0	350	186	16	10	-	-	-	-	475	163	7	2	-	-	-	-
HPF(ATTR+)	175	71	8	2	361	202	20	10	-	-	-	-	477	156	2	4	-	-	-	-
HPF(ATTRG+)	173	72	13	3	343	182	15	6	-	-	-	-	474	161	5	2	-	-	-	-
HPF(RVO+)	**184**	**79**	6	4	**384**	**211**	24	10	**477**	**245**	42	21	**479**	**169**	3	0	**1524**	**704**	75	35

TABLE III: Comparison on metrics TP, FN, MOTP, and MOTA. Larger values of TP, MOTP, and MOTA are better. Smaller values of FN are better.

	Zara01				*Zara02*				*Student*				*Hotel*				Avg.			
	TP	FN	MOTP	MOTA	TP	FN	MOTP	MOTA	TP	FN	MOTP	MOTA	TP	FN	MOTP	MOTA	TP	FN	MOTP	MOTA
LIN	81.6	15.0	68.2	67.0	80.0	14.0	67.1	66.1	63.9	27.4	68.0	36.9	89.0	10.3	76.6	78.9	78.6	16.7	70.0	62.2
LTA	82.1	15.3	69.1	67.2	79.8	15.4	67.4	64.6	63.2	29.7	68.5	33.9	91.6	7.8	75.4	84.1	79.2	17.1	70.1	62.5
LTA+	83.0	14.3	69.2	69.1	80.5	14.2	67.4	66.5	65.2	27.0	68.0	38.7	91.9	7.7	75.2	84.4	80.2	15.8	70.0	64.7
ATTR	79.1	17.6	69.0	61.9	78.3	17.1	67.4	61.3	60.3	32.4	68.7	28.3	90.9	8.7	76.2	82.5	77.2	19.0	70.3	58.5
ATTR+	82.1	14.6	68.9	67.9	80.3	14.2	66.9	66.3	64.7	27.3	68.0	37.8	91.0	8.4	75.9	82.8	79.5	16.1	69.9	63.7
ATTRG	78.5	18.5	69.5	60.4	77.4	18.3	67.7	59.3	59.5	33.0	68.3	26.9	91.5	8.0	75.7	83.7	76.7	19.5	70.3	57.6
ATTRG+	82.4	15.2	68.8	67.6	80.0	15.4	66.8	64.8	63.3	29.0	67.9	34.7	90.8	8.7	75.8	82.4	79.1	17.1	69.8	62.4
RVO	86.3	12.2	68.9	74.5	83.6	13.0	67.4	70.7	64.6	27.7	68.4	37.4	89.1	10.4	75.8	78.9	80.9	15.8	70.1	65.4
RVO+	86.8	11.8	68.9	75.4	84.1	12.0	67.3	72.3	65.4	27.1	68.3	38.7	**93.2**	6.6	76.2	86.7	82.4	14.4	70.2	68.3
pHPF(LIN)	78.5	18.4	68.3	60.6	77.6	16.8	65.8	61.0	64.2	27.0	69.8	37.6	88.2	10.5	74.0	77.8	77.1	18.2	69.5	59.3
pHPF(RVO+)	83.0	14.6	70.1	68.9	81.2	15.6	68.2	65.9	65.7	26.4	70.3	39.7	90.7	8.8	74.9	82.1	80.2	16.4	70.9	64.2
HPF(LTA+)	84.6	12.4	68.2	72.7	81.2	13.0	65.1	68.4	-	-	-	-	92.0	7.6	74.6	85.5	-	-	-	-
HPF(ATTR+)	84.8	11.8	68.3	73.4	81.6	12.3	65.1	69.5	-	-	-	-	92.3	6.5	76.3	86.0	-	-	-	-
HPF(ATTRG+)	86.0	11.3	68.2	75.0	79.7	14.6	65.2	65.3	-	-	-	-	92.1	6.7	76.5	85.7	-	-	-	-
HPF(RVO+)	**88.6**	**10.1**	68.2	**78.9**	**84.6**	**11.9**	65.3	**72.9**	**70.2**	**21.2**	68.4	**49.4**	**93.2**	**6.5**	76.8	**86.8**	**84.2**	**12.4**	69.7	**72.0**

using RVO+ and other models. When the two targets are near to each other and have similar appearances, particles of one target are easily "hijacked" by the other target. Here, the reciprocal collision avoidance of RVO+ prevents hijacking.

Using HPF consistently increases STs compared to PF (i.e., RVO+), with an overall ST increase by $\sim$6%. In addition, IDSs decrease by 30% when using HPF, compared to RVO+. In Table III, the results are consistent with the previous observations. With respect to TP, FN and MOTA, HPF (RVO+) outperforms other PF based methods. Besides, based on the results we have on *Zara01*, *Zara02*, and *Hotel*, HPF improves the tracking performance for most of the crowd motion models (e.g. LTA+ vs. HPF (LTA+)). It is worth noting that there are some missing values for HPF (LTA+), HPF (ATTR+), and HPF (ATTRG+) in Tables II and III. This is because we could not get the results within a reasonable amount of time. Overall, the combinations of LTA+/ATTR+ and HPF have similar tracking performances compared with HPF (RVO+), but they have an additional runtime overhead (see Sec. V-D for a detailed explanation). These results suggest that HPF (RVO+) is capable using the longer-term prediction capability of RVO to overcome noise and occlusion. Figure 6 shows an example of how predictions from previous time-steps in HPF help overcome occlusions in tracking. Figure 7 illustrates four examples showing HPF tracking under frequent interactions and occlusions. Figure 9 shows additional multi-person tracking results using HPF on $Zara01$, $Zara02$ and $Hotel$. Figure 10 and 11 also demonstrate two sets of comparisons, LIN vs HPF and RVO+ vs HPF, respectively. Compared with LIN, HPF takes advantages of RVO's capability in collision-free simulation. As illustrated in Figure 10, the tracker using LIN often switch the identities of pedestrians that walk close to each other, since its underlying motion model does not consider collision avoidance. From Figure 11, the tracker using RVO+ drifts due to the failure of the observation model in highly clustered situation. HPF is able to maintain tracking using higher-order predictions even though there are occlusions. Figure 12

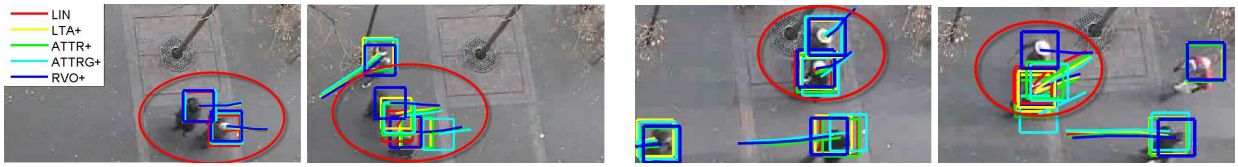


Fig. 4: Two tracking examples (the first row and the second row) from *Hotel*. Each example has two key frames with a circle highlighting that, as two pedestrians walking together, particles of one pedestrian are hijacked by the other pedestrian due to similar appearance, and hence the tracking boxes converge to one of the pedestrians. The motion prior provided by RVO+ keeps the tracking boxes on the right targets.

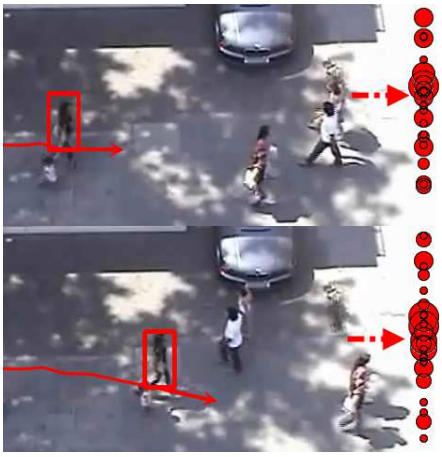
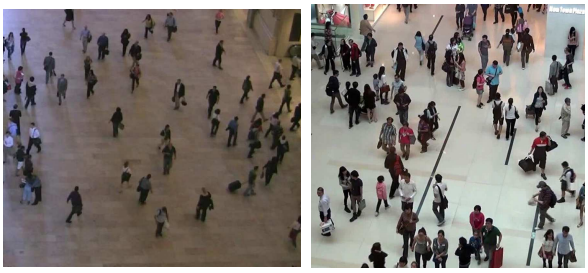


Fig. 5: Example of desired velocity. For the target pedestrian, the arrow indicates the velocity of the person. On the right, the dot-line arrow is the mean desired velocity, and the red circles are the weighted particles (the size indicates the importance), representing desired velocities projected to destinations. When the person deviates to avoid collision with oncoming pedestrians, the desired velocity remains almost the same.

Fig. 6: HPF example on *Zara01*. The red circles represent the 1st-order particle distribution and the green crosses represent 2nd-order particle distribution. When the target pedestrian bypass the oncoming couple, a large portion of 1st-order particles deviate due to the occlusion whereas the majority of 2nd-order particles maintain the tracking boxes on the right target.



(a) (b)

Fig. 8: (a) *Grand* dataset. (b) *Mall* dataset.

show additional examples of how the desired velocity is learned online and helps the trackers. In the figure, the online learned desired velocity is more accurate than the current velocity for the purpose of predicting the final goals.

pHPF (using LIN or RVO+) from [28], which has not been tested in low-fps video, overall performs worse than simple LIN for prediction and tracking. The reason of its poor performance is two-fold. First, a new particle is the weighted average of previous particle predictions. If these predictions are different (but equally valid), then the average prediction will be somewhere in between and may no longer be valid. Second, the weights are trained offline and do not depend on the observations, which causes problems for particles generated during occlusion. HPF addresses both these problems by using a multimodal transition distribution in Eq. 11, and using weights that depend on the observations in Eq. 26.

Finally, it is worth mentioning that our results (LTA+/ATTR+ in Table II) using online-estimation of internal goals are consistent with [38], which uses offline estimation based on labeled goals. For $N = 16$, the total STs on *Zara01*, *Zara02* and *Student* with 2.5fps are 850/837/840 for LIN/LTA/ATTR in [38], and 906/876/882 for LIN/LTA+/ATTR+ reported here. For $N = 24$, the total STs are 394/394/390 for LIN/LTA/ATTR in [38], and 432/430/451 for LIN/LTA+/ATTR+ here. These performance differences among LIN/LTA/ATTR are consistent with those in [38] and our results are generally better than these results due to a more accurate observation model.

Dense crowd videos. We have also evaluated the

(a) (b) (c) (d)

Fig. 7: Four examples from *Zara01*/*Student* using HPF for tracking pedestrians. Each example shows the trajectories of tracking in two frames. The first 3 examples include pedestrian pairs tracked through interaction and occlusions. There is one person (ID=8) lingering around with irregular trajectories and frequent occlusions, and HPF still tracks her throughout the video. The 4th example shows the benefits of our motion model when the appearance model fails; although the interaction takes place in the shadow, HPF maintains the track.

performance on two short video sequences with higher fps (16 fps), the *Grand* dataset [42] and the *Mall* dataset [33]. *Grand* was captured from the Grand Central Station in New York, and *Mall* was captured from the hall of a shopping mall in Hong Kong. Both videos consist of dense crowds with frequent interactions (see Fig. 8). The tracking results are shown in Table IV. Based on the TP/MOTA metrics, in dense crowd situations, there are no significant differences among these methods. This is mainly because pedestrians generally exhibit simple motions in short time gaps, which can be approximated by straight lines and can be well modeled by all motion models. In addition, in dense situations, pedestrians usually slow down, instead of deviating from the current movement direction. This explains why the methods with learned desired velocity (e.g. LTA+ or ATTR+) do not perform well in these two datasets. Nevertheless, the ST/MOTA metrics indicate that our methods (RVO+ and HPF (RVO+)) still have some advantages over other methods in dense crowd video, even though the advantages are not as significant. On the *Mall* dataset, ATTRG obtains the highest MOTA due to the benefits of the group prior. In general, the tracking performance on *Mall* is worse than that of *Grand*, due to the higher density of the crowd and the frequent occurrences of long-term occlusions.

TABLE IV: Results on *Grand* and *Mall*.

	Grand					*Mall*				
	ST	TP	FN	MOTP	MOTA	ST	TP	FN	MOTP	MOTA
LIN	98	64.4	35.6	69.3	58.8	82	49.0	48.4	61.4	20.5
LTA	101	63.6	36.0	68.5	57.6	88	48.7	47.9	61.0	20.8
LTA+	111	62.7	37.1	68.6	55.6	83	47.6	49.2	60.8	18.4
ATTR	107	62.6	37.1	68.7	55.5	89	48.4	48.2	60.5	20.3
ATTR+	107	61.4	38.4	69.2	53.1	77	47.9	49.0	61.7	19.0
ATTRG	116	62.8	36.7	68.5	55.6	84	**49.7**	**46.6**	57.7	**23.2**
ATTRG+	111	61.9	38.0	69.0	53.5	82	48.2	48.4	59.3	19.7
RVO	113	63.2	36.4	68.2	56.8	88	48.2	48.2	63.7	20.0
RVO+	122	63.4	36.6	68.3	56.4	88	49.0	48.2	59.5	20.8
HPF(RVO+)	**123**	**64.6**	**35.4**	68.8	**59.4**	**90**	49.2	**46.6**	59.6	22.6

TABLE V: We run different algorithms on the same sequences from *Zara01* (w/ 11 persons) and *Student* (w/ 29 persons) respectively and record the average time (second) per frame during tracking.

	Zara01	*Student*
LIN	12.5	33.6
RVO+	12.7	34.3
LTA+	26.5	69.9
ATTRG+	26.4	69.9
HPF(RVO+)	22.3	62.4
HPF(LTA+)	251.2	890.3
HPF(ATTRG+)	234.9	1250.3

D. Implementation

We have implemented the particle filter based multi-target tracking system in Matlab and integrated with the crowd motion model RVO, which is in C++ (http://gamma.cs.unc.edu/RVO2/). We have also implemented the core parts of LTA/ATTR in C++. All results shown in this paper are computed using only a single core of CPU. According to Algorithm 1, during each frame, the complexity of PF (or HPF w/ $K = 1$) is $O(KMN)$ while that of HPF w/ $K > 1$ is $O(K^2MN^2)$, where N is the number of agents, M is the number of the particles, and K is the order of HPF. Since the complexity of RVO is $O(N)$, the complexities of PF (RVO+) and HPF (RVO+) are $O(KMN^2)$ and $O(K^2MN^3)$, respectively.

We implemented LTA and ATTR in the same framework, but our methods obtain better timing performance (refer to Table V). As observed from Table V, the average time of HPF (RVO+) is twice the runtime cost of RVO+, while the running time of HPF (LTA+) and HPF (ATTR+) are around 10 times higher than those of LTA+ and ATTR+. For example, in the *Student* benchmark, as there are more pedestrians or agents, HPF (LTA+)

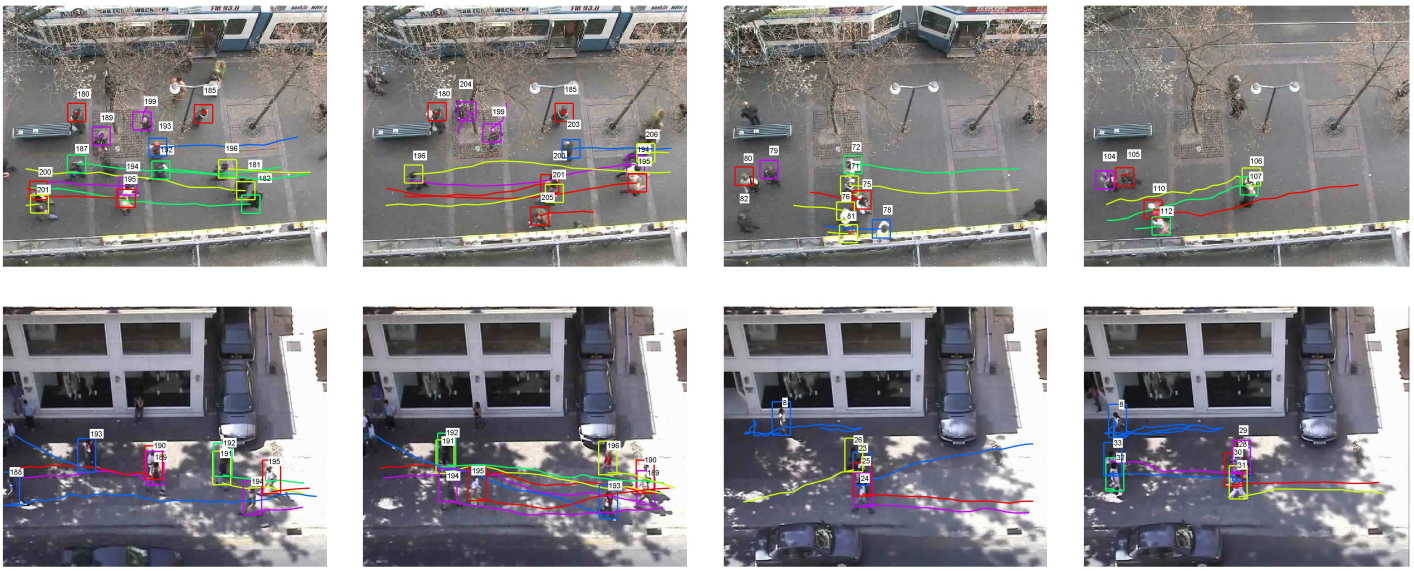


Fig. 9: Eight tracking examples on *Hotel*, *Zara01* and *Zara02*. HPF keeps track of pedestrians well with the interactions and occlusions among pedestrians.

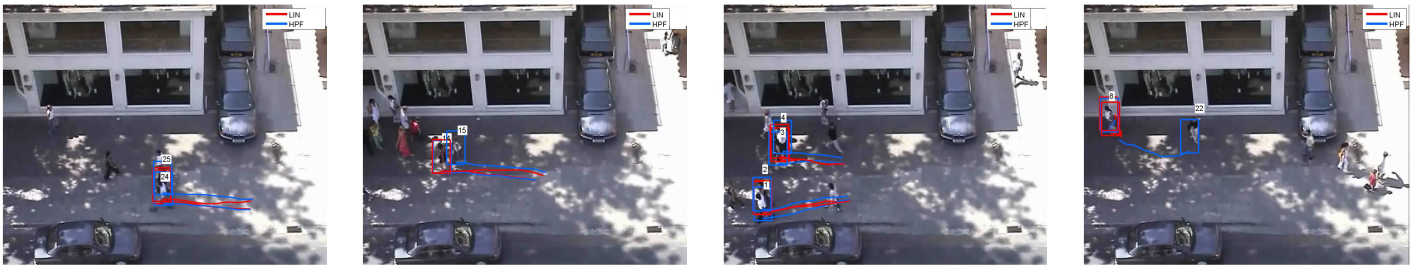


Fig. 10: Four tracking examples on *Zara01* comparing LIN and HPF. As the targets are close to each other, LIN has ID switches (one's particles are 'hijacked' by the other), while HPF does not have this problem.

and HPF (ATTR+) algorithms take about 15 to 20 minutes for tracking. This is because RVO is formulated as convex optimization and the complexity of RVO is $O(N)$, while LTA and ATTR need to compute the crowd state via non-linear non-convex optimization for each agent in a single step. In addition, for HPF, predicting the crowd state K-step ahead requires the crowd motion model to iteratively simulate the crowd state from the timestep $t-K$ for K steps, which is much more time-consuming as the number of agents becomes large.

VI. Conclusions and Future Works

We introduce a multiple-person tracking algorithm that improves existing agent-based tracking methods under low frame rates and without scene priors. We adopt a more precise agent-based motion model, RVO, and integrate it into the particle filter that enables online estimation of the desired velocities. Our PF framework also improves tracking for other agent-based motion models. Moreover, we derive a higher-order particle filter to better leverage the longer-term predictive capabilities of the crowd model. In experiments, we demonstrate that our framework is suitable for predicting pedestrians' behaviors, and improves online-tracking in real-world scenes. In our experiments, we found that our method can work with higher FPS videos, but the benefits may not be as large as when used with low FPS video. Therefore, our algorithm mainly aims at applications, where low FPS video streaming can be deployed in an online manner, with the pedestrians frequently interacting with others.

The main focus of this work is to improve online tracking accuracy by integrating the RVO and other online crowd motion models with the particle filter. In the future work, we will extend our approach to perform real-time crowd tracking for different applications, in particular surveillance, where the input videos are typically of low quality. Currently, our implementation uses a single CPU core. However, our approach can be extended to make use parallel architecture features,

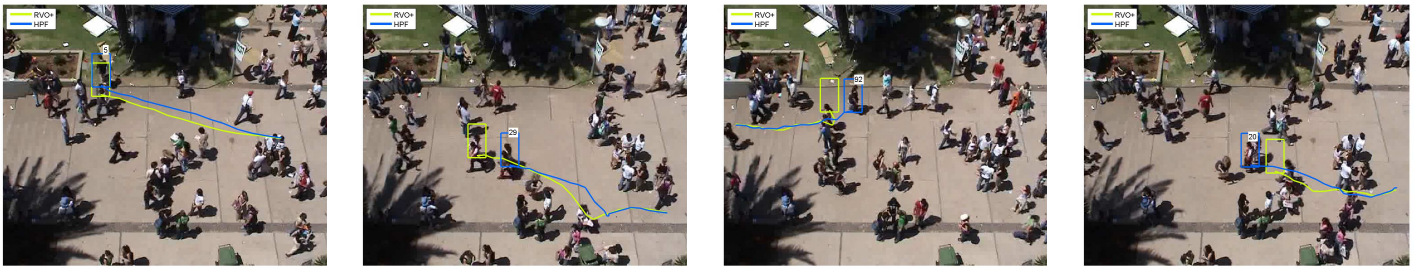


Fig. 11: Four representative tracking examples on *Student* comparing RVO+ and HPF. As pedestrians are occluded, RVO+ drifts away or has ID switches. In constrast, HPF performs better using higher-order particles.

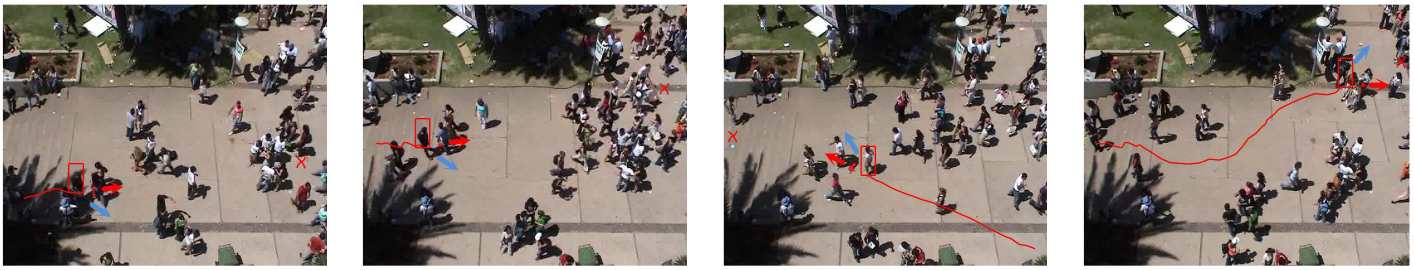


Fig. 12: We show the online-learned desired velocity on *Student*. The blue arrows indicate the current velocities while the red arrows indicate the estimated desired velocities. The red crosses refer to the ground-truth goals of the target persons. The small red circles near the persons refer to particles of desired velocities. (The sizes of the red circles indicate the importances of the particles.) As shown in the examples, the estimated desired velocities remain stable and correct in guiding the tracker towards the destinations even if the person deviates from the correct direction to avoid oncoming pedestrians.

including multiple cores and SIMD of current CPUs and GPUs. In particular, particle filters can be extended using parallel computation techniques to achieve real-time performance, similar to [7, 26].

Furthermore, in our experiments, our methods still have some limitations in terms of tracking dense crowds in higher FPS video. First, PF with learned desired velocity (e.g. LTA+ or ATTR+) suffers from a drifting problem, because the desired velocity in the crowd state can be used to predict pedestrians' collision avoidance but may also introduce uncertainty. Besides, we use a naive appearance model that may fail during long-term occlusions. As part of the future work, we would like to adaptively adjust the dependency of the desired velocity based on the detection confidence during tracking. On the other hand, inspired by [16, 43], we may also adaptively and individually train the appearance model during tracking.

Acknowledgment

We thank Sujeong Kim and Stephen Guy for their prior work as a starting point of this project. This work was partially supported by three grants from the Research Grants Council of Hong Kong (Project No.: CityU 110513, CityU 123212, and CityU 115112), and a SRG grant from City University of Hong Kong (Project No.: 7002768). It is also supported by NSF awards 1117127, 1305286, and a grant from The Boeing Company.

References

[1] S. Ali and M. Shah. Floor fields for tracking in high density crowd scenes. In *ECCV*, 2008.

[2] A. Andriyenko, K. Schindler, and S. Roth. Discrete-continuous optimization for multi-target tracking. In *CVPR*, 2012.

[3] G. Antonini, S. Martinez, M. Bierlaire, and J. Thiran. Behavioral priors for detection and tracking of pedestrians in video sequences. *IJCV*, 2006.

[4] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE TPAMI*, 2011.

[5] L. Bazzani, M. Cristani, and V. Murino. Decentralized particle filter for joint individual-group tracking. In *CVPR*, 2012.

[6] B. Benfold and I. Reid. Stable multi-target tracking in real-time surveillance video. In *CVPR*, 2011.

[7] A. Bera, N. Galoppo, D. Sharlet, A. Lake, and D. Manocha. Adapt: real-time adaptive pedestrian tracking for crowded scenes. In *Conf. on Robotics and Automation*, 2014.

[8] J. Berclaz, F. Fleuret, and P. Fua. Multiple object tracking using flow linear programming. In *Winter-PETS*, 2009.

[9] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE TPAMI*, 2011.

[10] A. Butt and R. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *CVPR*, 2013.

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[12] A. Doucet, N. de Freitas, and N. Gordon. *An introduction to sequential Monte Carlo methods*. Springer, 2001.

[13] M. Felsberg and F. Larsson. Learning higher-order markov models for object tracking in image sequences. In *Advances in Visual Computing*. 2009.

[14] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7), 1998.

[15] S. Guy, S. Curtis, M. Lin, and D. Manocha. Least-effort trajectories lead to emergent crowd behaviors. *Physical Review E*, 85(1), 2012.

[16] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011.

[17] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51(5), 1995.

[18] R. Hess and A. Fern. Discriminatively trained particle filters for complex multi-object tracking. In *CVPR*, 2009.

[19] H. Jiang, S. Fels, and J. Little. A linear programming approach for multiple object tracking. In *CVPR*, 2007.

[20] Z. Jin and B. Bhanu. Single camera multi-person tracking based on crowd simulation. In *ICPR*, 2012.

[21] B. Keni and S. Rainer. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008.

[22] Z. Khan, T. Balch, and F. Dellaert. An MCMC-based particle filter for tracking multiple interacting targets. In *ECCV*, 2004.

[23] L. Kratz and K. Nishino. Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. *IEEE TPAMI*, 2012.

[24] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3), 2007.

[25] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *CVPR*, 2009.

[26] A. Montemayor, J. Pantrigo, Á. Sánchez, and F. Fernández. Particle filter on GPUs for real-time tracking. In *ACM SIGGRAPH 2004 Posters*, 2004.

[27] K. Okuma, A. Taleghani, N. De Freitas, J. Little, and D. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*. 2004.

[28] P. Pan and D. Schonfeld. Visual tracking using high-order particle filtering. *IEEE Signal Processing Letters*, 2011.

[29] D. Park, J. Kwon, and K. Lee. Robust visual tracking using autoregressive hidden markov model. In *CVPR*, 2012.

[30] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: modeling social behavior for multi-target tracking. In *ICCV*, 2009.

[31] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *ECCV*. 2002.

[32] M. Rodriguez, S. Ali, and T. Kanade. Tracking in unstructured crowded scenes. In *ICCV*, 2009.

[33] J. Shao, C. Loy, and X. Wang. Scene-independent group profiling in crowd. CVPR, 2014.

[34] X. Song, X. Shao, H. Zhao, J. Cui, R. Shibasaki, and H. Zha. An online approach: learning-semantic-scene-by-tracking and tracking-by-learning-semantic-scene. In *CVPR*, 2010.

[35] J. van den Berg, S. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. *Robotics Research*, 2011.

[36] J. Vermaak, A. Doucet, and P. Pérez. Maintaining multimodality through mixture tracking. In *ICCV*, 2003.

[37] H. Xiong, D. Zheng, Q. Zhu, B. Wang, and Y. Zheng. A structured learning-based graph matching method for tracking dynamic multiple objects. *IEEE TCSVT*, 2013.

[38] K. Yamaguchi, A. Berg, L. Ortiz, and T. Berg. Who are you with and where are you going? In *CVPR*, 2011.

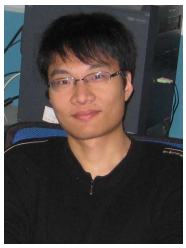
[39] B. Yang and R. Nevatia. Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *CVPR*, 2012.

[40] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.

[41] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008.

[42] B. Zhou, X. Tang, and X. Wang. Measuring crowd collectiveness. In *CVPR*, 2013.

[43] O. Zoidi, A. Tefas, and I. Pitas. Visual object tracking based on local steering kernels and color histograms. *IEEE TCSVT*, 2013.



Wenxi Liu is currently PhD student at City University of Hong Kong. He received B.S. degree from Shenzhen University, China, in 2009. His research interests include computer vision, pedestrian motion model, and crowd simulation.

Antoni B. Chan received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, in 2000 and 2001, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, in 2008. From 2001 to 2003, he was a Visiting Scientist with the Vision and Image Analysis Laboratory, Cornell University, Ithaca, NY, and in 2009, he was a Postdoctoral Researcher with the Statistical Visual Computing Laboratory, UCSD. In 2009, he joined the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, as an Assistant Professor. His research interests include computer vision, machine learning, pattern recognition, and music analysis. Dr. Chan was the recipient of an NSF IGERT Fellowship from 2006 to 2008, and an Early Career Award in 2012 from the Research Grants Council of the Hong Kong SAR, China.

Rynson W. H. Lau received his Ph.D. degree from University of Cambridge. He was on the faculty of Durham University and Hong Kong Polytechnic University. He is now with City University of Hong Kong.

Rynson serves on the Editorial Board of Computer Animation and Virtual Worlds and IEEE Trans. on Learning Technologies. He has served as the Guest Editor of a number of journal special issues, including *ACM Trans. on Internet Technology*, *IEEE Trans. on Multimedia*, *IEEE Trans. on Vis. and Computer Graphics*, and *IEEE Computer Graphics & Applications*. He has also served in the committee of a number of conferences, including Program Co-chair of *ACM VRST 2004*, *ACM MTDL 2009*, *IEEE U-Media 2010*, *ICWL 2014*, and Conference Co-chair of *CASA 2005*, *ACM VRST 2005*, *ICWL 2007*, *ACM MDI 2009*, *ACM VRST 2014*. His research interests include computer graphics, image processing and multimedia systems.

Dinesh Manocha is currently a Phi Delta Theta/Mason Distinguished Professor of Computer Science at the University of North Carolina at Chapel Hill. He received his Ph.D. in Computer Science at the University of California at Berkeley 1992. He has published more than 330 papers in computer graphics, geometric computation, robotics and many-core computing and received 12 best-paper awards. Some of the software systems developed by his group on collision and geometric computations, interactive rendering, crowd simulation, and GPU-based algorithms have been downloaded by more than 100K users and widely licensed by commercial vendors. Manocha has served in the program committees of more than 100 leading conferences and in the editorial board of more than 10 leading journals. He has won many awards including NSF Career Award, ONR Young Investigator Award, Sloan Fellowship, IBM Fellowship, SIGMOD IndySort Winner, Honda Research Award and UNC Hettleman Prize. He is a Fellow of *ACM*, *AAAS*, and *IEEE* and received Distinguished Alumni Award from Indian Institute of Technology, Delhi.