REACH - Realtime Crowd tracking using a Hybrid motion model

Aniket Bera[1] and Dinesh Manocha[1]
http://gamma.cs.unc.edu/REACH

***Abstract*— We present a novel, real-time algorithm to extract the trajectory of each pedestrian in moderately dense crowd videos. In order to improve the tracking accuracy, we use a hybrid motion model that combines discrete and continuous flow models. The discrete model is based on microscopic agent formulation and is used for local navigation, interaction, and collision avoidance. The continuum model accounts for macroscopic behaviors, including crowd orientation and flow. We use our hybrid model with particle filters to compute the trajectories at interactive rates. We demonstrate its performance in moderately-dense crowd videos with tens of pedestrians and highlight the improved accuracy on different datasets.**

I. INTRODUCTION

Tracking of pedestrians in a crowded scene is important to ensure safe navigation of autonomous mobile robots and vehicles. With recent advances in self-driving vehicles and autonomous wheelchairs, it is important to track the pedestrians to compute smooth, collision-free trajectories [1], [13]. As self-driving cars are increasingly deployed on urban streets and autonomous wheelschairs are used in public places (e.g. malls or airports), it is important to accurately compute the pedestrian trajectories at interactive rates (e.g. tens of milliseconds).

Besides robotics, pedestrian tracking is also used for surveillance, disaster prevention, and data-driven crowd simulation in virtual environments. This problem has been extensively studied in robotics, computer vision, and image processing. Despite recent advances, it remains difficult to track pedestrians at interactive rates in real-world videos, especially for moderately dense crowds. Some of the challenges in pedestrian tracking arise due to inter-pedestrian occlusion, changes in lighting conditions or pedestrian appearance, accurate modeling of intent or goal position of each pedestrian. In this paper, we restrict ourselves to online and realtime trackers [8], [10], [22]–[24], [33], which compute the trajectories based on current and prior frames. Many of these trackers use motion priors to update the trajectories of the pedestrians between successive frames, then propagate the search space from one frame to the next. The simplest algorithms to model the motion are based on constant velocity or constant acceleration formulations. However, these techniques are unable to model the interaction between the pedestrians, as the *crowd density* (i.e. the number of human agents per squared meter) increases.

In real-world scenarios, the trajectory of each pedestrian is governed by its intermediate goal location, crowd flow, and by interactions with other pedestrians and obstacles in the scene. In a dense crowd, the behavior of each pedestrian changes in response to the environment, including crowd density and overall flow. Many studies in pedestrian dynamics and psychology literature have suggested that the pedestrian movement varies based on based on the crowd density or the *Fundamental Diagram* [7], [19], [40]. As a result it is hard to model all these pedestrian interactions and movements using a single or uniform motion model.

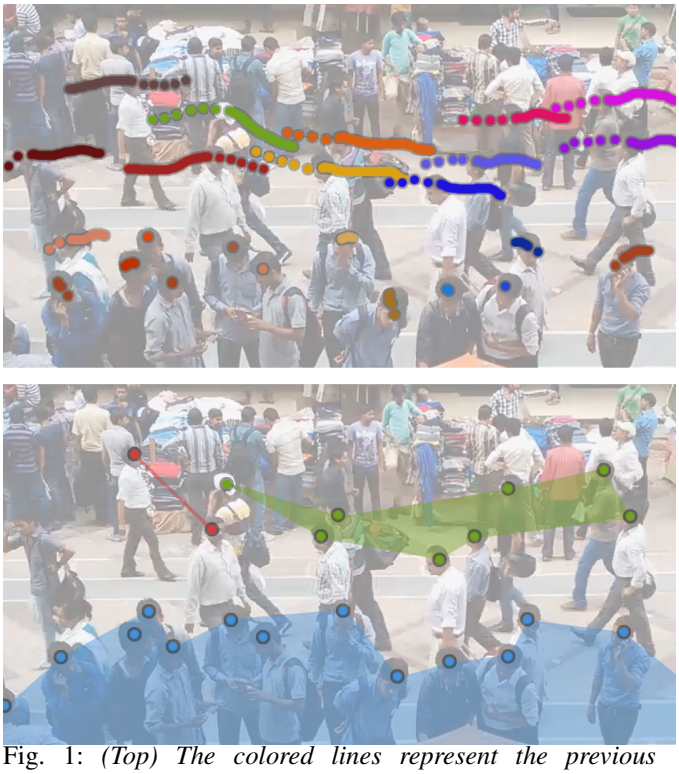

Fig. 1: *(Top) The colored lines represent the previous and current frame positions; the dotted lines represent the corresponding predicted future positions using the discrete motion models used by prior pedestrian tracking algorithms. (Bottom) The colored dots represent pedestrians belonging to the same crowd "cluster"; we track the movement of each cluster using continuum methods. Our overall hybrid algorithm combines discrete and continuous motion models and can result in tracking accuracy of more than 70%.*

Main Results: We present a hybrid motion model for online pedestrian tracking in medium to high-density crowd videos. Our work builds on prior research on simulation models in pedestrian dynamics, robotics, and computer graphics. Specifically, our hybrid model combines discrete

[1]Aniket Bera and Dinesh Manocha are with the Department of Computer Science at the University of North Carolina at Chapel Hill, Chapel Hill, 27516-3175, NC, USA {ab, dm}@cs.unc.edu

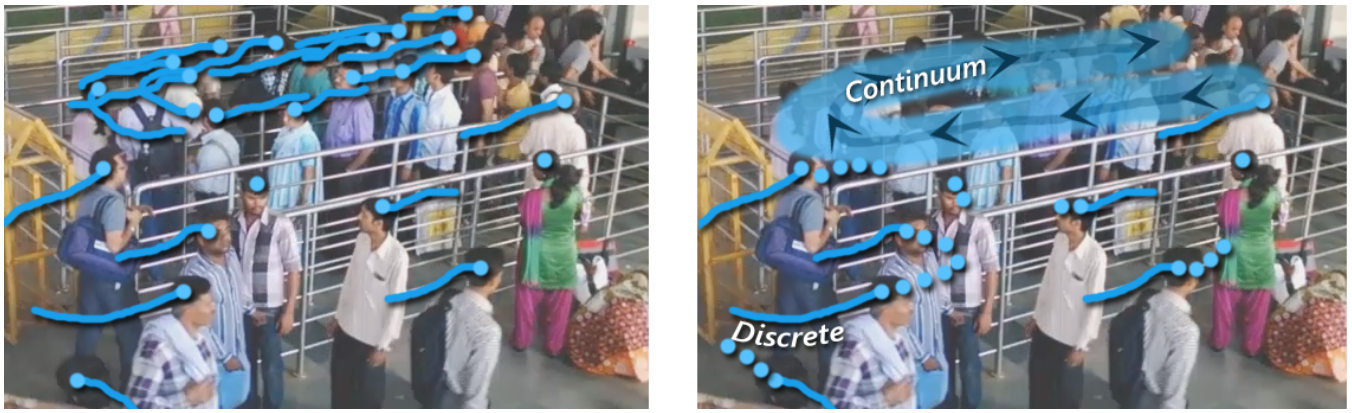

Fig. 2: *The left image highlights the tracked trajectories based on discrete motion models. The image on the right demonstrates the use of a hybrid motion model, using the continuum method for a cluster of pedestrians as well as discrete motion models for individuals. These clusters are computed in realtime based on frame coherence and pedestrian flow. The hybrid motion model can improve the tracking accuracy in these dense scenarios by 7-12% over prior methods.*

(microscopic) and continuum (macroscopic) models. The discrete model is used to predict the local interactions and collision-avoidance behaviors of each pedestrian. The continuum method is used to model the flow of homogeneous clusters within a crowd. Our primary contributions include:

- We cluster pedestrians in a crowd based on different characteristics including their positions, velocity, inter-pedestrian distance, orientations, etc.
- We model the trajectory of each large cluster of pedestrians using a continuum flow model.
- We model the motion of small clusters and individual pedestrians using an adaptive microscopic multi-agent algorithm.
- We combine these discrete and continuum models with particle filters, enabling us to track the pedestrians at interactive rates.

Our algorithm can track tens of pedestrians in medium- to high-density crowds at realtime rates (i.e. more than 25fps) on a multi-core PC and can achieve accuracy of ~65-80% on moderately dense crowd videos. We also compare the accuracy and runtime performance of our hybrid motion-prior algorithm with prior techniques.

The rest of the paper is organized as follows. Section II reviews related work in tracking and motion models. Section III introduces our notation and terminology, and gives an overview of our hybrid motion model. We describe the overall tracking algorithm in Section IV and highlight its performance on different benchmarks in Section V.

II. Related Work

In this section, we briefly review some prior work on pedestrian tracking. Multi-pedestrian tracking has attracted a lot of attention in recent years, and many offline and online tracking algorithms have been proposed [11], [37], [39]. In this section, we limit our discussion to the use of motion models in online and realtime tracking algorithms.

The problem of modeling pedestrian behaviors and trajectories has received significant attention in various disciplines and a number of motion models have been proposed. These include discrete and continuum models. In discrete models, pedestrian agents regard other pedestrians as moving obstacles and try to compute a trajectory that avoids collisions with all other obstacles in the environment (including other pedestrians). Many discrete motion models represent each individual or pedestrian in a crowd as particles (or as 2D circles in a plane) to model the interactions. These include models based on simple interaction rules [32], repulsive forces [16] and velocity-based optimization algorithms [20], [31], [35]. Other recently-developed discrete approaches are based on cognitive models [9], affordance [12], short-term planning using a discrete approach [2], linear trajectory avoidance (LTA) [29], or perceptual models [28].

In continuum methods, agents are clustered, and the density and velocity field of each cluster is computed by accumulating the individual agents' positions and velocities [15], [18]. Treuille et al. [34] used a global approach based on continuum theory for the flow of pedestrians. Narain et al. [27] used a dual representation that uses discrete agents as well as a continuous formulation based on a variational constraint.

There has been considerable work on improved motion models to increase the accuracy of these trackers [4]–[6], [25] but these approaches only use discrete motion models and and may not be able to accurately capture continuum crowd flows.

III. Our Approach

In this section, we give an overview of our approach. We also introduce the notation and terminology used in the paper.

A. Overview

We compute large, homogeneous clusters of pedestrians. We use continuum techniques to model the flow of pedestrians in large clusters. The motion for the rest of pedestrians is modeled using discrete, microscopic models. We use an optimization technique to compute the pedestrian model of each discrete agent. Similarly, we compute the flow of every

cluster based on continuum crowd behaviors [18]. Finally, we combine these motion model predictions with standard particle filter-based trackers.

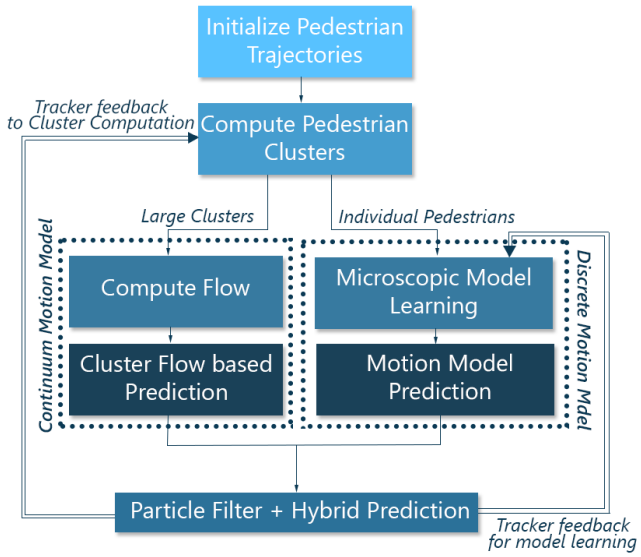

Fig. 3: *Our Crowd Tracking Algorithm: We start with an initial set of trajectories and compute the pedestrian clusters. Based on the cluster size, we compute either the best discrete motion model or the combined motion flow of the cluster. We combine the predictions from the discrete and continuum models and integrate them with particle filter-based trackers. We also use the positions computed by the tracker for computing the clusters and discrete motion models for future frames.*

B. Notation and Terminology

We use the following notations in our paper:

- S represents the state (position and velocity) of an arbitrary pedestrian as computed by the overall tracker.
- X represents the state (position and velocity) of an arbitrary pedestrian inside a motion model.
- Y represents the state (position and velocity) of an arbitrary pedestrian in a cluster C.
- f_n represents the n^{th} discrete motion model from a collection of motion models (as explained later in section IV).
- m represents the "best configured" motion model from the mixture of motion models $\{f_1, f_2, ...\}$.
- **bold fonts** are used to represent values for all the pedestrians in the crowd; for example $\mathbf{S}$ represents the states (positions and velocities) of all pedestrians as computed by the tracker.
- subscripts are used to indicate time; for example m_t represents the "best configured" motion model at timestep t, and $\mathbf{S}_{t-k:t}$ represents all states of all agents for all successive timesteps between $t-k$ and t, as computed by the tracker.

We use an optimization scheme to compute the "best" motion model for each pedestrian. This computed motion model can be used as follows: $X_{t+1} = m_t(X_t)$ or $\mathbf{X}_{t+1} = m_t(\mathbf{X}_t)$ to compute the motion of one arbitrary pedestrian or all pedestrians, respectively.

Data Representation: We use two different pedestrian data representations. The first keeps track of the state (position and velocity) of each pedestrian for the last k timesteps or frames. These are referred to as the k-states of each pedestrian. These k-states are initialized by pre-computing the states from the first k timesteps. The k-states are updated at each timestep by maintaining a queue data structure for each agent: removing the agents' state from the oldest frame and adding the latest tracker-estimated state. In our formulation, we define a 'cluster' as a group of pedestrians that are close and share similar characteristics like walking speed, orientation. We also keep track of cluster density and the flow of different pedestrian formations in the crowd. Depending on the cluster size, we choose between modeling the motion of a pedestrian using a discrete motion model or a continuum model (Pedestrian Flow). In the case of a continuum model, we also represent the clusters along with the associated pedestrians.

The discrete (microscopic) model is chosen by selecting the best motion model from the known multi-agent pedestrian motion models, including Boids [32], RVO [35], Social forces [16], etc. This microscopic mixture motion model is then used to compute the best motion model for the agents during each frame. First, we compute the optimal motion parameters of every discrete motion model to best match the recent k-states data and select the model that best matches a specific metric. Second, we use the "best optimized" motion model to make a prediction on the agents' next state.

Pedestrian Flow is a flow vector of a cluster (see Section IV (a)),and computed using the clusters' average flow using a bottom-up approach. This "flow" captures the movement of the pedestrians in a certain direction.

C. Particle Filters for Tracking

Although we could use any online tracker which requires a motion-prior model, we chose particle filters as our underlying tracking algorithm. The particle filter is a parametric method that solves non-Gaussian and non-linear state estimation problems [3]. Particle filters are frequently used for object tracking, since they can recover from lost tracks and occlusions. The particle tracker's tracking uncertainty is represented in a Markovian manner: it considers only information from present and past frames. The motion priors are used to estimate the next state of each agent. This tracker also employs a confidence estimation scheme that dynamically computes the number of particles for each agent; this confidence estimation allows the particle filter to balance runtime cost with accuracy.

IV. Algorithm

In this section we give details of the various stages of our algorithm (shown in Figure 3). Our hybrid motion model

consists of two parts: the discrete, or microscopic, model and the continuum model. Both these models return a future pedestrian state. Here are the basic components of our approach:

1) Computing Pedestrian Clusters: Our algorithm identifies pedestrian clusters based on a bottom-up hierarchical clustering approach. We initially assign each pedestrian to a separate cluster, one consisting of a single pedestrian. We then merge these clusters by analyzing their relative velocities and their geometric proximity, which is a function of the Euclidean distance between the clusters, the speed of each agent, and their motion. In our experiments, we found that a bottom-up approach is more efficient than a top-down approach for crowds composed of small clusters.

We improve on the group-expand procedure of [26] by including many additional crowd features for clustering the pedestrians. A connectivity graph is constructed [14] among the pedestrians and we measure the graph density based on intra-cluster proximity.

We compute a cluster graph for each cluster. For any cluster $l \geq 1$, the vertices of the connectivity graph CG_l correspond to the pedestrians in the cluster. There is an edge between vertex n_i and n_j if and only if pedestrian i and pedestrian j are together for some period of time and their velocities are close to each other. The density of this graph helps us define intra-cluster proximity as follows. Let e_l be the total number of edges in CG_l and $\hat{e}_{l+1}$ be the minimal number of edges desired in CG_{l+1} after including pedestrian p_i in CG_l. A pedestrian i can be added to an existing cluster of size l if and only if it is connected with at least half of the existing pedestrians in the cluster, i.e., the degree of $n_i \geq \lceil \frac{l}{2} \rceil$. We then have $\hat{e}_{l+1} = e_l + \lceil \frac{l}{2} \rceil$. By definition, $e_1 = \hat{e}_1 = 0$. For $l \geq 1$, given the basis condition that $\hat{e}_2 = 1$ and $\hat{e}_3 = 2$, we derive

$$\hat{e}_l = \begin{cases} (\frac{l}{2})^2 & \text{when } l \text{ is even,} \\ \frac{l-1}{2}(1 + \frac{l-1}{2}) & \text{when } l \text{ is odd.} \end{cases} \quad (1)$$

Two clusters CG_m and CG_n satisfy the intra-cluster proximity criterion if and only if

$$e_{m+n} \geq (\hat{e}_{m+n} + e_m - \hat{e}_m + e_n - \hat{e}_n) \quad (2)$$

2) Microscopic Representation: For each individual pedestrian, we compute the motion model that best fits its position as tracked over recent frames. In essence, our approach chooses the "best" discrete motion model from a fixed set of choices. In this case, the "best" motion model is the one that most accurately matches agents' immediately past states, as per a given error metric.This is different from most models, which are based on specific rules and ideas; a model based on a single idea cannot take into account different pedestrian interactions(such as intermediate goal locations, intrinsic behaviors, and local interactions with other pedestrians and obstacles in the scene). For the sake of simplicity, in this section we will refer to 'discrete motion models' simply as 'motion models'.

This "best" motion model is determined by an optimization framework, which automatically finds the motion model parameters that minimize an error metric. Wolinski et al. [36] designed an optimization framework for evaluating crowd motion models that computes the optimal motion model parameters in an offline manner for a single homogeneous simulation model. Instead we use a realtime approach, compute the optimal motion model every few frames and choose the optimal pedestrian parameters (e.g. size, velocity, or force) for that motion model. This computation can be performed at realtime rates.

A motion model is defined as an algorithm f which starts with a collection of agent states $\mathbf{X}_t$ and derives new states $\mathbf{X}_{t+1}$ for these agents. It represents their motion over a timestep towards the agents' immediate goals $\mathbf{G}$:

$$\mathbf{X}_{t+1} = f(\mathbf{X}_t, \mathbf{G}, \mathbf{P}), \quad (3)$$

where $\mathbf{P}$ denotes the individual pedestrian parameters.

Our mixture motion model can include any generic discrete motion model that can be represented as Equation (3), but in our system we have currently used the following discrete, microscopic motion models -

- Reciprocal Velocity Obstacles [35]: This is a local collision-avoidance and navigation algorithm. Given each agent's state at a certain timestep, it computes a collision-free state for the next timestep.
- Boids Model [32]: It computes appropriate forces between the agents. When the predicted distance between the agents becomes too low, a separation force is computed and added to an attraction force that is pulling each agent towards its goal.
- Social Forces Model [16]: This uses attractive and repulsive forces for each agent. The interactions between the pedestrians are modeled using repulsive forces and attraction forces are applied on each agent towards their goal positions.

Formally, at any timestep t, we define the agents' (k+1)-states (as computed by the tracker) $\mathbf{S}_{t-k:t}$:

$$\mathbf{S}_{t-k:t} = \bigcup_{i=t-k}^{t} \mathbf{S}_i. \quad (4)$$

Similarly, a motion model's corresponding computed agents' states $f(\mathbf{S}_{t-k:t}, \mathbf{P})$ can be defined as:

$$f(\mathbf{S}_{t-k:t}, \mathbf{P}) = \bigcup_{i=t-k}^{t} f(\mathbf{X}_i, \mathbf{G}, \mathbf{P}), \quad (5)$$

initialized with $\mathbf{X}_{t-k} = \mathbf{S}_{t-k}$ and $\mathbf{G} = \mathbf{S}_t$.

At timestep t, considering the agents' k-states $\mathbf{S}_{t-k:t}$, computed states $f(\mathbf{S}_{t-k:t}, \mathbf{P})$ and a user-defined error metric $error()$, our algorithm computes:

$$\mathbf{P}_t^{opt,f} = \underset{\mathbf{P}}{\operatorname{argmin}}\, error(f(\mathbf{S}_{t-k:t}, \mathbf{P}), \mathbf{S}_{t-k:t}), \quad (6)$$

where $\mathbf{P}_t^{opt,f}$ is the parameter set which, at timestep t, results in the closest match between the states computed by the motion algorithm f and the agents' k-states.

For several motion algorithms $\{f_1, f_2, ...\}$, we can compute the algorithm which best matches the agents' k-states $\mathbf{S}_{t-k:t}$ at timestep t:

$$m_t = f_t^{opt} = \underset{f}{\operatorname{argmin}}\, error(f(\mathbf{S}_{t-k:t}, \mathbf{P}_t^{opt,f}), \mathbf{S}_{t-k:t}), \tag{7}$$

and consequently, the best (as per the error in the $error()$ metric itself) prediction for the agents' next state obtainable from the motion algorithms for timestep $t+1$ is:

$$\mathbf{X}_{t+1} = m_t(\mathbf{S}_t). \tag{8}$$

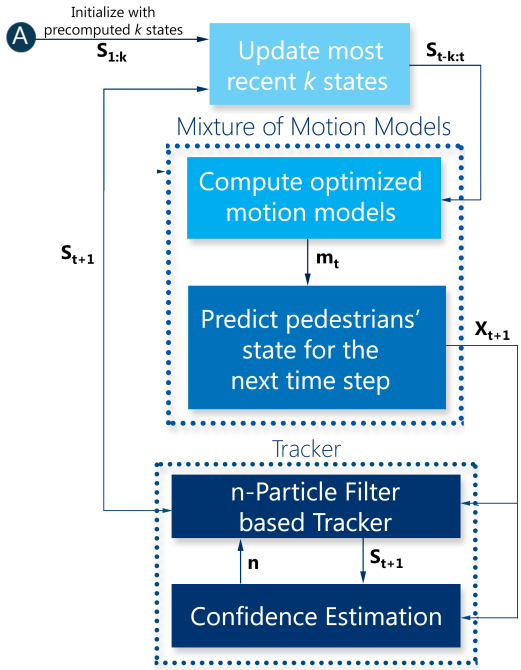

Fig. 4: *Our Macroscopic Model Algorithm/ Mixture of Motion Models: The symbols used in this figure are explained in Section III-B. We use the trajectory computed over prior k frames, expressed as a succession of states, to compute the new motion model; we use our mixture motion model to compute next states using a particle filter based on an optimization framework. The confidence estimation module adapts the number of particles used in our system to optimize computational overhead.*

In total we tested three global optimization approaches: *Greedy algorithm*, *Simulated Annealing*, and *Genetic Algorithm* to optimize our pedestrian parameters P to best-fit prior tracker output. In our final results we use the genetic algorithm as the underlying optimization technique, because it offers the best compromise between the optimization results and runtime performance; efficiency is especially important since our goal is realtime pedestrian tracking. Genetic algorithms seek to overcome the problem of local minima in optimization. This is accomplished by keeping a pool of parameter sets and, during each iteration of the optimization process, creating a new pool of potential solutions by combining and modifying these parameter sets.

During each iteration, our algorithm evaluates and ranks all possible parameter sets (i.e. solutions). If a few iterations of the algorithm offer no improvement, the overall algorithm terminates. Otherwise, individual parameter values in each solution have a probability of being iterated. If so, this iteration has a probability of either being a crossover or a mutation. If it is a crossover, a value from the corresponding parameter from a better-ranked solution is selected; if it is a mutation, a new value is sampled from a probability distribution.

An error metric is also needed to compute the term in Equation (6). In our case, we've chosen a metric that simply computes the average 2-norm between the observed agent positions and the tracker-computed positions. This metric is well adapted to our formulation as the number of considered past frames, k, is relatively small (around 10). This means that it is difficult for the simulated agents to stray too far from the observed trajectories and the metric can be effectively used to compare the observed and computed trajectories. Formally, this error metric is defined at timestep t as follows:

$$\sum_{i=t-k}^{t} \|\mathbf{S}_i - \mathbf{X}_i\|. \tag{9}$$

3) Continuum Representation: After estimating every cluster, we calculate the flow per cluster based on Hughes et al. [18] continuum crowd behavior. To derive the equations that govern the pedestrian flow, we need to combine the unsteady continuity equation [18] with the following three hypotheses that govern the pedestrian motion:

- The speed at which pedestrians walk is determined by the density of surrounding pedestrians.
- Pedestrians have a common sense of the task of reaching their common destination, such that any two individuals at different locations having the same potential would see no advantage to exchanging places
- Pedestrians seek to minimize their estimated travel time (start position to destination time) but change this behavior this behavior to avoid extreme densities. This tempering is assumed to be separable, such that pedestrians minimize the product of their travel time as a function of density.

The above hypotheses lead to the basic governing equations for the flow of a single pedestrian. These equations are

$$-\frac{\delta\rho}{\delta t} + \frac{\delta\rho g(\rho) f^2(\rho)\frac{\delta\varphi}{\delta x}}{\delta x} + \frac{\delta\rho g(\rho) f^2(\rho)\frac{\delta\varphi}{\delta y}}{\delta y} = 0, \tag{10}$$

and

$$g(\rho)f(\rho) = \frac{1}{\sqrt{(\frac{\delta\varphi}{\delta x})^2 + (\frac{\delta\varphi}{\delta y})^2}}, \tag{11}$$

where φ is the remaining travel time, which is a measure of the instantaneous goal, ρ is the density of the crowd,

$f(\rho)$ is the speed of pedestrians as a function of density, $g(\rho)$ is a factor related to the preferred velocity at a given density, and (x, y, t) denotes the horizontal space and time coordinates. This represents our flow state Y_t. Derivation of these equations and relevant details are given in [17].

4) Tracking: We consider the combination of "best configured" motion model m_t and the cluster flow from Y_t, as well as the error Q_t in the prediction that this "best configured" motion model has generated. Our tracker's observations can be represented by a function $h()$ that projects the state X_t or Y_t (depending on what motion model was used) to a previously computed state S_t. We denote the error between the observed states and the ground truth as R_t. We can now phrase them formally in terms of a standard particle filter as below. In all the equations below X_t can be replaced Y_t.

$$S_{t+1} = m_t(X_t) + Q_t, \tag{12}$$

$$S_t = h(X_t) + R_t. \tag{13}$$

Particle filtering is a Monte Carlo approximation to the optimal Bayesian filter, which monitors the posterior probability of a first-order Markov process:

$$p(X_t|S_{t-k:t}) = \alpha p(S_t|X_t) \int_{X_{t-1}} p(X_t|X_{t-1}) p(X_{t-1}|S_{t-k:t-1}) dX_{t-1}, \tag{14}$$

where X_t is the process state at time t; S_t is the observation; $S_{t-k:t}$ is all of the observations through time t; $p(X_t|X_{t-1})$ is the process dynamical distribution; $p(S_t, X_t)$ is the observation likelihood distribution; and α is the normalization factor. Since the integral does not have a closed form solution in most cases, particle filtering approximates the integration using a set of weighted samples $X_t^{(i)}, \pi_t^{(i)}{}_{i=1,\ldots,n}$, where $X_t^{(i)}$ is an instantiation of the process state, known as a particle, and $\pi_t^{(i)}$'s are the corresponding particle weights. With this representation, the Monte Carlo approximation to the Bayesian filtering equation is:

$$p(X_t|S_{t-k:t}) \approx \alpha p(S_t|X_t) \sum_{i=1}^{n} \pi_{t-1}^{(i)} p(X_t^{(i)})|p(X_{t-1}^{(i)}), \tag{15}$$

where n refers to the number of particles.

In our formulation, we use the motion model to infer dynamic transition, $p(X_t|X_{t-1})$, for particle filtering.

V. Results

In this section, we describe our implementation and highlight the results on different crowd datasets. We implemented these algorithms on a Intel©Haswell, Core®i7- 4771 Processor (8 Cores) with an 8MB Cache, 3.90 GHz and Intel©HD Graphics 4600. Our approach is implemented in C++, and some components uses OpenMP and OpenCL for multi-core implementations. Specifically, we perform agent-level parallelism: individual pedestrian tracking or cluster tracking computations are distributed across the CPU cores, except for motion-model computation, where pedestrian behavior is inter-linked and the computation is mostly sequential. An average of 7-8% of the total computation time is spent on discrete motion-model computations, 4-6% on continuum model computations (including the cost of computing the cluster), and the rest is used by the particle filter based tracker.

A. Evaluation

We use the **CLEAR MOT** [21] evaluation metrics to analyze our algorithm's performance. We use the **MOTP** and the **MOTA** metrics. **MOTP** evaluates the alignment of tracks with the ground truth, while **MOTA** produces a score based on the number of false positives, missed detections, and identity switches. These metrics have become standard for evaluating detection and tracking algorithms, and we refer the interested reader to [21].

We analyze these metrics across the density groups and the different motion models (Table III).

B. Tracking Results

We highlight the performance of our algorithm based on the hybrid model on different benchmarks, comparing the performance of our algorithm with single, homogeneous motion model methods: constant velocity model (LIN), LTA [29], Social Forces [38], Boids [32] and RVO [35]. We also compare our approach using only a flow-based continuum model [17]. LIN models the velocities of pedestrians as constant, and is the underlying motion model frequently used in the standard particle filter. In our implementation, we replace the state transition process of a standard particle filtering algorithm with different motion models.

We evaluate on some challenging datasets [6] which are available publicly and also some standard datasets [30] from the pedestrian tracking community. These videos were recorded at 24-30 fps. We manually annotated these videos and corrected the perspective effect by camera calibration. We also compare our algorithm's performance to that of a baseline mean-shift tracker (Table IV). We show the number of correctly tracked pedestrians and the number of ID switches. A track is counted as "successful" when the estimated mean error between the tracking result and the ground-truth value is less than 0.8 meter in groundspace; this value comes from the average human stride length (about 0.8 meter), and we consider the tracking to be incorrect if the mean error is more than this value. Our method provides 9-18% higher accuracy over LIN for medium density crowds (Table IV). We also compare the performance of our hybrid tracking algorithm with that of a particle filter using only a discrete motion model and with that of a continuum-only model (Table II).

C. Benefits of REACH

Our method offers realtime performance on multi-core desktop PC and has higher accuracy than other state-of-the-art online-tracking algorithms that use a single, homogeneous motion algorithm to model the tracking prior.

Dataset	Challenges	Density	Agents
NDLS-1	BV, PO, IC	High	131
IITF-1	BV, PO, IC, CO	High	167
IITF-3	BV, PO, IC, CO	High	189
IITF-5	BV, PO, IC, CO	High	71
NPLC-1	BV, PO, IC	Medium	79
NPLC-3	BV, PO, IC, CO	Medium	144
IITF-2	BV, PO, IC, CO	Medium	68

Dataset	Challenges	Density	Agents
IITF-4	BV, PO, IC, CO	Medium	116
NDLS-2	BV, PO, IC, CO	Low	72
NPLC-2	BV, PO	Low	56
seq_hotel	IC, PO	Low	390
seq_eth	BV, IC, PO	Low	360
zara01	BV, IC, PO	Low	148
zara02	BV, IC, PO	Low	204

TABLE I: Crowd Scenes used as Benchmarks. We highlight many attributes of these pedestrian tracking datasets, along with density and the number of number of pedestrians tracked. We use the following abbreviations about some characteristics of the underlying scene: Background Variations (BV), Partial Occlusion (PO), Complete Occlusion (CO) and Illumination Changes (IC)

	High Density								Medium Density					
	NDLS-1		**IITF-1**		**IITF-3**		**IITF-5**		**NPLC-1**		**NPLC-3**		**IITF-2**	
	ST	FPS	ST	FPS	ST	FPS	ST	FPS	ST	FPS	ST	FPS	ST	FPS
Discrete (Microscopic) Model Only	60	28	70	29	51	27	68	27	71	28	69	26	40	26
Continuum (Macroscopic) Model Only	55	28	67	29	49	29	62	26	70	29	67	28	32	28
REACH (Our Hybrid Motion Model)	63	27	73	28	57	26	67	26	77	28	71	26	44	26

	Medium Density		Low Density											
	IITF-4		**NDLS-2**		**NPLC-2**		**seq_hotel**		**seq_eth**		**zara01**		**zara02**	
	ST	FPS	ST	FPS	ST	FPS	ST	FPS	ST	FPS	ST	FPS	ST	FPS
Discrete (Microscopic) Model Only	58	27	72	28	75	26	254	29	268	29	61	27	63	29
Continuum (Macroscopic) Model Only	55	27	69	28	71	27	238	29	259	28	59	28	60	26
REACH (Our Hybrid Motion Model)	63	27	79	28	78	26	252	28	267	29	63	27	68	28

TABLE II: We compare the number of successful tracks (ST) and average tracking frames per second (FPS) of three motion models: only discrete-motion model, only continuum-motion model, and our hybrid motion model. We combine each of these with adaptive particle filtering.

	LIN			Boids			Helbing			RVO			Continuum Model			**REACH**		
	LD	MD	HD	LD	MD	HD	LD	MD	HD	LD	MD	HD	LD	MD	HD	**LD**	**MD**	**HD**
MOTP	64.42%	52.82%	33.21%	67.24%	57.10%	43.14%	70.52%	61.33%	49.88%	72.19%	63.17%	51.31%	70.14%	61.32%	48.20%	**73.98%**	**69.23%**	**54.29%**
MOTA	49.42%	35.3%	31.37%	50.59%	26.42%	40.88%	53.28%	44.19%	33.51%	53.95%	48.81%	35.83%	52.93%	42.32%	31.29%	**54.18%**	**50.16%**	**38.83%**

TABLE III: We compare the MOTA and MOTP values across the density groups and the different motion models (LD is Low Density, MD is Medium Density and HD is High Density).

	High Density								Medium Density								Low Density			
	NDLS-1		**IITF-1**		**IITF-3**		**IITF-5**		**NPLC-1**		**NPLC-3**		**IITF-2**		**IITF-4**		**NDLS-2**		**NPLC-2**	
	ST	IS	ST	IS	ST	IS	ST	IS	ST	IS	ST	IS	ST	IS	ST	IS	ST	IS	ST	IS
LIN	53	17	63	27	51	35	59	18	67	15	60	29	36	22	52	36	68	23	69	21
Boids	58	15	66	23	56	33	65	14	73	13	65	26	40	19	52	35	70	22	72	19
Helbing	56	16	66	26	52	33	62	15	74	11	68	23	41	19	59	31	75	18	72	14
LTA	54	17	65	22	51	32	60	17	68	11	62	28	42	18	54	32	69	23	70	20
RVO	57	14	69	20	53	29	64	13	71	10	64	26	42	18	53	32	72	20	74	16
Continuum (Flow-based)	55	19	67	24	49	33	62	17	70	11	67	28	32	18	55	32	69	27	71	20
MeanShift	27	32	31	38	23	52	34	29	39	36	41	31	22	33	39	45	31	28	45	28
REACH	63	12	73	19	57	27	67	10	77	7	71	20	44	16	63	28	79	17	78	14

TABLE IV: We compare the number of successful tracks (ST) and ID switches (IS) of our hybrid motion model with prior homogeneous motion models - LIN, Boids, Helbing, LTA, RVO, a baseline mean-shift tracker and a continuum (flow-based) model. Our hybrid motion model provides higher accuracy compared to homogeneous motion models and lesser ID switches. These crowd datasets are publicly available at http://gamma.cs.unc.edu/RCrowdT/.

VI. Limitations, Conclusions, and Future Work

We present a realtime algorithm for pedestrian tracking in moderately-dense crowd videos. Our algorithm includes a new hybrid motion model that can capture varying pedestrian behaviors and interactions. We have demonstrated our algorithm's performance on well-known benchmarks and highlighted the improvements it offers over prior methods.

Our formulation does not, however, take into account physiological and psychological pedestrian traits. All pedestrians are modeled as though they share the same sensitivities and behaviors (including density preferences, age, and gender) with all other pedestrians in the crowd. Our algorithm also does not take into account heterogeneous agent characteristics such as variations in personality, which also affect overall pedestrian behavior and movement.

As part of future work, we would like to incorporate pedestrian personality characteristics and other characteristics from pedestrian dynamics, such as 'fundamental diagrams.'. We would like to parallelize the approach on a GPU to handle more complex pedestrian datasets in realtime. We would also like to implement our pedestrian tracker on mobile platforms and integrate it with various consumer-grade robots to perform autonomous navigation.

	seq_hotel		seq_eth		zara01		zara02	
	ST	IS	ST	IS	ST	IS	ST	IS
LIN	182	92	187	58	51	27	49	27
Boids	192	78	202	59	52	27	54	26
Helbing	221	73	232	48	54	26	55	25
LTA	238	70	249	42	60	24	62	25
RVO	241	71	258	37	61	22	65	23
Continuum (Flow-based)	238	71	259	36	59	22	60	24
MeanShift	98	171	112	139	32	41	33	39
REACH	252	68	267	34	63	20	68	21

TABLE V: We compare the number of successful tracks (ST) and ID switches (IS) of our mix motion model algorithm (MMM) with discrete motion models: LIN, Boids, Helbing, LTA, RVO, a baseline mean-shift tracker and only using a continuum flow-based model with standard datasets - seq_hotel, seq_eth, zara01, zara02 [30].

VII. Acknowledgements

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

References

[1] A Pedro Aguiar and Joao P Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *Automatic Control, IEEE Transactions on*, 52(8):1362–1379, 2007.

[2] Gianluca Antonini, Santiago Venegas Martinez, Michel Bierlaire, and Jean Philippe Thiran. Behavioral priors for detection and tracking of pedestrians in video sequences. *INT. J. COMPUT. VIS*, 69(2):159–180, 2006.

[3] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 2002.

[4] Aniket Bera, Nico Galoppo, Dillon Sharlet, Adam Lake, and Dinesh Manocha. Adapt: real-time adaptive pedestrian tracking for crowded scenes. In *ICRA*, 2014.

[5] Aniket Bera, Sujeong Kim, and Dinesh Manocha. Efficient trajectory extraction and parameter learning for data-driven crowd simulation. In *Proc. of Graphics Interface*, 2015.

[6] Aniket Bera and Dinesh Manocha. Realtime multilevel crowd tracking using reciprocal velocity obstacles. In *ICPR*, 2014.

[7] A Best, S Narang, S Curtis, and D Manocha. Densesense: Interactive crowd simulation using density-dependent filters. In *SCA*, pages 97–102, 2014.

[8] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE PAMI*, 2011.

[9] Shu-Yun Chung and Han-Pang Huang. A mobile robot that understands pedestrian spatial behaviors. In *IROS*, 2010.

[10] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000.

[11] Markus Enzweiler and Dariu M Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE PAMI*, 2009.

[12] Brett R. Fajen. Affordance-based control of visually guided action. *ECOLOGICAL PSYCHOLOGY*, 19(4):383–410, 2007.

[13] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[14] Weina Ge, Robert T Collins, and Barry Ruback. Automatically detecting the small group structure of a crowd. In *Applications of computer vision (wacv), 2009 workshop on*, pages 1–8. IEEE, 2009.

[15] Abhinav Golas, Rahul Narain, and Ming Lin. Hybrid long-range collision avoidance for crowd simulation. In *I3D*, pages 29–36, 2013.

[16] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 1995.

[17] Roger L Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, 36(6):507–535, 2002.

[18] Roger L Hughes. The flow of human crowds. *Annual review of fluid mechanics*, 35(1):169–182, 2003.

[19] Verne Thompson Inman, Henry James Ralston, and Frank Todd. *Human walking*. Williams & Wilkins, 1981.

[20] Ioannis Karamouzas, Peter Heil, Pascal van Beek, and Mark H. Overmars. A predictive collision avoidance model for pedestrian simulation. In *Motion in Games*, pages 41–52, 2009.

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

[22] Zia Khan, Tucker Balch, and Frank Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV*. 2004.

[23] Yuan Li, Haizhou Ai, Takayoshi Yamashita, Shihong Lao, and Masato Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *IEEE PAMI*, 2008.

[24] Z Li, QL Tang, and N Sang. Improved mean shift algorithm for occlusion pedestrian tracking. *Electronics Letters*, 2008.

[25] Wenxi Liu, Antoni B Chan, Rynson WH Lau, and Dinesh Manocha. Leveraging long-term predictions and online-learning in agent-based multiple person tracking. *arXiv preprint arXiv:1402.2016*, 2014.

[26] Clark McPhail and Ronald T Wohlstein. Using film to analyze pedestrian behavior. *Sociological Methods & Research*, 10(3):347–375, 1982.

[27] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C Lin. Aggregate dynamics for dense crowd simulation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 122, 2009.

[28] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph.*, 29(4):123–123, July 2010.

[29] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, 2009.

[30] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *ECCV*, pages 452–465. Springer, 2010.

[31] Julien Pettré, Jan Ondřej, Anne-Hélène Olivier, Armel Cretual, and Stéphane Donikian. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *SCA*, pages 189–198, 2009.

[32] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87*, pages 25–34, 1987.

[33] Juan C SanMiguel, Andrea Cavallaro, and José M Martínez. Standalone evaluation of deterministic video tracking. In *ICIP*, 2012.

[34] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1160–1168, 2006.

[35] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *International Symp. on Robotics Research*. 2011.

[36] David Wolinski, Stephen J. Guy, Anne-Hélène Olivier, Ming C. Lin, Dinesh Manocha, and Julien Pettré. Parameter estimation and comparative evaluation of crowd simulations. In *Eurographics*, 2014.

[37] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. pages 2411–2418, 2013.

[38] Kota Yamaguchi, Alexander C Berg, Luis E Ortiz, and Tamara L Berg. Who are you with and where are you going? In *CVPR*, 2011.

[39] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm Computing Surveys (CSUR)*, 2006.

[40] Jun Zhang, Wolfram Klingsch, Andreas Schadschneider, and Armin Seyfried. Transitions in pedestrian fundamental diagrams of straight corridors and t-junctions. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(06):P06004, 2011.