Realtime Multilevel Crowd Tracking using Reciprocal Velocity Obstacles

Aniket Bera & Dinesh Manocha
University of North Carolina at Chapel Hill
http://gamma.cs.unc.edu/RCrowdT

***Abstract*—We present a novel, realtime algorithm to compute the trajectory of each pedestrian in moderately dense crowd scenes. Our formulation is based on an adaptive particle filtering scheme that uses a multi-agent motion model based on velocity-obstacles, and takes into account local interactions as well as physical and personal constraints of each pedestrian. Our method dynamically changes the number of particles allocated to each pedestrian based on different confidence metrics. Additionally,we use a new high-definition crowd video dataset to evaluate the performance of different pedestrian tracking algorithms. This dataset consists of videos of indoor and outdoor scenes recorded at different locations, each with 30-80 pedestrians. Using this dataset, we highlight the performance benefits of our algorithm over prior techniques. In practice, our algorithm can compute trajectories of tens of pedestrians on a multi-core desktop CPU at interactive rates (27-30 frames per second). To the best of our knowledge, our approach is 4-5 times faster than prior methods that provide similar accuracy.**

I. INTRODUCTION

Tracking pedestrians in a crowd is a well-studied problem in computer vision, robotics, and related areas.The goal of pedestrian tracking is to compute the trajectory of each moving pedestrian in a video using spatial and temporal localization. It is increasingly important to track and predict pedestrian motion and behavior at realtime rates, as autonomous robots and driverless cars are increasingly sharing physical space with tens or hundreds of pedestrians, it is important to *track*, and also *predict* motion and behavior at realtime rates. We also need real-time crowd tracking capabilities for surveillance activities [1], evaluating crowd behaviors [2], detecting anomalous behavior [3], crowd counting [4], realtime evacuation planning [5], and collision-free navigation in dynamics scenes, among other applications.

Because of their variable behaviors, pedestrians are as difficult to track as any object. Pedestrians tend to change their speed to avoid collisions with obstacles and other pedestrians. Large variations in their appearance and illumination makes it hard for color-based template tracking algorithms to track a single pedestrian continuously. In crowded scenes, the pairwise interactions between pedestrians increase significantly, adding to the complexity of predictive tracking schemes.

Some of the most reliable tracking methods have been developed for offline, non-realtime applications, because they can make multiple passes over each video frame and can take advantage of their knowledge of future frames. Approaches that have been proposed for online or realtime pedestrian tracking are currently limited to simple scenes with only a few pedestrians (fewer than 10). The realtime handling of scenes with higher crowd density (a high number of pedestrians - 3-4 pedestrians per squared meter - located in a small area) is an especially difficult challenge.

Fig. 1: *Tracking Street Crowds: Our algorithm achieves high accuracy in this dense street dataset with 144 pedestrians and can track at 27fps on a multi-core desktop CPU (Dataset - NPLACE-3).*

The primary challenge in real-time tracking for crowded scenes comes from the need to exploit typical pedestrian behavior. In real-world scenarios, the trajectory of each pedestrian is governed by two factors: its intermediate goal location, and local interactions with other pedestrians and obstacles. These local interactions include maneuvers to avoid collisions and shifts in trajectory to maximize the efficiency of the pedestrian's path towards the goal. A successful crowd tracker must exploit these characteristics by using an appropriate motion model for each pedestrian. Models that do not take into account the high variability of pedestrian trajectories in densely crowded environments, including many widely used motion models that are based on constant velocity or constant acceleration [6] models, fail to exploit these pedestrian behaviors.

Main Results: To track pedestrians in dense crowds, we present a novel realtime multilevel tracking algorithm based on particle filtering. Our approach dynamically changes the number of particles allocated to each pedestrian based on multiple confidence metrics. To compute the confidence metrics, we use a non-linear parametric multi-agent motion model, Reciprocal Velocity Obstacles(RVO) [7], which takes into account reactive behavior of pedestrians in a dense setting. Our approach, which can easily be generalized to other multi-model particle filters, aims to significantly decrease the computational cost for realtime tracking.

We use RVOs first to model the state transition distribution, which includes the motion-prior pedestrian state; we then estimate and iteratively refine the RVO parameters, improving the motion model's accuracy for tracking prediction and for the confidence measures.

We evaluate the performance of our algorithm on new datasets, which include both indoor and outdoor scenes recorded at different locations with 30 - 80 pedestrians, compare its performance with that of prior methods. Our algorithm can track tens of pedestrians at realtime rates (i.e. more than 25fps) on a multi-core CPU; in practice, our approach is about 4-5 times faster than prior methods that provide similar accuracy.

II. Related Work

In this section, we briefly review some prior work on pedestrian tracking. For overviews of the field of pedestrian tracking, we refer the reader to two surveys [10], [11].

Pedestrian tracking algorithms can be classified as either online or offline: online trackers use only the present or past frames, while offline trackers also use data from future frames. Some state-of-the-art accurate tracking methods are offline [12] [13]. However, some of these methods, which require future-state information and may make multiple passes over the video frames, are not useful for realtime applications. Therefore, we will survey prior work on online tracker algorithms only. Zhang et al. [14] proposed an online approach that uses non-adaptive random projections to model the structure of the image feature space of objects. Oron et al. [15] described a method to estimate the amount of local deformation in rigid or deformable objects. The color-based probabilistic tracking method proposed by Perez et al. [16] is fast but prone to loss of trajectories from occlusion. Collins's method [17] tracked blob via mean-shifts, and Jia et al. [18] presented a method to track objects using a local sparse appearance model. Kwon et al. [19] proposed a method that adaptively switches trackers; the trackers are sampled using the Markov Chain Monte Carlo method from a predefined tracker space. Particle filters have been widely used for online tracking [20] [21] [22]. Many crowd tracking algorithms use motion models to improve tracking accuracy and prediction. Song et al. [23] proposed an approach to cluster pedestrian trajectories based on the assumption that "persons only appear/disappear at entry/exit". Ali et al. [24] presented a method based on floor-fields to compute the probability of motion in highly dense crowded scenes. Kratz et al. [25] and Zhao et al. [26] presented an approach using local motion patterns in dense videos. Rodriguez et al. [27] used a large collection of public crowd videos to learn crowd motion patterns by extracting global video features.These methods are well suited for dense crowds that can be characterized by a given global motion pattern; however, the most commonly used motion models are linear single-agent models, including constant velocity and constant acceleration [6]. Other motion models used in pedestrian tracking are the Social Force model [28] [29] [3], LTA [30], RVO [31] and ATTR [32].

III. Our Approach

A. Algorithm

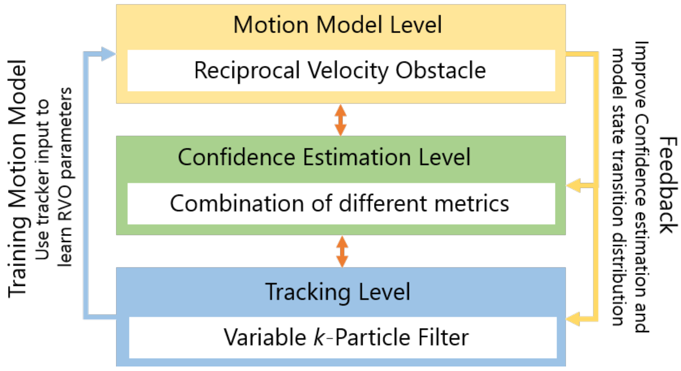

Fig. 3: *Our algorithm uses three levels to track each pedestrian in a crowd. In the first level, we calculate the tracker output using a variable particle filter based approach. In the second level, we calculate the confidence of our tracker using a motion-model-centric metric approach. The number of particles used to track a pedestrian, k, varies over different frames based on the confidence estimate. In the third level, we estimate and iteratively refine the RVO-based motion parameters, which provides a continuous feedback loop to the other levels.*

In this section, we give an overview of our approach and present the details of our motion model. Our underlying tracking algorithm is based on particle filters. The particle filter is a parametric method which solves non-Gaussian and non-linear state estimation problems [33]. Since it can recover from lost tracks and occlusions, particle filters are frequently used in object tracking. However, particle filters' performance can be computationally intensive, since the cost is directly proportional to the number of particles being used per pedestrian. There is a trade-off between accuracy and efficiency, since the more particles per pedestrian, the higher the probability of tracking a pedestrian accurately. To balance computation cost with accuracy, we must use the optimal number number of particles for each pedestrian. Ideally, we would use fewer particles (lower k) at most times and increase k only when needed, such as when there is a large change in motion trajectory, lighting, appearance or partial occlusions. We therefore propose a *multi-level*(MLPF) approach that adaptively computes k for every pedestrian at each timestep.

A particle filter-based tracker depends on its motion model, which predicts the particles' propagation to the next state. Some of the commonly used single-agent motion models based on constant velocity or constant acceleration may not work in dense crowds, since the increased interactions between the pedestrians can break down the assumptions of constant velocity or acceleration.

B. Reciprocal Velocity Obstacles

In contrast to the assumptions of constancy in the common single-agent models, the RVO motion model uses the

(a) Online Boosting (b) Mean-shift (c) Our Approach

Fig. 2: *Performance comparison of our approach with other algorithms on a crowded scene.: (a) Online Boosting [8](11 fps); (b) MeanShift algorithm [9] (33 fps); (c) Our Approach MLPF-RVO (28 fps). Overall, the accuracy of our approach is comparable to Online Boosting and the performance is a little slower than the MeanShift algorithm (Dataset - NPLACE-2)*

state information from the current timestep to predict all the pedestrians' states at every timestep [7]. RVO is basically a local collision-avoidance and navigation algorithm; it works by tracking one time-step at a time, allowing the state of each agent to evolve into locally collision-avoidant states during the next time step. Each agent or pedestrian is represented as a 2D circle in the plane; the state information for each agent consists of radius, current position and velocity, and preferred velocity for the next timestep. This preferred velocity is governed by the intermediate goal position. The RVO algorithm assumes that each pedestrian also knows the current position and velocity of other nearby agents.

Let $\mathbf{v}_{pref}$ be the preferred velocity, based on the intermediate goal location, for a pedestrian. The RVO formulation takes into account the position and velocity of each neighboring pedestrian to compute the new velocity. The velocity of the neighbors is used to formulate the ORCA constraints for local collision avoidance [7]. The computation of new velocity is expressed as an optimization problem for each pedestrian. If an agent's preferred velocity is forbidden by the ORCA constraints, that agent chooses the closest velocity that lies in the feasible region:

$$\mathbf{v}_{RVO} = \underset{\mathbf{v} \notin OCRA}{\arg\max} \|\mathbf{v} - \mathbf{v}_{pref}\|. \tag{1}$$

The ORCA constraints are represented as the boundary of a half plane containing the space of all collision-free velocities. We highlight the computation for two pedestrians, say x_1 and x_2. The minimum vector $\mathbf{u}$ of the change in relative velocity to avoid a collision is computed. ORCA requires each pedestrian to change its current velocity by at least $\frac{1}{2}\mathbf{u}$. Then the boundary of the ORCA constraint corresponds to a line containing the point $\mathbf{v} + \frac{1}{2}\vec{u}$ in the velocity space, with the direction perpendicular to $\mathbf{u}$. The ORCA constraint on x_1's velocity induced by x_2 is given as:

$$ORCA_{x_1|x_2} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}_{x_1} + \frac{1}{2}\mathbf{u})) \cdot \hat{\mathbf{u}} \geq 0\}, \tag{2}$$

where $\mathbf{v}_{x_1}$ is x_1's current velocity and $\hat{\mathbf{u}}$ is the normalized vector $\mathbf{u}$. More details and mathematical formulations of the ORCA constraints are given in [7].

C. Multi-Level Particle Filter (MLPF)

Our approach has three levels, as shown in Fig. 3. The first is the 'tracking level', where we fit each pedestrian RVO state into the standard particle-filter formulation. The second level is 'confidence estimation', where we use multiple metrics to measure the reliability of the tracker and adaptively modify the number of active particles per pedestrian. The third level is the 'motion model', where we estimate and iteratively refine the RVO parameters to best match our input video. We use this trained RVO model as an input to the levels below.

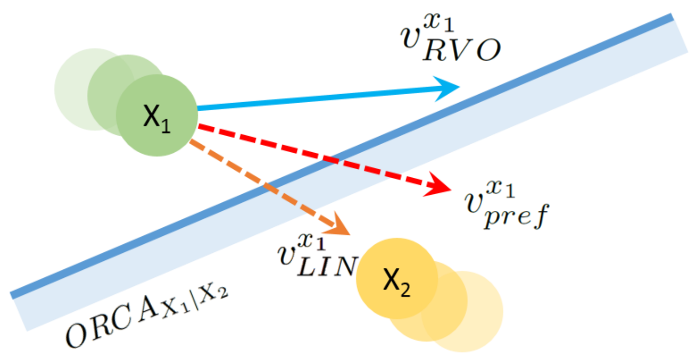

Fig. 4: *RVO Multi-agent Motion Model illustrating a pedestrian x_1's preferred velocity (v_{pref}) and the optimal collision-free velocity computed by the RVO model (v_{RVO}). It is the velocity closest to v_{pref}, that lies in the feasible region. We also show the pedestrian velocity computed using the constant velocity model (v_{LIN}), which leads to a future collision state and hence an incorrect prediction.*

Tracking Level: We use the standard particle filter and combine it with RVO parameters. Given a pedestrian's RVO state x_t at time step t, RVO offers collision-free motion dynamics inference, denoted as f, to predict the agent's next state x_{t+1}. We denote the error in the prediction generated by the underlying RVO motion model as q. The observations of our framework or tracker can be represented by a function h that projects the state x_t to an observed state, denoted as y_t. We denote the error between the observed states and the ground truth as r. We can now phrase them formally in terms of a standard particle filter as below:

$$x_{k+1} = f(x_k) + q, \tag{3}$$
$$y_k = h(x_k) + r. \tag{4}$$

In our formulation, we use RVO to infer dynamic transition, $p(x_t|x_{t-1})$, for particle filtering.

D. Confidence Estimation Level:

We analyze the confidence of our tracker based on the number of particles. We use two metrics: propagation reliability and motion model reliability.

a) Propagation Reliability: This is a measure of how well the object matches the target candidate during each frame. Once the normalized weights of all the particles in the our algorithm fall below a certain threshold, we remove those particles and reduce the total number of particles used for that pedestrian. The particles are expected to have higher weights at locations that correspond to the actual positions of the tracked objects. If the number of total particles becomes less than a threshold N, we resample the particles for that pedestrian and make sure that each pedestrian is approximated by at least N particles.

b) Motion Model Reliability: This is a key metric in our confidence estimation. We calculate the normalized difference between the tracked state in our particle-filtering framework, t_{PF}, and the predicted motion model state t_{RVO}. If this difference, d, is high, more particles are introduced into the system, which is then resampled.. Otherwise, we gradually reduce the number of particles and retest the confidence at each timestep.The computations related to each particle are independent and have the same overhead; therefore, the computation cost is directly proportional to the number of particles used. *(see Algorithm 2, Fig. 5)*

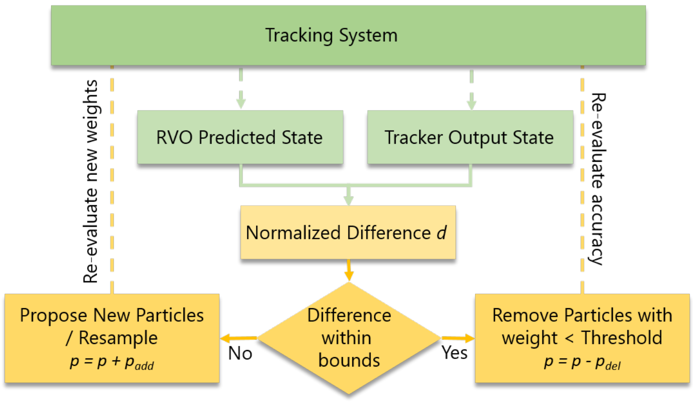

Fig. 5: *Motion Model Reliability: We highlight the motion model metric computation of the confidence estimation algorithm based on the information from the trained RVO and the k-particle filter. (See Algorithm 2)*

E. High-Level Motion Model Level:

In this level, we learn the RVO parameters and refine our motion model framework to better match the behavior of each pedestrian. The system uses statistical inferencing techniques to compute and predict the agent trajectories from the noisy tracker data.

In Fig. 6 we highlight how the motion model computes the trajectory of each moving pedestrian and uses tracked data to adaptively learn the simulation parameters. The resulting motion for each agent is computed using statistical techniques: the Ensemble Kalman filter (EnKF) is combined with the maximum likelihood estimation algorithm to learn individual motion parameters [34].

The current pedestrian state is computed recursively; we use the tracker output to continuously re-estimate each pedestrian's current state. The model combines the EM (Expectation-Maximization) algorithm with an ensemble Kalman Filtering

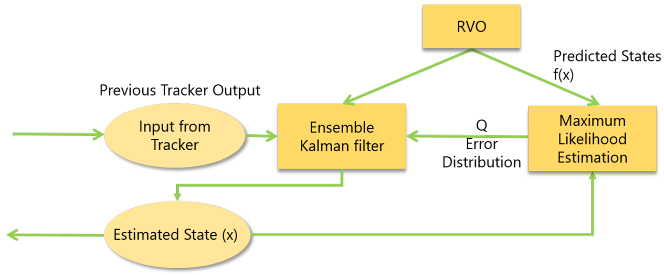

Fig. 6: *Overview of the High-Level Motion Model Augmentation Level. This level draws input from the Confidence Estimation level, learns model parameters and improves tracking by providing feedback. The feedback is bidirectional and the model is re-trained after a fixed number of frames.*

approach to iteratively approximate the motion model state of each agent at every timestep.

We perform Bayesian learning for each pedestrian. Every pedestrian can be represented by a motion model state vector $\mathbf{x}$. Given a pedestrian's state (position, velocity and preferred velocity), we use the motion model f to predict the pedestrian's next state $\mathbf{x}_{k+1}$. We denote the motion model's error in predicting the state as $\mathbf{q}$ and it follows a Gaussian distribution with covariance $\mathbf{E}$. Hence,

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + \mathbf{q}. \tag{5}$$

Additionally, we assume that our output from the tracking stage can be represented by a function h that projects the predicted state x_k to an observed state z_k. r is the error between the observed state and the ground truth. Hence:

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{r}. \tag{6}$$

Our motion model level uses RVO to represent the function f and EnKF to estimate the simulation parameters that best fit the observed data. The EM-algorithm is used to estimate the model error for each pedestrian. Better estimation of the model error improves the Kalman Filtering process, which in turn improves the pedestrian state prediction. We perform EnKF and EM steps for each pedestrian, separately, but we do take into account all nearby pedestrians used in RVO motion model computation $f(x)$. This improves the accuracy of our predictor and the overall trajectory computation in dense scenes or scenes with cross-flow pedestrian motion.

Fig. 7: *Adaptive Refinement and Prediction. As new data is observed (red dot), we re-estimate the distribution of likely values of the RVO states (shown as a dashed ellipse). $z_0 ... z_t$ are the set of observations for each pedestrian at time t; $x_0 ... x_t$ correspond to the predicted RVO state that best reproduces the actual trajectory. The blue arrow indicates the predicted velocity vector.*

Fig. 8: The results of our approach on the different datasets. From top-right, clockwise: *IITF-1, IITF-2, NPLACE-1, NPLACE-2, IITF-3, NPLACE-3, IITF-4, NDLS-1, NDLS-2, IITF-5*. These datasets are available on our website, *http://gamma.cs.unc.edu/RCrowdT*

	IITF-1		IITF-2		NPLACE-1		NPLACE-2		IITF-3		NPLACE-3		IITF-4		NDLS-1		NDLS-2		IITF-5	
	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS
Online Boosting	74%	7	46%	6	74%	8	76%	11	57%	6	67%	7	56%	6	58%	6	75%	7	66%	7
KMS	28%	31	17%	32	26%	29	31%	33	23%	29	27%	26	23%	31	25%	29	34%	30	26%	29
SMS	68%	14	38%	13	64%	15	66%	19	48%	14	59%	13	49%	15	51%	14	63%	16	55%	13
ASLA	72%	7	39%	7	70%	8	70%	12	51%	6	60%	7	50%	6	49%	7	68%	8	60%	8
Frag	41%	20	34%	21	40%	19	57%	19	54%	18	51%	21	48%	20	50%	22	66%	18	51%	19
MLPF-LIN	63%	27	36%	26	67%	27	69%	28	51%	26	60%	28	52%	26	53%	26	68%	27	59%	27
SLPF-LIN	64%	12	38%	12	68%	10	69%	12	51%	11	61%	11	53%	10	53%	9	68%	11	60%	12
SLPF-RVO	71%	11	42%	10	73%	10	74%	11	53%	11	66%	11	53%	10	58%	10	72%	13	65%	11
MLPF-RVO	**69%**	**27**	**42%**	**26**	**71%**	**26**	**73%**	**28**	**53%**	**26**	**64%**	**27**	**53%**	**26**	**57%**	**26**	**72%**	**27**	**64%**	**27**

TABLE I: *We compare the accuracy (in terms of pedestrians tracked in the video sequence) and speed (in terms of frames per second) of the following online algorithms: Online Boosting [8], KMS [35], SMS [17], ASLA [18], and Frag [36] and also with MLPF-RVO, SLPF-RVO, MLPF-LIN and SLPF-LIN. (Abbreviation: Acc-Tracking Accuracy, FPS- Average frames per sec)*

TABLE II: *Crowd scenes used as benchmarks. We highlight many attributes of crowds in these videos, including density and the number of number of pedestrians tracked. We use the following abbreviations: Background Variations(BV), Partial Occlusion(PO), Complete Occlusion(CO), Illumination Changes(IC)*

Dataset	Challenges	Density	Pedestrians tracked
NDLS-1	BV, PO, IC	High	131
NDLS-2	BV, PO, IC, CO	Medium	72
NPLACE-1	BV, PO, IC	Medium	79
NPLACE-2	BV, PO	Low	56
NPLACE-3	BV, PO, IC, CO	High	144
IITF-1	BV, PO, IC, CO	High	167
IITF-2	BV, PO, IC, CO	High	68
IITF-3	BV, PO, IC, CO	High	189
IITF-4	BV, PO, IC, CO	High	116
IITF-5	BV, PO, IC, CO	High	71

We use this trained motion model in our particle filter for dynamic transition, and we use the predicted RVO state to calculate the confidence in the 'Confidence Estimation' level of the algorithm

IV. Implementation and Results

In this section, we highlight the performance of our algorithm on different benchmarks and compare the performance with that of some prior techniques. (See Table II, Fig. 8)

We tested these algorithms on an Intel©x86 Processor (8 Cores). 8MB Cache, 3.90 GHz. Our algorithm is implemented in C++, and many components use OpenMP for exploiting multiple cores.

For our experiment we have divided our system into two phases: *Training:* This is the 'motion model' level of our algorithm shown in Fig. 3. We run our input video for k frames and estimate the RVO parameters. *Predict:* After training, we use the predicted state and the trained motion model for improving accuracy and for confidence calculation.

For our k-particle filter, we vary k in the following manner. If there is a loss in confidence, we increase k in multiples of 100. After every subsequent increase, we keep it constant for 10 frames, unless the confidence drops below our threshold. After we achieve a stable confidence estimate, we gradually decrease the number of particles by removing particles with low weights. Please refer to Fig. 9.

As shown in Table 1, we have compared our approach with many well-known online tracking algorithms. In order to demonstrate the benefits of our multi-level tracker and the improved motion model, we consider the following four combinations:

- SLPF-LIN: In this case, the particle filter uses a constant number of particles along with a constant velocity motion model.
- SLPF-RVO: This uses a constant number of particles along with RVO as the motion model.
- MLPF-LIN: We dynamically change the number of particles along with constant velociy.
- MLPF-RVO: This uses an adaptive particle filter along with RVOs. In our benchmarks, this version achieves realtime performance with the best accuracy.

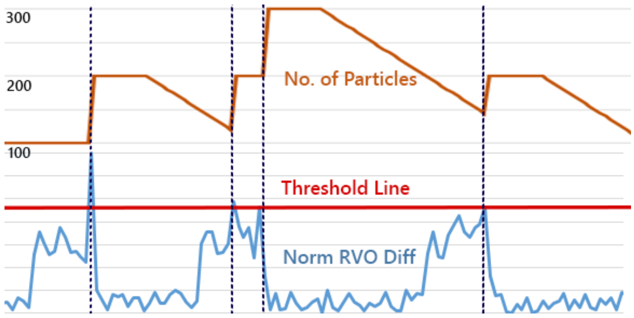

Fig. 9: *k-particle filter implementation. The blue graph below denotes the normalized difference metric,* d*. Once* d *exceeds a certain threshold (red line), we increase the number of particles,* p *(denoted by orange line) by 100. After a while, we start decreasing the particles slowly to see if we are able to maintain the required confidence. This process is repeated for every pedestrian. X-Axis represents number of active particles.*

V. Limitations, Conclusions, and Future Work

We present a realtime algorithm for pedestrian tracking in crowded scenes. Our algorithm provides a good balance between accuracy and speed. We highlight its performance on many pedestrian datasets, and show that our algorithm can track crowded scenes in realtime on a PC with a multi-core CPU. As compared to prior algorithms of similar accuracy, we obtain 4-5 times speedup.

Our approach has some limitations related to our motion model. RVOs do not take into account physiological and psychological pedestrian traits. All pedestrians are modeled withthe same sensitivity towards gender and density; the model doesn't take into account heterogeneous characteristics. These may have introduced additional errors in our confidence estimation. In practice, the performance of the algorithm can vary based on various other attributes of the input video.

For future work, we plan to use improved motion models that exploit 'fundamental diagrams' [37], which should result in improved prediction in highly dense scenarios. In terms of performance, we would like to exploit GPU capabilities and evaluate the performance on mobile devices.

VI. Acknowledgements

This work was supported by NSF awards 1000579, 1117127, 1305286, Intel and a grant from The Boeing Company.

References

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

[2] N. Shoval and M. Isaacson, "Application of tracking technologies to the study of pedestrian spatial behavior*," *The Professional Geographer*, 2006.

[3] R. Mehran, A. Oyama, and M. Shah, "Abnormal crowd behavior detection using social force model," in *CVPR*, 2009.

[4] O. Sidla, Y. Lypetskyy, N. Brandle, and S. Seer, "Pedestrian detection and tracking for counting applications in crowded situations," in *AVSS*, 2006.

[5] N. Doulamis, "Evacuation planning through cognitive crowd tracking," in *IWSSIP*, 2009.

[6] A. Del Bimbo and F. Dini, "Particle filter-based visual tracking with a first order dynamic model and uncertainty adaptation," *Computer Vision and Image Understanding*, 2011.

[7] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, 2011.

[8] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *CVPR*, 2009.

[9] Y. Cheng, "Mean shift, mode seeking, and clustering," *PAMI*, 1995.

[10] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *PAMI*, 2009.

[11] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm Computing Surveys (CSUR)*, 2006.

[12] P. Sharma, C. Huang, and R. Nevatia, "Unsupervised incremental learning for improved object detection in a video," in *CVPR*, 2012.

[13] M. Rodriguez, I. Laptev, J. Sivic, and J.-Y. Audibert, "Density-aware person detection and tracking in crowds," in *ICCV*, 2011.

[14] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *ECCV*, 2012.

[15] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," in *CVPR*, 2012.

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

[17] R. T. Collins, "Mean-shift blob tracking through scale space," in *CVPR*, 2003.

[18] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *CPVR*, 2012.

[19] J. Kwon and K. M. Lee, "Tracking by sampling trackers," in *ICCV*, 2011.

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

[21] Z. Khan, T. Balch, and F. Dellaert, "An mcmc-based particle filter for tracking multiple interacting targets," in *ECCV*, 2004.

[22] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "Object tracking with an adaptive color-based particle filter," in *Pattern Recognition*, 2002.

[23] X. Song, X. Shao, Q. Zhang, R. Shibasaki, H. Zhao, J. Cui, and H. Zha, "A fully online and unsupervised system for large and high-density area surveillance: Tracking, semantic scene learning and abnormality detection," *TIST*, 2013.

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

[25] L. Kratz and K. Nishino, "Going with the flow: pedestrian efficiency in crowded scenes," in *ECCV*, 2012.

[26] X. Zhao, D. Gong, and G. Medioni, "Tracking using motion patterns for very crowded scenes," in *ECCV*, 2012.

[27] M. Rodriguez, J. Sivic, I. Laptev, and J.-Y. Audibert, "Data-driven crowd analysis in videos," in *ICCV*, 2011.

[28] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, 1995.

[29] A. Bera, N. Galoppo, A. T. Lake, and D. Manocha, "Adapt: Real-time adaptive pedestrian tracking for crowded scenes," in *ICRA*, 2014.

[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] W. Liu, A. B. Chan, R. W. Lau, and D. Manocha, "Leveraging long-term predictions and online-learning in agent-based multiple person tracking," in *arXiv:1402.2016v2*.

[32] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?" in *CVPR*, 2011.

[33] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *Signal Processing, IEEE Transactions on*, 2002.

[34] S. Kim, S. J. Guy, W. Liu, R. W. Lau, M. C. Lin, and D. Manocha, "Predicting pedestrian trajectories using velocity-space reasoning," in *WAFR*, 2012.

[35] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *PAMI*, 2003.

[36] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *CVPR*, 2006.

[37] S. Curtis and D. Manocha, "Pedestrian simulation using geometric reasoning in velocity space," in PEDS, 2012.