INTERACTIVE TRACKING, PREDICTION, AND BEHAVIOR LEARNING OF PEDESTRIANS IN DENSE CROWDS

Aniket Bera

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2017

Approved by:

Dinesh Manocha

Ming C. Lin

Ron Alterovitz

Nico Galoppo

Carol O'Sullivan

ABSTRACT

Aniket Bera: Interactive Tracking, Prediction, and Behavior Learning of Pedestrians in Dense Crowds.
(Under the direction of Dinesh Manocha)

The ability to automatically recognize human motions and behaviors is a key skill for autonomous machines to exhibit to interact intelligently with a human-inhabited environment. The capabilities autonomous machines should have include computing the motion trajectory of each pedestrian in a crowd, predicting his or her position in the near future, and analyzing the personality characteristics of the pedestrian. Such techniques are frequently used for collision-free robot navigation, data-driven crowd simulation, and crowd surveillance applications. However, prior methods for these problems have been restricted to low-density or sparse crowds where the pedestrian movement is modeled using simple motion models.

In this thesis, we present several interactive algorithms to extract pedestrian trajectories from videos in dense crowds. Our approach combines different pedestrian motion models with particle tracking and mixture models and can obtain an average of 20% improvement in accuracy in medium-density crowds over prior work. We compute the pedestrian dynamics from these trajectories using Bayesian learning techniques and combine them with global methods for long-term pedestrian prediction in densely crowded settings. Finally, we combine these techniques with Personality Trait Theory to automatically classify the dynamic behavior or the personality of a pedestrian based on his or her movements in a crowded scene. The resulting algorithms are robust and can handle sparse and noisy motion trajectories. We demonstrate the benefits of our long-term prediction and behavior classification methods in dense crowds and highlight the benefits over prior techniques.

We highlight the performance of our novel algorithms on three different applications. The first application is interactive data-driven crowd simulation, which includes crowd replication as well as the combination of pedestrian behaviors from different videos. Secondly, we combine the prediction scheme with proxemic characteristics from psychology and use them to perform socially-aware navigation. Finally, we present novel techniques for anomaly detection in low-to medium-density crowd videos using trajectory-level behavior learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ORCA	Optimal Reciprocal Collision Avoidance
PDL	Pedestrian Dynamics Learning
RVO	Reciprocal Velocity Obstacle
MMM	Motion Model Mixture
GLMP	Realtime Pedestrian Path Prediction using **G**lobal and **L**ocal **M**ovement **P**atterns
PTC	Personality Trait computation
GMD	Global Movement Dynamics
GVO	Generalized Velocity Obstacles
PEN	Psychoticism - Extraversion - Neuroticism Model

CHAPTER 1

Introduction

The ability to automatically recognize human movements and activities is key for autonomous machines to interact intelligently with a human-inhabited environment (i.e. humans walking towards their goal and avoiding collisions with other humans and obstacles while interacting with the environment). The possibility of sensing human crowd motion has received considerable attention in the literature. It is a well-studied problem that has applications in many domains. For example, it is considered in computer animation to generate human motion for special effects (Kovar et al., 2002; Badler, 1997a). In virtual environments, it is used to generate movements and interactions for human avatars and virtual agents (Baylor, 2009; Bainbridge, 2007; Badler, 1997b). Surveillance applications include evaluating atypical and suspicious movements that differ from those of the crowd (Wu et al., 2010; Mahadevan et al., 2010). Similarly, behavior modeling discusses this problem as it relates to the analysis of personalities and behaviors based on their interaction in the real world (Cristani et al., 2013; Yi et al., 2015). In other applications, this problem has implications for disaster prevention and planning evacuations, crowd scene analysis (analyzing global characteristics of the crowd), and collision-free navigation of robots or autonomous vehicles in crowded or real-world scenarios.

The following are some specific applications that best illustrate the need for a better pedestrian or crowd motion model:

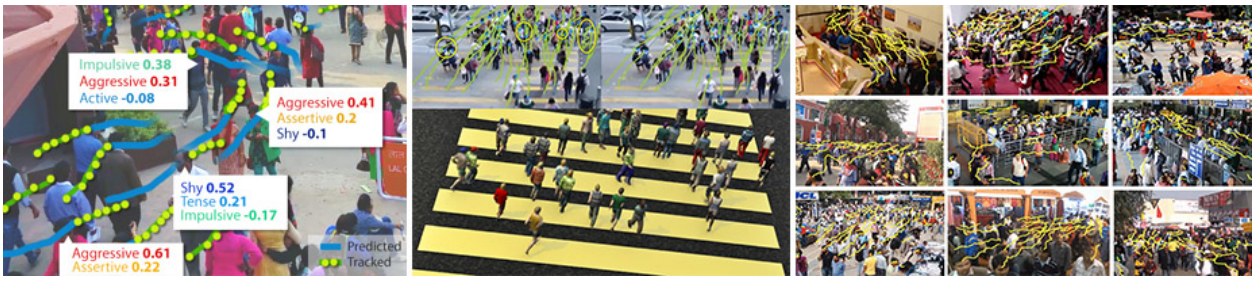


Figure 1.1: The modeling of pedestrian movements has received considerable attention in multiple fields, including computer-aided design, urban planning, robotics, and evacuation planning. In many of these applications, the goal is to capture the trajectories and behaviors of the pedestrians. This image shows a subset of the many applications that model pedestrian movements (e.g., pedestrian movement prediction, personality recognition, data-driven crowd simulation, large-scale tracking, etc.).

- As robots are increasingly used in households, offices, and public places, they navigate among humans or pedestrians more and more. Because humans are dynamic agents (changing directions, positions, etc.), these scenarios result in many new challenges related to navigation and the awareness of human motion and interactions. Robots must move through crowds of people while preventing collisions with each other and humans. In such scenarios, the robots need to interface with not only the physical environment, but also the social environment, and should interact well with pedestrians.

- One of the key challenges in surveillance is to devise methods that can automatically analyze the behavior and movement patterns in crowd videos to detect anomalous or atypical behaviors (Li et al., 2015). Furthermore, since many of the surveillance systems need realtime planning for security, most of these applications benefit from interactive performance, and do not rely on a priori learning or labeling. However, current methods are typically limited to sparse crowds or are designed for offline or non-realtime applications.

- Crowds have also been studied in computer animation for use in generating special effects. Frequently, designers or animators must go through a tedious process to generate scene-specific behavior rules such as events, trajectories, or interactions. To account for these factors, they have to manually generate scene-specific behaviors or trajectories. This can be time consuming and, further, generating realistic behaviors or simulations using such methods involves considerable tweaking of or variations in simulation parameters. As the number of agents or the complexity of the scenarios increases, it becomes increasingly difficult to model the diversity of behaviors or the interactions between the agents.

1.0.1 Pedestrian Movements and Behaviors

From the problems and applications described above, it is apparent that it is important to design and develop algorithms which not only efficiently capture pedestrian motion and behavior, but do it interactively. This interactivity is especially important for applications such as realtime surveillance, autonomous vehicle planning, and robot planning. For this work, we consider capturing pedestrian motion and behavior as a collection of three interconnected components:

- **Tracking:** Locating a pedestrian (or pedestrians) along a window of time. We limit tracking to the projected trajectory on a 2D plane assuming that the pedestrian is represented as a small circle.

Figure 1.2: **Overview:** We highlight the different aspects of movements (tracked trajectories marked with yellow in (1), predicted trajectories as blue dots in (2) and personalities/behaviors in (3)) of pedestrians in the same scene.

- **Prediction:** Determining future pedestrian positions and velocities based on past data. We define short term prediction as future pedestrian positions for 1–2 seconds and long term prediction as future positions for 5 or more seconds.
- **Behavior Learning:** For this work we restrict ourselves to trajectory-level motion patterns and personality traits based on prior work in psychology and various interactions with environment.

We take these three important problems and classify them as a part of '*pedestrian sensing*'. However, sensing pedestrians in a crowded scene is regarded as a difficult problem due to, for example, intra-pedestrian occlusion (i.e. one pedestrian blocking others) and changes in lighting and pedestrian appearance. Similarly, predicting the trajectory of a pedestrian in a dense environment can also be very challenging. In general, pedestrians have varying behaviors and can change their speed to avoid collisions with the obstacles and other pedestrians in a scene. In crowded scenarios, the interactions between the pedestrians tend to increase significantly, which affects their behavior and movement. While researchers in various fields like psychology and other social sciences have been studying and observing human behavior for decades, modeling realistic crowd behaviors is challenging. Further, there are no widely accepted models capable of capturing a wide variety of behaviors. As a result, the highly dynamic nature of pedestrian movement makes it hard to estimate a pedestrians current or future positions.

1.0.2 Motion Models

A motion model is a mathematical model, that defines a set of rules related to the local movement of a pedestrian in a given scenario or environment (Kirchner and Schadschneider, 2002). There is an extensive

body of work in robotics, multi-agent systems, crowd simulation, and computer vision on modeling pedestrian motion in crowded environments. We limit the scope of the motion models discussed in this thesis to those that mainly compute trajectory-level movements or behaviors of each independent individual, which we call the *agent* or *pedestrian*, in the crowd. These models can be broadly classified into the following categories: potential-based models, which model virtual agents in a crowd as particles with potentials and forces (Helbing and Molnar, 1995a); boid-like approaches, which create simple rules to steer agents (Reynolds, 1999a); geometric optimization models, which compute collision-free velocities (Van Den Berg et al., 2011a); and field-based methods, which generate fields based on continuum theory (Treuille et al., 2006). Among these approaches, velocity-based motion models (Karamouzas et al., 2009; Karamouzas and Overmars, 2010; van den Berg et al., 2008b; Van Den Berg et al., 2011a; Pettré et al., 2009) have been successfully applied to the simulation and analysis of crowd behaviors and to multi-robot coordination (Snape et al., 2011). Velocity-based models have also been shown to have efficient implementations that closely match real human paths (Guy et al., 2010). These models form the basis for modeling the pedestrian motion in our work (i.e. pedestrian tracking, prediction, and behavior learning).

1.0.3 Pedestrian Tracking with Motion Models

Tracking pedestrians temporally and spatially in a video is an essential and significant task in any intelligent video surveillance system, because it provides the fundamental information for semantic understanding of the video. Prior work in pedestrian tracking (Cui et al., 2005; Kratz and Nishino, 2011) attempts to improve tracking accuracy by making simple assumptions about pedestrian movement, such as constant velocity and constant acceleration. More recently, *higher-order motion models* and pairwise interaction rules have been combined with tracking to improve the accuracy. (Bruce and Gordon, 2004) and (Gong et al., 2011) first estimate pedestrians' destinations and then predict their motions along the path towards the estimated goal positions. (Liao et al., 2003) extract a Voronoi graph from the environment and predict people's motion along the edges. (Luber et al., 2010) apply Helbing's social force model to track people using a Kalman filter based tracker. (Pellegrini et al., 2009a) use an energy function to build up a goal-directed short-term collision-avoidance motion model. (Yamaguchi et al., 2011a) present a pedestrian-tracking algorithm that uses an agent-based behavioral model called ATTR, with additional social and personal properties learned from the behavioral priors (e.g., grouping information and destination information). Most of these methods

are slow and work offline. Additionally, these methods are well-suited for modeling motion in dense crowds with few distinct motion patterns; however, they may not work in heterogeneous crowds.

1.0.4 Pedestrian Prediction

Crowd trajectory prediction has been studied extensively for robot navigation and related areas. (Fulgenzi et al., 2007) use a probabilistic velocity-obstacle approach combined with dynamic occupancy grid. This method assumes constant linear velocity motion of the obstacles. (Du Toit and Burdick, 2010) present a robot planning framework that accounts for each pedestrian's anticipated future location information to reduce the uncertainty of the predicted belief states. Other techniques use potential-based approaches for robot path planning in dynamic environments (Pradhan et al., 2011). Some methods learn the trajectories from collected data. (Ziebart et al., 2009a) use pedestrian trajectories collected in the environment for prediction using Hidden Markov Models. (Bennewitz et al., 2005) apply Expectation Maximization clustering to learn typical motion patterns from pedestrian trajectories before using Hidden Markov Models to predict future pedestrian motion.(Henry et al., 2010) use reinforced learning from example traces, estimating pedestrian density and flow with a Gaussian process. (Kretzschmar et al., 2014) consider pedestrian trajectories as a mixture probability distribution of a discrete as well as a continuous distribution, and then use Hamiltonian Markov chain Monte Carlo sampling for prediction. (Kuderer et al., 2012) use maximum entropy-based learning to learn pedestrian trajectories and use a hierarchical optimization scheme to predict future trajectories. Many of these methods involve a priori learning and may not work in new or unknown environments. Variations of Bayesian filters for pedestrian path prediction have been studied in (Schneider and Gavrila, 2013; Mogelmose et al., 2015). Most of these methods are not suitable for real-time applications or may not work well for dense crowds. They also tend to fail for trajectories predicted over a longer time (> 5 seconds).

1.0.5 Learning Pedestrian Behaviors

Behavior models provide a way to control local navigation and collision avoidance behaviors of agents. Unlike local interaction behaviors, which can be modeled with mathematical formulations, higher-level behaviors can hardly be formulated due to the complexity of human behaviors. Instead, it has been a common practice to script or manually specify behavior rules for pedestrian movement. For example, cognitive approaches focus on defining rules or behavior to mimic cognition or the decision-making process of the crowd (Shao and Terzopoulos, 2005; Ulicny and Thalmann, 2002). These methods can simulate very specific

and detailed behaviors when used with corresponding scenarios such as buying a ticket or waiting in line. Also, Finite State Machines (FSM) are commonly used to encode procedural behaviors or a set of goals based on an agent's state (Bandini et al., 2006; Paris and Donikian, 2009; Sean Curtis, 2013). There is also recent work on analyzing pedestrian motion patterns on a semantic-level (Wang et al., 2017, 2016).

There is extensive work in computer vision, multimedia, and robotics that analyzes the behaviors and movement patterns in crowd videos, as surveyed in (Li et al., 2015; Borges et al., 2013). The main objectives of these works include human behavior understanding and crowd activity recognition for the detection of abnormal behaviors (Hu et al., 2004). Many of these methods use a large number of training videos to learn the patterns offline (Zen and Ricci, 2011; Solmaz et al., 2012). Other methods utilize motion models to learn crowd behaviors (Pellegrini et al., 2012) or use machine learning algorithms (Zhou et al., 2012b; Cheung et al., 2016). Some techniques focus on classifying the most common behavior patterns in a scene using offline learning. These include activity prototypes using a convex learning algorithm (Zen and Ricci, 2011) and detection of popular behavior patterns like bottlenecks, fountainheads, lanes, arches, and blocking (Solmaz et al., 2012).

Crowd behavior learning using motion or simulation has been used for different applications. Parameter learning has been used to predict pedestrian motion for tracking (Pellegrini et al., 2012). However, these techniques use either manual selection or offline learning techniques to estimate the goal positions. Other researchers have used low-density tracking data to learn agent intentions (Musse et al., 2007) or online Bayesian motion-prediction methods for human-robot interactions, data-driven crowd simulation (Kim et al., 2016), and offline training (Zhong et al., 2015). Different approaches have been used to model pedestrian behavior. (Funge et al., 1999) use cognitive modeling to empower agents to plan and perform high-level tasks (Godoy et al., 2016).

1.0.6 Crowd Density

Crowd density is an important property to consider when analyzing crowd characteristics for different applications. There is no universal classification of crowd densities but for our work, we define low density as crowds with 0–1 pedestrians per squared-meter, medium density as 1–3 pedestrians per squared-meter and more than 3 pedestrians per squared-meter as high density.

Density	**Average pedestrians/m^2**
Low	<1
Medium	1–3
High	>3

Table 1.1: We define our classification for different pedestrian densities

As a general rule, the higher the density the more difficult it becomes to track/predict pedestrian motion. In most cases, the appearance model (algorithm for matching a statistical model of object shape and appearance to a new image) isn't capable of capturing partially-visible pedestrians in a crowd.

Figure 1.3: We show crowds at different densities (pedestrian marked with a red dot). We define low density as crowds with 0–1 pedestrians per squared-meter, medium density as 1–3 pedestrians per squared-meter and more than 3 pedestrians per squared-meter as high density.

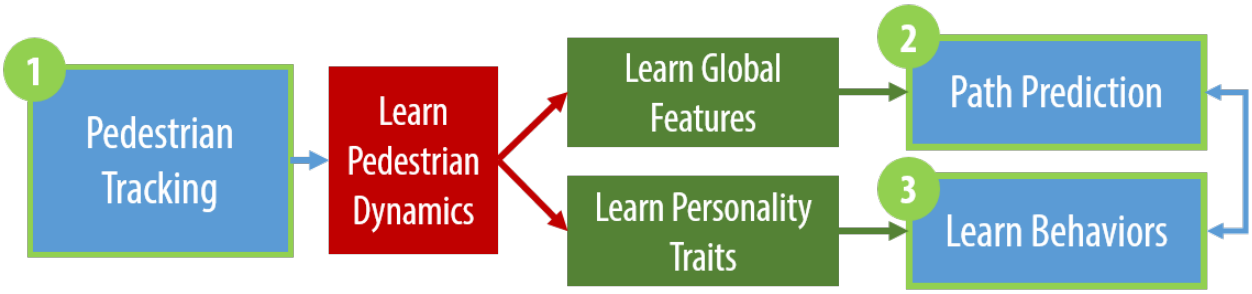


Figure 1.4: **Overview:** In this thesis, we present several interactive algorithms to extract pedestrian trajectories from videos in dense crowds. Our approach combines different pedestrian motion models with particle tracking and a mixture of motion models. We compute the pedestrian dynamics (a collection of different behavior and movement patterns, (Kim & Bera et al. 2014)) from these trajectories using Bayesian learning techniques and combine with global methods for long-term pedestrian prediction in dense crowds. Finally, we combine these techniques with Personality Trait Theory from Psychology to automatically classify the dynamic behavior or personality of a pedestrian based on his or her movements in a crowded scene. The resulting methods are robust and can handle sparse and noisy trajectories. We demonstrate the benefits of our long-term prediction and behavior classification methods in dense crowds and highlight the benefits over prior techniques.

1.1 Thesis Statement

Our thesis statement is as follows:

Higher order motion models and learning techniques can be used to design accurate algorithms for pedestrian tracking, long term prediction and behavior learning.

In this thesis, we propose interactive algorithms to track, predict and learn pedestrian behaviors. A large part of our research borrows ideas related to understanding and observing human-like behaviors and their interactions from other fields including psychology, physics, and machine learning. As a result, both short-term, local interaction and long-term, high-level behavior models are improved. Our approaches use online methods to learn the trajectory-level behaviors for each agent by combining non-linear motion models and Bayesian inference. Moreover, we highlight applications of our pedestrian tracking, prediction and behavior learning algorithms to many different areas, including computer animation, computer vision, and robotics.

1.2 Main Results

Figure 1.4 illustrates the major algorithmic results of the thesis, shown in separate blocks, and their applications.

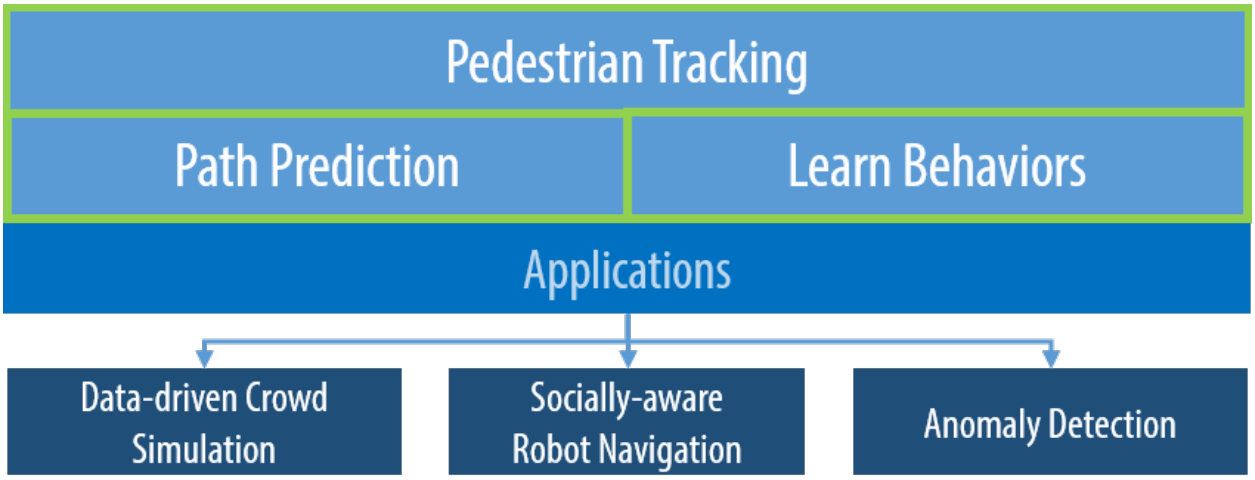


Figure 1.5: **Application:** We demonstrate the performance of our novel algorithms on three different applications. The first application is an interactive data-driven crowd simulation that includes crowd replication, crowd merging, and a combination of pedestrian behaviors from different videos. Secondly, we combine the prediction scheme with proxemic characteristics from psychology and use the result to perform socially-aware navigation. Finally, we present novel techniques for anomaly detection in low to medium density crowd videos using trajectory-level behavior learning.

1.2.1 Realtime Adaptive Pedestrian Tracking for Crowded scenes

We present new pedestrian tracking algorithms that are based on the use of particle filters to perform realtime pedestrian tracking in moderately crowded scenes. We use the notion of a crowd model as a motion prior. We demonstrate that using a crowd motion model (in this case, specifically using velocity obstacles) can improve pedestrian tracking in dense scenes, compared to using a constant velocity or constant acceleration model. Ours is also a hybrid approach which is computationally optimized and is adaptive to the crowd size, dynamics etc.

We also describe a parallel approach in which different crowd motion models are used to model each pedestrian's trajectory. Motion algorithms usually have several parameters that can be tuned to change the agents' behaviors. We assume that each parameter can have a different value for each pedestrian. The motion model of the crowd parameter estimation is formulated as an optimization problem, and the resulting algorithm solves the optimization problem in a motion model-independent manner. Overall, this formulation computes the best-fit mixture (i.e. the most optimized parameter set corresponding to every pedestrian for each model). To characterize the heterogeneous, dynamic behavior of each agent, we use an optimization-based scheme to perform the following steps:

Figure 1.6: We highlight different motion models (Boids, Social-Forces, or reciprocal velocity obstacles) used for the same pedestrian (marked in red) over different frames/time. We believe that it is not possible to model the trajectory of all pedestrians based on a single, homogeneous motion. Instead, we adaptively choose the best-fit model for every pedestrian in the scene, which can vary based on the environment or the crowd conditions.

- Choose, every few frames, the new motion model that best describes the local behavior of each pedestrian based on tracked data.
- Compute the optimal set of parameters for that motion model that best fit this tracked data.
- Compute the adaptive number of particles for each pedestrian based on a combination of metrics for optimizing performance.

The resulting mixture model is used to predict the state of the pedestrian for the next frame. In other words, the next state is used as motion prior input for the tracker; it is also combined with a confidence estimation computation to dynamically compute the number of particles. As a final step, the tracker's definitively estimated next state is fed back into the loop.

Our approach can track the positions of tens of pedestrians in around 40-50 milliseconds over long-intervals. Furthermore, we demonstrate its benefits over prior real-time prediction algorithms.

1.2.2 Realtime Pedestrian Path Prediction

We present a realtime algorithm that learns movement flows from real-world pedestrian 2D trajectories extracted from a video Fig. 1.7 gives a broad overview of our approach, including the computation of movement flows (a cluster of local pedestrian movement descriptors) and their use for pedestrian prediction. The input of the method consists of a live or streaming crowd video. We extract the initial set of trajectories using our online particle-filter based pedestrian tracker (described above). These trajectories are time-series observations of the positions of each pedestrian in the crowd. The output is the predicted state of each agent,

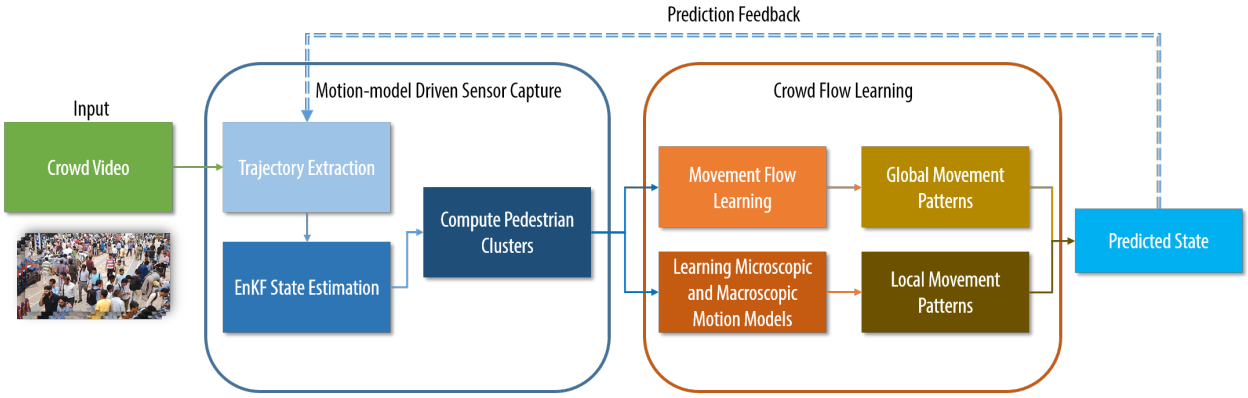


Figure 1.7: **Prediction Overview:** We highlight various components of our real pedestrian path prediction algorithm. Our approach computes both local and global movement patterns using Bayesian inference from 2D trajectory data and combines them to improve prediction accuracy

which is based on learning the local and global pedestrian motion patterns, combining different local and global pedestrians patterns learned from the trajectory data. Our approach is interactive and operates based on current and recent states; in other words, it does not require future knowledge of an entire data sequence and does not have to re-perform offline training steps whenever new, real-world pedestrian trajectory data is acquired or generated. As a result, our approach can not only effectively capture local behaviors but also individual motion variations. We describe the algorithm in detail in Chapter 3. Overall, our approach offers the following benefits over prior work:

- Our algorithm is general and can compute global and local movement patterns in real-time with no prior learning.
- We can robustly handle sparse and noisy trajectory data generated using current online pedestrian trackers.
- We observe up to 24% increase in prediction accuracy as compared to prior real-time methods that are based on simple filters or only local movement patterns.

Our approach for pedestrian prediction is general, makes no assumption about pedestrian movement or density, and performs no pre-computation and can be combined with other real-time pedestrian trackers.

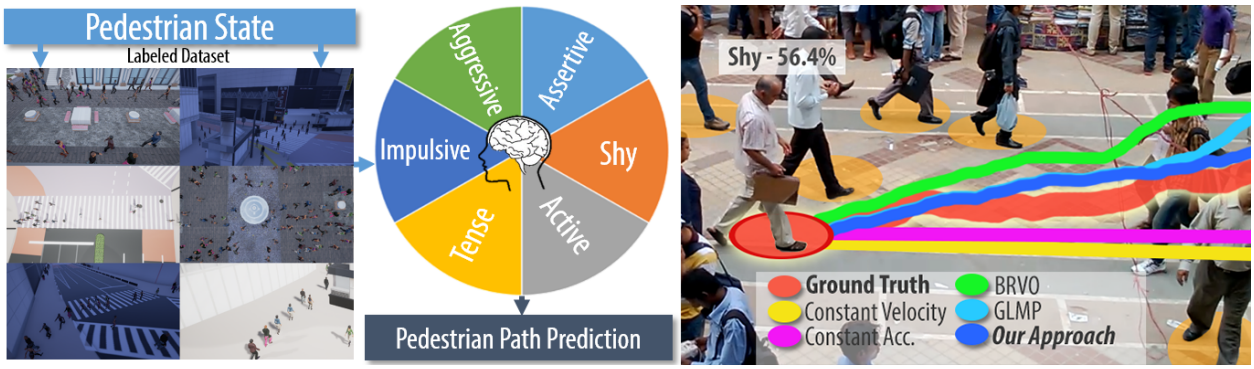


Figure 1.8: **Behavior Learning:** Our approach can automatically classify the behavior of each pedestrian in real-time. This behavior information is used to dynamically compute the motion parameters and improve the performance of our long term prediction algorithm (shown in dark blue). Our results are very close to ground truth (shown in red) and offer up to 24% improvement over prior real-time prediction algorithms, whose predicted trajectories are shown in different colors.

1.2.3 Behavior Learning

We present a novel learning algorithm to classify pedestrian behaviors (motion patterns, personality traits) based on their movement patterns. Our approach is general and makes no assumptions about the size or density of the crowd or the pedestrian's movement. We extract the trajectory of each pedestrian in a video and use a combination of Bayesian learning and pedestrian dynamics techniques to compute the pedestrian characteristics at interactive rates. The characteristics include the time-varying motion model that is used to compute the personality traits. We also present new statistical algorithms to learn high-level characteristics and global movement patterns. We combine these characteristics with Eysenck's 3-factor PEN model (Eysenck and Eysenck, 1985) and characterize the personality into six weighted behavior classes: *aggressive, assertive, shy, active, tense,* and *impulsive*. We also use the individual personalities to predict the state of the crowd under different environmental scenarios. Our approach offers many benefits:

- **Robust:** Our approach is robust, can account for noise in the pedestrian trajectories, and classifies the behavior using time-varying pedestrian movement dynamics.
- **General:** Our approach is applicable to indoor and outdoor crowd videos and makes no assumption about their size or density.
- **Crowd Analysis and Prediction:** Our approach can be used to analyze and estimate the future movement or behavior of the crowd. Furthermore, it can be used to predict different scenarios based

on the behaviors and global characteristics, e.g., the distribution and density of a large crowd at the National Mall in Figure 4.1.

To the best of our knowledge, this is the first approach that can automatically identify the behavior of each pedestrian in a crowd. We have evaluated its accuracy with a user study (88.48%) and evaluated its performance on different videos with tens of pedestrians. One example is the large crowd gathered in Washington, DC for the Presidential Inauguration (January 2017) using *PBS HD* video footage (see Chapter 6.4).

1.2.4 Applications

We demonstrate three applications from graphics, robotics and crowd surveillance based on our novel algorithms for tracking, prediction and behavior learning. -

- Improved data-driven crowd simulation, including crowd replication, augmented crowds and merging the behavior of pedestrians from multiple videos (Chapter 6.2).
- A socially-aware navigation of a robot among pedestrians. Our approach computes time-varying behaviors of each pedestrian using Bayesian learning and Personality Trait theory to improve long-term path prediction and generate proxemic characteristics for each pedestrian. We combine these psychological constraints with social constraints to perform human-aware robot navigation (Chapter 6.3).
- Finally, we improve anomaly detection in low- to medium- density crowd videos using trajectory-level behavior learning. These learned behaviors are used to segment the trajectories and motions of different pedestrians and to detect anomalies (Chapter 6.4).

CHAPTER 2
Real-time Pedestrian Tracking

2.1 Introduction

Technologies dedicated to pedestrian crowd traffic management are emerging. The tracking of human crowd motion is a key problem in this field. Despite many recent advances, it is still difficult to accurately track pedestrians in real-world scenarios, especially as the crowd density increases. The problem is hard problem due to the following reasons: intra-pedestrian occlusion (one pedestrian blocking another), changes in lighting and pedestrian appearance, and the difficulty of modeling human behavior or the intent of each pedestrian. In this context, our objective is to improve the accuracy of tracking algorithms.

In this chapter, we restrict ourselves to online and realtime trackers (Li et al., 2008b; Breitenstein et al., 2011; Li et al., 2008a; Khan et al., 2004; Comaniciu et al., 2000; SanMiguel et al., 2012), which tend to 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, and 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 increases.

The simpler motion models assume that agents will ignore any interactions with other pedestrians, instead assuming that they will follow "constant-speed" or "constant-acceleration" paths to their immediate destinations. However, the accuracy of this assumption decreases as crowd density in the environment increases (e.g. to 2-4 pedestrians per square meter). More sophisticated pedestrian motion models take into account interactions between pedestrians, formulated either in terms of attraction or repulsion forces or collision-avoidance constraints.

In real-world scenarios, the trajectory of each pedestrian is governed by its intermediate goal location, intrinsic behaviors, as well as local interactions with other pedestrians and obstacles in the scene. In a dense crowd setting, the behavior of each pedestrian changes in response to the environment, the overall crowd

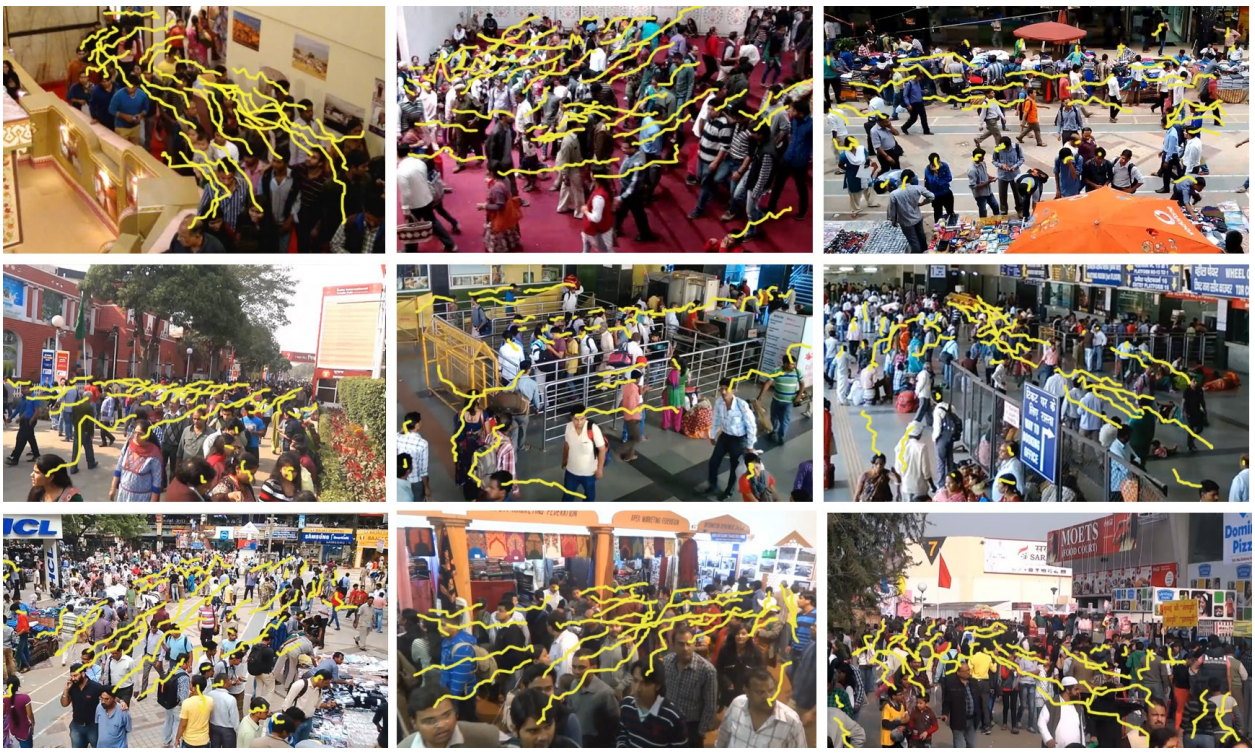


Figure 2.1: **Improved Realtime Tracking** The results of our approach on some challenging datasets. We highlight the performance of our algorithm for realtime tracking of pedestrians in indoor and outdoor scenes (shown above) with many tens of pedestrians. In such challenging scenarios, our algorithm can track up to 79% of pedestrians in a frame at 26 fps (on average). We observe up to 20% improvement in the accuracy over prior interactive methods.

density and flow, and the behavior of other pedestrians. It may not be possible, therefore, to model the overall behavior of each pedestrian with a single, homogeneous motion model. Furthermore, each of these homogeneous models is described using some parameters that may correspond to the size, speed, anticipation period, or local navigation constraints of each pedestrian. The accuracy of each motion model is governed by the choice of these parameters. As the behavior of each pedestrian responds to changes in a dynamic environment, these model parameters should be recomputed or updated to improve the resulting motion model's accuracy. Overall, we need efficient techniques that can take into account heterogeneous behaviors based on constantly changing models and underlying parameters.

Main Results:

We present a hybrid formulation that combines that combines discrete (microscopic) and continuum (macroscopic) pedestrian motion models. The discrete model is used to predict the local interactions and

collision avoidance behaviors of each pedestrian whereas 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.
- For each large cluster, we model its trajectory using a continuum flow model.
- For small clusters and individual pedestrians, we model their motion using an adaptive microscopic mixture motion model algorithm.
- We combine the discrete and continuum models with particle filters to track the pedestrians at interactive rates.

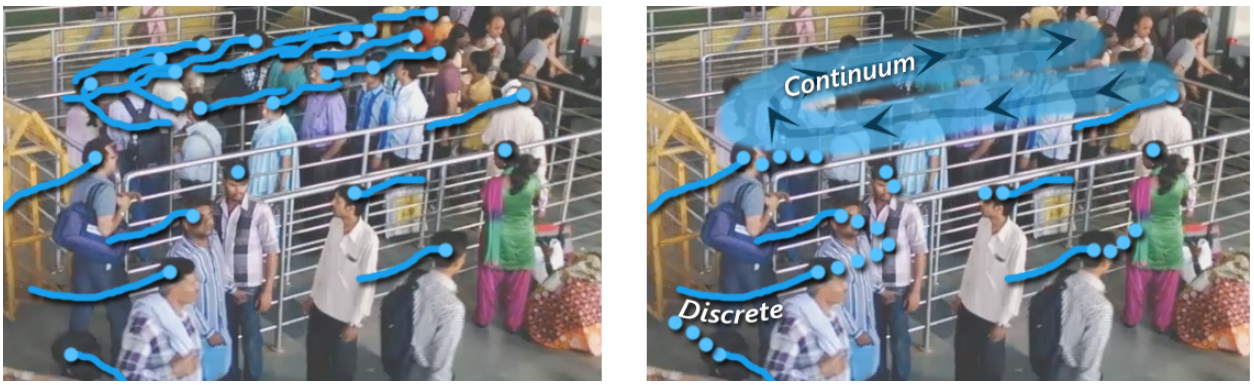


Figure 2.2: The left image highlights the tracked trajectories based on discrete motion models. The image on the right demonstrates the use of a hybrid motion model, using the continuum method for a cluster of pedestrians as well as discrete motion models for individuals. These clusters are computed in realtime based on frame coherence and pedestrian flow. The hybrid motion model can improve the tracking accuracy in these dense scenarios by 20% over prior methods.

The motion model parameter (for microscopic clusters) estimation is formulated as an optimization problem, and we use an approach that solves this combinatorial optimization problem in a model-independent manner and that is hence scalable to include any multi-agent pedestrian motion model. Our formulation computes the best-fit microscopic mixture motion model for each pedestrian based on prior tracked data. Our approach can be viewed as a feedback pipeline. In order to characterize the heterogeneous, dynamic behavior of each agent, we use an optimization-based scheme to perform the following steps:

- Choosing, every few frames, the new motion model that best describes the local behavior of each pedestrian based on tracked data.

Figure 2.3: Our microscopic mixture motion model can accurately compute the trajectories in real time. We highlight different motion models (Boids, Social-Forces, or reciprocal velocity obstacles) used for the same pedestrian (marked in red) over different frames. We believe that it is not possible to model the trajectory of all pedestrians based on a single, uniform model. Instead, we adaptively choose the best-fit model for every pedestrian in the scene that can be adapted to the environment or the crowd conditions

- Computing the optimal set of parameters for that motion model that best fit this tracked data.
- Computing the adaptive number of particles for each pedestrian based on a combination of metrics for optimizing performance.

The resulting mixture model is used to predict the next state of the pedestrian for the next frame. In other words, the next state is used as motion prior input for the tracker; it is also combined with a confidence estimation computation to dynamically compute the number of particles. As a final step, the tracker's definitively estimated next state is fed back into the loop, becoming the most recent agent state. Our approach can track the positions of tens of pedestrians in around 40-50 milliseconds over long-intervals. Furthermore, we demonstrate its benefits over prior real-time prediction algorithms.

The rest of our chapter is organized as follows. Section II gives a brief overview of prior work in tracking and motion models. We present our algorithm in Section III. We highlight its performance on different crowd video datasets in Section IV and compare its performance with prior methods.

2.2 Related Work

Robots navigating in complex, noisy, dynamic environments have prompted the development of realtime pedestrian tracking methods. Fulgenzi et al. (Fulgenzi et al., 2007) use a probabilistic velocity-obstacle approach combined with the dynamic occupancy grid; this method assumes constant linear velocity motion of the obstacles. DuToit et al. (Du Toit and Burdick, 2010) present a robot planning framework that takes into account pedestrians' anticipated future location information to reduce the uncertainty of the predicted belief

states. Other techniques use potential-based approaches for robot path planning in dynamic environments (Pradhan et al., 2011). Some methods learn the trajectories from collected data. Ziebart et al. (Ziebart et al., 2009a) use pedestrian trajectories collected in the environment for prediction using Hidden Markov Models. Bennewitz et al. (Bennewitz et al., 2005) apply Expectation Maximization clustering to learn typical motion patterns from pedestrian trajectories, before using Hidden Markov Models to predict future pedestrian motion. Henry et al. (Henry et al., 2010) use reinforced learning from example traces, estimating pedestrian density and flow with a Gaussian process. Kretzschmar et al. (Kretzschmar et al., 2014) consider pedestrian trajectories as a mixture probability distribution of a discrete as well as a continuous distribution and then use Hamiltonian Markov chain Monte Carlo sampling for prediction. Guzzi et al. (Guzzi et al., 2013) present a distributed method for multi-robot human like local navigation. Some of these methods are either not suitable for real-time applications or may not work well for dense crowds or can't efficiently capture the varying pedestrian dynamics.

Many non-particle-based motion modeling techniques have also been proposed; these techniques are useful mainly for crowded scenes in which pedestrians display similar motion patterns or movements. Song et al. (Song et al., 2013) proposed an approach that clusters pedestrian trajectories based on the assumption that "persons only appear/disappear at entry/exit." Ali et al. (Ali and Shah, 2008) presented a floor-field based method to determine the probability of motion in densely crowded scenes. Rodriguez et al. (Rodriguez and Sivic, 2011) used a large collection of public crowd videos and learned about crowd motion patterns by extracting global video features. Kratz et al. (Kratz and Nishino, 2012) and Zhao et al. (Zhao et al., 2012) used local motion patterns in dense videos for pedestrian tracking.

The key difference between our approach and previous ones derives from the following assessment of the state of the field. The accuracy of this category of trackers is improved by using realistic crowd motion models for computing motion prior. However, a single, homogeneous motion model is generally used. Every motion model relies upon one or more assumptions and has a limited validity range. It may not be possible, therefore, to model the overall behavior of each pedestrian with a single, homogeneous motion model, especially as the pedestrian or crowd conditions change. Furthermore, each of these homogeneous models is described using some parameters that may correspond to the size, speed, anticipation period, or local navigation constraints of each pedestrian. The accuracy of each motion model is governed by the choice of these parameters. As the behavior of each pedestrian responds to changes in a dynamic environment (due to other pedestrians or obstacles such as vehicles), these model parameters should be recomputed or updated to improve the resulting

motion model's accuracy. Overall, we need efficient techniques that can take into account heterogeneous behaviors based on constantly changing models and underlying parameters.

2.3 Microscopic Mixture Motion Model

In this section, we introduce the notion of a parameterized motion model. We then describe the different parameterized motion models that form the basis of the mixture motion model. Finally, we describe the microscopic mixture motion model itself. For the rest of the chapter, we refer microscopic mixture motion model by just mixture motion model or MMM.

2.3.1 Overview and Notations

We introduce the terminology and symbols used in the rest of the thesis. We use lowercase letters for scalars and bold letters for vectors. We refer to an agent in the crowd as the *pedestrian* whose *state* includes his/her trajectory and behavior characteristics. This state, denoted by the symbol $\mathbf{x} \in \mathbb{R}^5$, governs the pedestrian's position on the 2D plane:

$$\mathbf{x} = [\mathbf{p}\ \mathbf{v}^c\ \mathbf{v}^{pref}]^{\mathbf{T}}; \tag{2.1}$$

where $\mathbf{p}$ is the pedestrian's position, $\mathbf{v}^c$ is his/her current velocity, and $\mathbf{v}^{pref}$ is the *preferred velocity* on a 2D plane. A pedestrian's current velocity $\mathbf{v}^c$ tends to be different than the optimal velocity (the preferred velocity $\mathbf{v}^{pref}$) that he/she would take in the absence of other pedestrians or obstacles in the scene to achieve his/her intermediate goal. The union of the states of all the other pedestrians and the current positions of the obstacles in the scene is the current state of the environment denoted by the symbol $\mathbf{S}$. The state of the crowd, which consists of individual pedestrians, is a union of the set of each pedestrian's state $\mathbf{X} = \bigcup_i \mathbf{x_i}$, where subscript i denotes the i^{th} pedestrian. We do not explicitly model or capture pairwise interactions between pedestrians. However, the difference between $\mathbf{v}^{pref}$ and $\mathbf{v}^c$ provides partial information about the local interactions between a pedestrian and the rest of the environment.

Data Representation Our algorithm keeps track of the *state* (i.e. 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 removing the agents' state from the oldest frame and adding the latest tracker-estimated state.

The mixture motion model is a combination of several independent motion models. This mixture motion model is used to compute the best motion model for the agents during each frame. First, based on an optimization algorithm, we "configure" the motion models to "best" match the recent k-states data and select the best model based on a specific metric. Second, we use the "best configured" motion model to make a prediction on the agents' next state.

The tracker is a particle-filter based tracker that uses the motion prior, obtained from the microscopic Mixture of Motion Models, to estimate the agents' next state. This tracker further uses a confidence estimation stage to dynamically compute the number of particles that balance the trade-offs between the computation cost and accuracy.

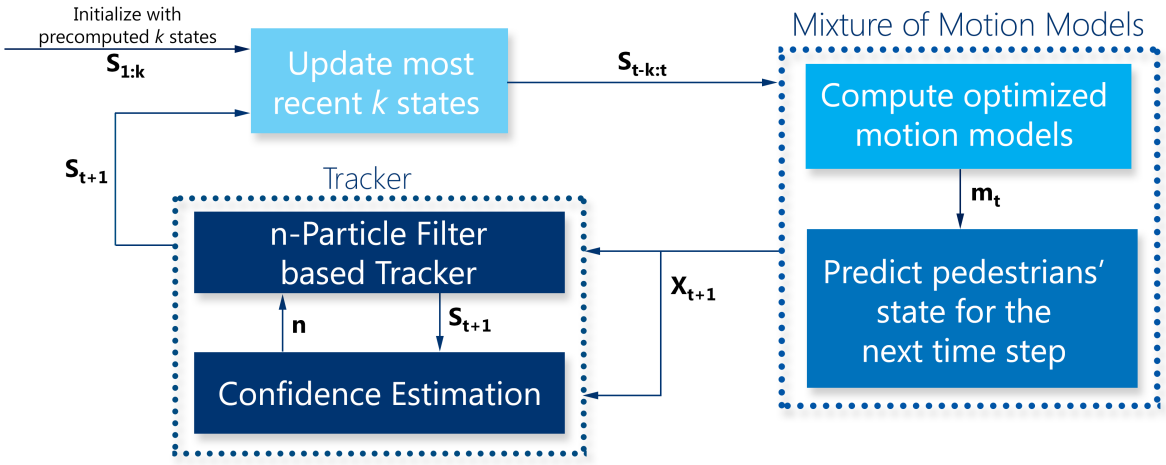


Figure 2.4: Overview of our real time tracking algorithm. The symbols used in this figure are explained in Section 2.3.1. We use the trajectory computed over prior k frames, expressed as a succession of states, to compute the new motion model; we use our microscopic mixture motion model to compute the next states using a particle filter.

We use the following additional notations specific this chapter:

- m represents the "best configured" motion model from the microscopic Mixture of Motion Models $\{f1, f2, ...\}$
- 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.

The "best configured" motion model can then 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.

2.3.2 Particle Filter for Tracking

Though any online tracker that requires a motion prior system can be used, we use particle filters as the underlying tracker algorithm. The particle filter is a parametric method that solves non-Gaussian and non-linear state estimation problems (Arulampalam et al., 2002). Particle filters are frequently used in object tracking, since they can recover from lost tracks and occlusions. The particle tracker's tracking uncertainty is represented in a Markovian manner by only considering information from present and past frames.

Here, we consider the "best configured" motion model m_t as well as the error Q_t in the prediction that this "best configured" motion model generated. Additionally, the observations of our tracker can be represented by a function $h()$ that projects the state x_t to a previously computed state S_t. Moreover, 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:

$$S_{t+1} = m_t(x_t) + Q_t, \tag{2.2}$$

$$S_t = h(x_t) + R_t. \tag{2.3}$$

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

$$\begin{aligned} &p(x_t|S_{t-k:t}) = \\ &\quad \alpha p(S_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}) p(x_{t-1}|S_{t-k:t-1}) dx_{t-1}, \end{aligned} \tag{2.4}$$

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:

$$p(x_t|S_{t-k:t}) \approx \alpha p(S_t|x_t) \sum_{i=1}^{n} \pi_{t-1}^{(i)} p(x_t^{(i)})|p(x_{t-1}^{(i)}), \tag{2.5}$$

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.

We optimize our computation speed by adaptively modifying the number of active particles in our system using a combination of confidence metrics. A brief overview is given in Section 2.3.6.

2.3.3 Parameterized Motion Model

A motion model is defined as an algorithm f, which, from a collection of agent states $\mathbf{x}_t$, derives new states $\mathbf{x}_{t+1}$ for these agents, representing their motion over a timestep towards the agents' immediate goals $\mathbf{G}$:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{G}). \tag{2.6}$$

Motion algorithms usually have several parameters that can be tuned in order to change the agents' behaviors. We assume that each parameter can have a different value for each pedestrian. By changing the value of these parameters, we get some variation in the resulting trajectory prediction algorithm. We use $\mathbf{P}$ to denote all the parameters of all the pedestrians. Typically, for a crowd of 50 pedestrians, the dimension of $\mathbf{P}$ could be anywhere in the range of 150-300 depending on the motion model. In our formulation, we denote the resulting parameterized motion model as:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{G}, \mathbf{P}). \tag{2.7}$$

2.3.4 Motion Models

Our mixture motion model can include any generic motion model that conforms to Equation (2.7). Here we describe the three component motion models that currently make up the mixture motion model in our current implementation.

2.3.4.1 Reciprocal Velocity Obstacles

RVO 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(Van Den Berg et al., 2011b). Each agent is represented as a 2D circle in the plane, and the parameters (used for optimization) for each agent consist of the representative circle's radius, maximum speed, neighbor distance, and time horizon (only future collisions within this time horizon are considered for local interactions).

Let V_{pref} be the preferred velocity for a pedestrian that is based on the immediate goal location. 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 (Van Den Berg et al., 2011b). The computation of the 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:

$$V_{RVO} = \underset{V \notin ORCA}{\arg\max} \|V - V_{pref}\|. \tag{2.8}$$

More details and mathematical formulations of the ORCA constraints are given in (Van Den Berg et al., 2011b). As per Equation (2.7), f returns the states obtained with the admissible velocity that is closest to the preferred velocity.

2.3.4.2 The Boids Model

Initially developed to simulate the flocking behavior of birds, this model has later been extended to pedestrian motion in a crowd. Broadly, three rules are enforced on Boids agents:

- **Separation**: steer to avoid crowding local agents
- **Alignment**: steer towards the average heading of local agents
- **Cohesion**: steer to move toward the average position (center of mass) of local agents

Thus, as per Equation (2.7), f is a function of agents' positions at some specified future time (current time plus constant). When the predicted distance between the pedestrians gets too low, a separation force is

computed and added to the attraction force that is pulling the agents toward their goal. The parameters are radius (size of 2D circle agents) and comfort speed (i.e., speed when no interactions occur).

2.3.4.3 Social Forces Model

The social forces model is defined by the combination of three different forces: the personal motivation force, social forces, and physical constraints:

- **Personal Motivation force** (F^M): This is the incentive to move at a certain preferred velocity in a certain direction.
- **Social forces** (F^S): These are the repulsive forces from other agents and obstacles.
- **Physical Constraints** (F^P): These are the hard constraints other than the environment and other agents.

The net force $F^C = F^M + F^S + F^P$ then defines an agent's chosen new velocity. For a detailed explanation of the method, refer to (Helbing and Molnar, 1995b).

As per Equation (2.7), f is a function of the agents' positions from which all computed forces are derived. The parameters are radius and comfort speed.

2.3.5 Microscopic Mixture of Motion Models

We now present the algorithm to compute the mixture motion model, which essentially corresponds to computing the "best" motion model at any given timestep. 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 "best" motion model is determined by an optimization framework, which automatically finds the parameters $\mathbf{P}$ that minimize the error metric. Wolinski et al. (Wolinski et al., 2014) designed an optimization framework for evaluating crowd motion models but it computes the optimal parameters in an offline manner for a single homogenous simulation model. Our framework is online and iteratively computes the best heterogeneous motion every few frames and chooses the most optimized crowd parameters at a given time. The computation cost is considerably lower and hence useable for real-time tracking.

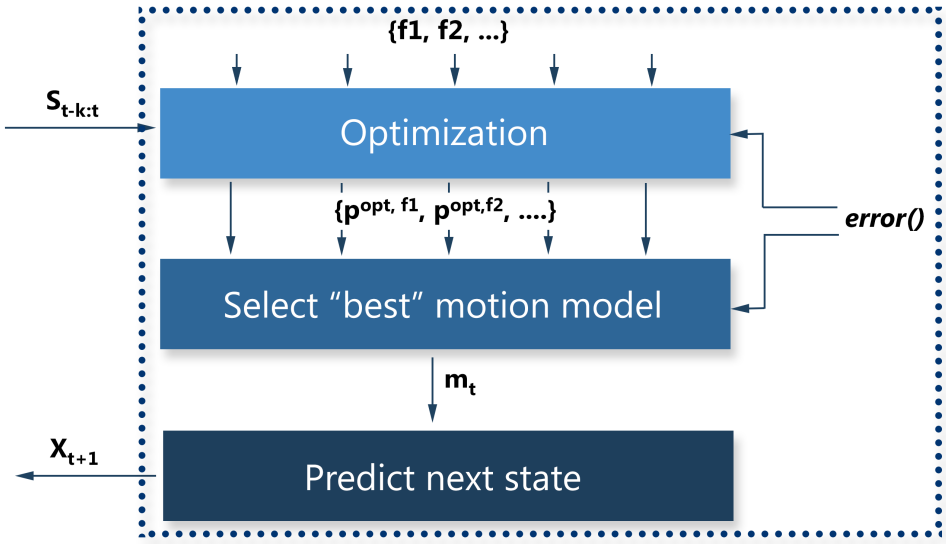


Figure 2.5: Our parameter optimization algorithm used in Figure 2.4. Based on the error metric, we compute optimal parameters for each motion model. The best motion model (from RVO2, Social Forces, Boids or LIN) is used for trajectory extraction and to predict the next state.

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

$$\mathbf{S}_{t-k:t} = \bigcup_{i=t-k}^{t} \mathbf{S}_i. \tag{2.9}$$

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

$$f(\mathbf{S}_{t-k:t}, \mathbf{P}) = \bigcup_{i=t-k}^{t} f(\mathbf{x}_i, \mathbf{G}, \mathbf{P}), \tag{2.10}$$

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:

$$\mathbf{P}_t^{opt,f} = \underset{\mathbf{P}}{\operatorname{argmin}}\, error(f(\mathbf{S}_{t-k:t}, \mathbf{P}), \mathbf{S}_{t-k:t}), \tag{2.11}$$

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

For several motion algorithms $\{f1, f2, ...\}$, we can then compute the algorithm which best matches the agents' k-states $\mathbf{S}_{t-k:t}$ at timestep t:

$$m_t = f_t^{opt} = \underset{f}{\operatorname{argmin}}\, error(f(\mathbf{S}_{t-k:t}, \mathbf{P}_t^{opt,f}), \mathbf{S}_{t-k:t}), \tag{2.12}$$

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:

$$\mathbf{x}_{t+1} = m_t(\mathbf{S}_t). \tag{2.13}$$

Optimization Algorithm and Error Metric The optimization of crowd parameters is a unique and challenging problem. Because most simulation methods have several parameters to tune for each agent, even moderately-sized scenarios with a few dozen agents can become a hundred-dimensional optimization problem.

In total we tested three global optimization approaches: *Greedy Algorithm*, *Simulated Annealing*, and *Genetic Algorithm*.

For the greedy approach we start by choosing random parameters for every agent. The chosen data similarity metric is then evaluated to establish a baseline measure of how well the simulation matches the data. After several iterations, where in each iteration starts with the best set of simulation parameter seen so far. This new set of parameters is evaluated, whichever set of parameters has the lowest error metric over all the iterations is chosen as the optimal parameters for the agents.

The main limitation with a greedy approach is that it will get stuck in local minimum in search space and also the final outcome depends on the starting point. Simulated Annealing addresses this problem. Analogous with thermodynamics, simulated annealing incorporates a '*temperature*' parameter into the minimization procedure. At high temperatures, we explore the parameter space whereas at lower temperature, we restrict the exploration.

Algorithm 1 gives the pseudocode for the process where:

neighborState(): pick a new random value for a random parameter according to the parameter's base distribution

move(): is $True$ iff $e_{new} < e_{old}$, $exp(\frac{e_{old}-e_{new}}{T})$.

Algorithm 1: Simulated annealing.

```
1  k ← 0                                                  // initialize loop counter
2  while k < K do
3      T ← temperature(k, K)                              // compute temperature
4      s_new ← neighborState(s)                           // try new neighbor
5      e_new ← cost(s)                                    // compute cost
6      if move(e, e_new, T) then                          // is new state better?
7          s ← s_new; e ← e_new                           // yes, change state
8      if e < e_best then                                 // did we find a new minimum?
9          s_best ← s; e_best ← e                         // save new optimum
10         k ← 0                                          // reset loop counter
11     k ← k + 1                                          // increase loop counter
```

temperature(): is $\frac{K-k}{K}$, k being the number of iterations with no improvement and K the number of such iterations allowed.

cost(): the cost as returned by the currently used metric.

We also use a Genetic algorithm (Holland, 1992). The underlying optimization technique as algorithm offers the best compromise between optimization results and speed. The efficiency component is important as 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.

Algorithm 2: Genetic algorithm.

```
1  pop ← initialize()                                     // initialize population
2  while true do
3      selection(pop)                                     // evaluate and select fittest
4      if termination() then                              // should we terminate?
5          stop                                           // yes, stop loop
6      pop ← reproduction(pop)                            // new generation
```

Algorithm 2 provides pseudocode for the method given the following functions:

- **initialize()**: parameters randomly initialized in accordance with the base distribution for each parameter.
- **selection()**: individuals are sorted according to their score and divided into 3 groups: Best, Middle and Worst.

- **termination()**: the algorithm is terminated after finding K successive loop iterations without any new optimum.
- **reproduction()**: based on which group it belongs to, a parameter set is attributed three probabilities α, β and γ. For each parameter of this individual, α decides if the value is changed or not, β decides if the value is changed by crossover or mutation and, finally, γ decides which type of mutation is done.
- crossover: a crossover is done by copying a value from an individual belonging to the Best group.
- mutation: a mutation is done by picking a new value at random based on either the base distribution or the current real distribution of an individual from the Best group (according to γ).

At each iteration, this algorithm evaluates and ranks all possible parameter sets (solutions) currently in the solution pool. If there have been a certain number of successive iterations without any improvement, the process is terminated. Otherwise, individual parameter values in each solution have a probability of being modified. If so, this modification has a probability of being either 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. This probability distribution can either be the one defined by the user (for instance, a preferred velocity could obey a normal law with mean $1.4m.s^{-1}$ and standard deviation $0.3m.s^{-1}$) or one that is computed on parameter values from better ranked solutions.

We have tested these algorithms both in terms of how well they minimize the error metric and in terms of how fast they converge. Figure 2.6 shows the scores after optimization with all three methods, for the Boids, Social-forces and RVO2 motion models (separately), as well as the Mixture of Motion Models. As can be observed, the genetic algorithm leads to the lowest scores, followed by the simulated annealing and greedy algorithms. Note that as expected, the Mixture of Motion Models gives the lowest error scores as it consistently selects the best motion model for each situation.

The greedy algorithm is here the fastest, followed by the genetic and simulated annealing algorithms. Note that the Mixture of Motion Models is only marginally slower than the other methods as the optimizations for the models constituent of the Mixture are performed in parallel. From these comparisons, we chose the genetic algorithm to generate all further results.

An error metric is also needed to compute the term in Equation (3.9). In our case, we have chosen a metric that simply computes the average 2-norm between the observed agent positions and the tracker-computed

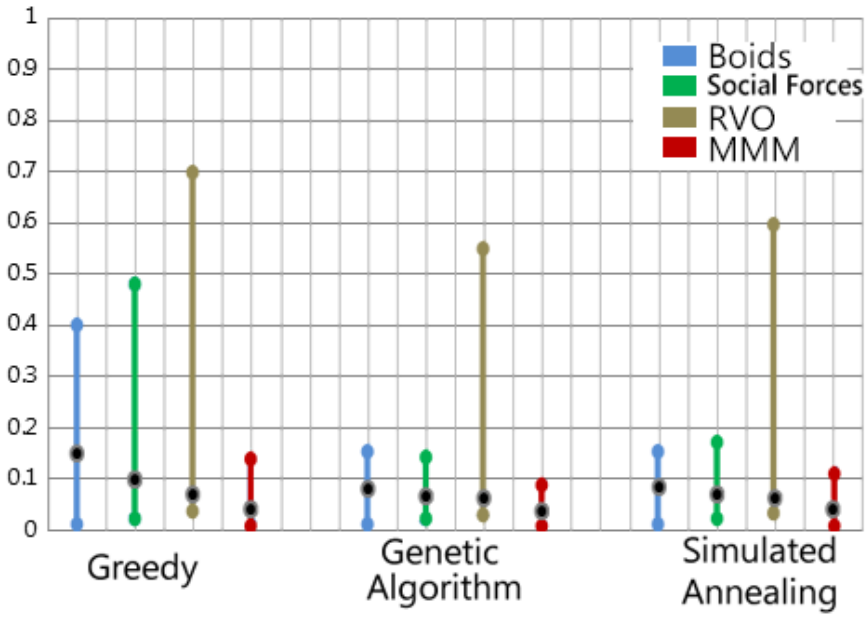


Figure 2.6: Comparing the score of the different optimization approaches. Each graph is a range of the scores (minimum and maximum) and the black dot is the mean score. We compute the score from the normalized error metric. A lower value indicates better optimization. MMM or the 'Motion-Model Mixture' is our approach.

positions. Formally, this metric is defined at timestep t as follows:

$$error(f(\mathbf{S}_{t-k:t}, \mathbf{P}), \mathbf{S}_{t-k:t}) = \sum_{i=t-k}^{t} \|\mathbf{S}_i - \mathbf{x}_i\|, \tag{2.14}$$

where the $\mathbf{x}_i$ are from $f(\mathbf{S}_{t-k:t}, \mathbf{P})$ as per Equation (2.10).

2.3.6 Adaptive Particle Selection

The performance of a particle filter is proportional to the number of particles used for each pedestrian, and the process can be expensive for a high number of particles. However, with more particles, the probability that a pedestrian will be tracked accurately is higher; fewer particles, though computationally less expensive, lowers the tracking accuracy. As a result, we need to use an appropriate number of particles to balance the trade-offs between computation cost and accuracy. Ideally, one would use fewer particles most of the time, increasing their number only when needed. This increase might be necessary when there is a large change in motion trajectory, lighting, appearance, or partial occlusions, for example.

To this end, we estimate tracker confidence and particle selection by using the motion model. We analyze the confidence of our tracker given the number of particles based on combining various metrics to measure the propagation and motion model reliability. The propagation reliability is a measure of how well the object matches the initial target candidate and also the last tracked object:

$$pr_t = g(\|O_t - O_{t-1}\|, \|O_t - O_0\|), \tag{2.15}$$

where pr_t is the propagation reliability at time t and O_t denotes the object representation at time t. Motion model reliability is a normalized difference measure between the tracked state and the predicted state $f(x_{t-1})$ given by the motion model f:

$$mmr_{t+1} = h(\|f(x_{t-1}) - S_t\|), \tag{2.16}$$

where mmr_{t+1} is the motion model reliability at timestep $t+1$ and h is a function varying linearly to the norm difference of the actual and simulated trajectories.

The combination of these metrics helps us in optimizing the number of active particles needed in the system. In our Mixture of Motion Models, our system chooses the optimal motion algorithm m_t from all possible motion models $\{f1, f2, ...\}$ (Equation (2.12)) with the optimal parameter set. Hence the motion model reliability is always higher compared to systems with homogeneous or non-varying motion models.

$$mmr_{t+1}^{opt} = h(\|m_t(x_{t-1}) - S_t\|). \tag{2.17}$$

In this section, we give details of various stages of our algorithm, shown in Figure 3. Our hybrid motion model consists of two parts, the discrete or the microscopic model and the continuum model. Both these models return a future pedestrian state. Next, we describe the various components of our approach.

2.3.6.1 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, that consists of a single pedestrian. Next we merge these clusters, by analyzing their geometric proximity and velocities. This proximity a function of the Euclidean

distance, the speed of each agent and their motion direction. In our experiments we found that a bottom-up approach is more efficient in crowds composed of small clusters than top-down.

We improve on the group-expand procedure of (McPhail and Wohlstein, 1982) by including many additional crowd features for clustering the pedestrians. A connectivity graph is constructed (Ge et al., 2009) 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 a pedestrian i and j are temporarily 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_k. A pedestrian i can be added to an existing cluster of size k if and only if it is connected with at least half of the existing pedestrians in the cluster, i.e., the degree of $n_i \geq \lceil \frac{k}{2} \rceil$, 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

$$\hat{e}_k = \begin{cases} (\frac{l}{2}^2) & \text{when } l \text{ is even,} \\ \frac{l-1}{2}(1 + \frac{l-1}{2}) & \text{when } l \text{ is odd.} \end{cases} \tag{2.18}$$

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

$$e_{m+n} \geq (\hat{e}_{m+n} + e_m - \hat{e}_m + e_n - \hat{e}_n) \tag{2.19}$$

2.3.6.2 Macroscopic/Continuum Representation

After estimating every cluster, we calculate the flow per cluster based on Hughes et al. (Hughes, 2003) continuum crowd behavior. To derive the equations that govern the flow of a pedestrian, we need to combine the unsteady continuity equation (Hughes, 2003) with the following three hypotheses governing 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 that they face to reach 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 temper 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

$$-\frac{\delta\rho}{\delta t}+\frac{\delta\rho g(\rho)f^2(\rho)\frac{\delta\varphi}{\delta x}}{\delta x}+\frac{\delta\rho g(\rho)f^2(\rho)\frac{\delta\varphi}{\delta y}}{\delta y}=0, \tag{2.20}$$

and

$$g(\rho)f(\rho)=\frac{1}{\sqrt{(\frac{\delta\varphi}{\delta x})^2+(\frac{\delta\varphi}{\delta y})^2}} \tag{2.21}$$

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 for further explanation we direct our readers to Hughes et al. 2002 (Hughes, 2002).

2.3.7 Evaluation Metrics

In general, crowd data can be analyzed with a wide variety of similarity metrics, which capture different aspects of the data. All the metrics capture different aspects of similarity (distance, path length, inter pedestrian distance, density etc) and are equally important. Here we summarize several metrics tested in this work. In all the data, we assume the reference data $\mathbf{z}_k$ consists of a vector of positions for all the pedestrians.

The **absolute difference metric (D)** computes the total distance in position over all agents over all timesteps:

$$D_k = \|\mathbf{z}_k - \mathbf{x}_k\|. \tag{2.22}$$

The **path length metric (L)** compares the difference in total length traveled between agents in the reference data and the simulated agents:

$$L_k = (\mathbf{z}_{k+1} - \mathbf{z}_k) - (\mathbf{x}_{k+1} - \mathbf{x}_k). \tag{2.23}$$

The **inter-pedestrian distance metric (I)** compares the difference in average distance (as a 2-norm) between every pair of agents. If P is the ensemble of all agent pairs $P = \bigcup\{i, j\}$, then:

$$I_k = \sum_P |\mathbf{x}_k^i - \mathbf{x}_k^j| - \sum_P |\mathbf{z}_k^i - \mathbf{z}_k^j|. \tag{2.24}$$

Finally, the **fundamental diagram metric (F)** compares the speed of an agent to the density of agents in its location. This metric is inspired by the field of pedestrian dynamics, where it is commonly used to measure pedestrian flow rates (e.g., (Curtis and Manocha, 2012)). Our implementation of the metric defines a "gate" area on the agent's path (as in (Chattaraj et al., 2009), which allows us to compute the density of population at an agent's location ($\frac{numberAgentsInsideGate}{areaOfGate}$) when the agent is inside this gate.

$$F_k = |\mathbf{z}_k(d_k) - \|\mathbf{v}_k\||, \tag{2.25}$$

where $\mathbf{d}_k$ is the density at the location of each agent while inside the gate, and the reference data $\mathbf{z}_k$ is a function that maps density to speed based on results known from human motion studies.

2.4 Model Analysis

Our Mixture Motion Model can include any generic motion model that conforms to Equation (2.7). Here we describe the four component motion models that currently make up the Mixture Motion Model in our current implementation. We selected 4 motion models: the Reciprocal Velocity Obstacle model (RVO2), the Boids model, Social-Forces model and the Constant Velocity model (LIN). We analyse why and where different model succeeded where other models failed. MMM will inherently choose the best-fit model due to the optimization, this section tries to analyze why one model was chosen over the other (Figure 2.8).

As detailed in Section 2.3.4, they cover complementary ranges of crowd densities. The Social-Forces model simulates local interactions between pedestrians as sets of repulsive forces. This is representative of what happens in dense situations, where people may enter in contact with one another. In lower densities, but still with highly cohesive motions, the Boids model simulates well how each pedestrian aligns his own motion with his or her nearest neighbors. Finally, in less dense scenarios, trajectories are individualized and anticipation plays a great role in interactions: RVO2 is the only one of the four techniques capable of simulating such behaviors.

Low Density Data (3-7 pedestrians): This data category regroups various cases of 3-7 pedestrians crossing ways. The following are two different categories of low density crowds.

- *A mostly uni-directional flow with a maximum of 1-2 pedestrians walking against flow* : In this case, most complex collision avoidance methods performed as good as the constant velocity model.
- *Mostly even distribution of flows (i.e. crosswalk) Random pedestrian directions without any major observable flows :* Both RVO and Social forces work good in terms of short-term collision anticipation and hence fit better to the observed data. Constant velocity works well too in some cases where the agents are far apart and the chance of collision is lesser.
- *Random pedestrian directions without any major observable flows :* Social forces and RVO perform better than the other algorithms. Constant velocity performs the worst.

Medium Density data (8-24 pedestrians): This data category is similar to the previous, except that more pedestrians are present.

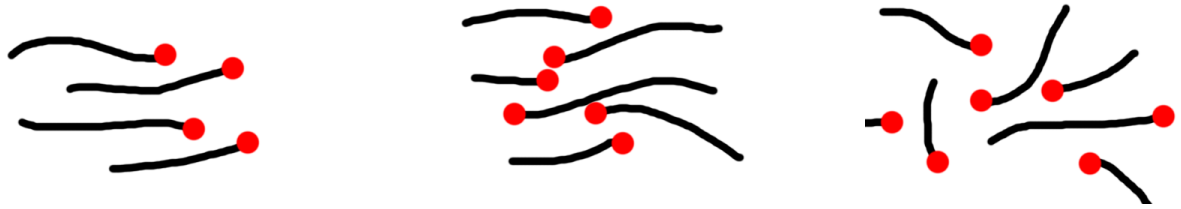


Figure 2.7: The three different categories of crowds based on increasing level of inter-pedestrian interaction: 1) Mostly unidirectional flow, 2) evenly distributed flow/crossflow, 3) high degree of randomness/no strong flows visible

- *A mostly uni-directional flow with a maximum of 1-4 pedestrians walking against flow :* Similar to what we observed in the last case, most complex collision avoidance methods performed as good as the constant velocity model.

- *Mostly even distribution of flows (i.e. crosswalk) Random pedestrian directions without any major observable flows:* RVO does a better job of anticipating long-term collision avoidance than the other algorithms. All other algorithms perform poorly.

- *Random pedestrian directions without any major observable flows :* As previously, social forces and RVO perform better than the other algorithms. RVO gives a better speed profile than the other algorithms, because its agents accelerate and decelerate when needed and are better synchronized with pedestrians. Constant velocity, as expected, performs the worst.

High Density data (more than 25 pedestrians):

- *A mostly uni-directional flow with a maximum of 1-6 pedestrians walking against flow:* Similar to what we observed earlier, most complex collision avoidance methods performed as good as the constant velocity model but as the number of pedestrian walking against the flow increased, it affected the collision avoidance and the constant velocity become progressively worse.

- *Mostly even distribution of flows (i.e. crosswalk):* RVO does a better job of anticipating long-term collision avoidance than the other algorithms. All other algorithms, inclusing social forces, perform poorly.

- *Random pedestrian directions without any major observable flows:* RVO2 agents anticipate future collisions and are spread in a pattern more similar to that of the real pedestrians. All other methods perform poorly.

General Trends:

- **Fundamental Diagram:** Boids-like algorithm gets easily stuck in scense with sharp corners, and the resulting simulations are far from the observed data. However, the Social-force and RVO2 algorithms fit the data well; RVO2 ultimately matches the fundamental diagrams better at higher densities. This is largely due to the Social-force agents displaying instabilities near walls at high densities

- **Denser Scenes:** Boids-like model lacks anticipation and fails to completely recreate the wanted behavior; RVO2 is more successful. This is likely due to RVO2 being the only one of the three methods to incorporate predictive collision avoidance but as the density becomes very high (≥ 50) the advantage of RVO2 decreased significantly. The convergence of the performance is expected, in part, because there is little room to anticipate trajectories in dense scenarios with lots of individuals in close quarters. RVO2 and the Social-force model score similarly in the Fundamental diagram metric for similar reasons.

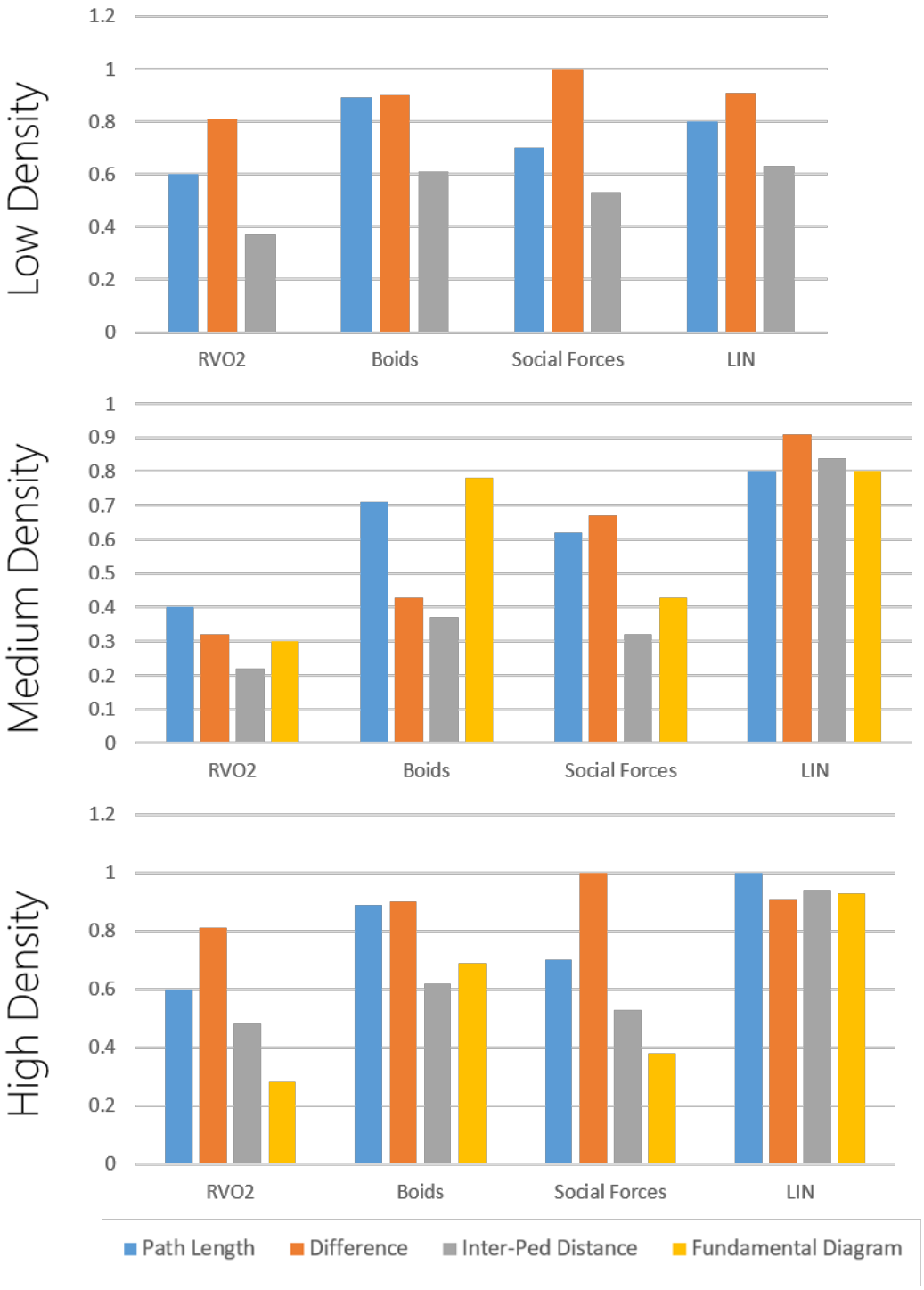


Figure 2.8: **Benchmarks:** Comparative scores (y-axis, lower is better) of the Boids-like, Social-force and RVO2 and LIN models at three different densities. Note: *The fundamental diagram metric was only analyzed for medium and high density videos.*

2.4.1 Tracking Evaluation

We use the **CLEAR MOT** (Keni and Rainer, 2008) evaluation metrics to analyze the performance analytically. 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 amount of false positives, missed detections, and identity switches. These metrics have become standard for evaluation of detection and tracking algorithms in the computer vision community, and we refer the interested reader to (Keni and Rainer, 2008) for a more detailed explanation. We analyze these metrics across the density groups and the different motion models (Table 2.3 and Figure 2.9).

2.4.2 Tracking Results

We highlight the performance of our algorithm based on a Mixture of Motion Models on different benchmarks, comparing the performance of our algorithm with single, homogeneous motion model methods: constant velocity model (LIN), LTA (Pellegrini et al., 2009a), Social-Forces (Yamaguchi et al., 2011b), Boids (Reynolds, 1999b), and RVO2 (Van Den Berg et al., 2011b). LIN models the velocities of pedestrians as constant, and is the underlying motion model frequently used in the standard particle filter. The other four models compute the pedestrian states based on optimizing functions, which model collision avoidance, destinations of pedestrians, and the desired speed. In our implementation, we replace the state transition process of a standard particle-filtering algorithm with different motion models.

We evaluate some challenging datasets (Bera and Manocha, 2014a) which are available publicly and also some standard datasets 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 performance to a baseline mean-shift tracker (Table 2.4).

For our evaluation, we have divided our system into two phases:

Initialization: Here we initialize the motion model estimation and parameter-optimization system with hand-drawn or ground truth data for a few initial frames, which is computed offline. For our experiments, we use the first 10 frames. We compute a score that is used to choose the best-fit model from our motion model set and the associated parameters.

Prediction: After learning from the initial data, we use the predicted set of parameters to model the state transition part of the standard Bayesian inference framework. We iteratively and incrementally recompute the score and update the motion model. This computation is performed in real time.

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. The average human stride length is about 0.8 meter and we consider the tracking to be incorrect if the mean error is more than this value. Our method provides 20% higher accuracy over LIN for medium density crowds (Table 2.4).

Model / Parameters	**min**	**max**	**mean**
Boids model			
radius (m)	0.1	1	0.3
comfort speed (m/s)	1	2	1.5
Social-Forces model			
radius (m)	0.1	1	0.3
comfort speed (m/s)	1	2	1.5
RVO2 model			
comfort speed (m/s)	1	2	1.5
neighbor distance (m)	2	20	11
radius (m)	0.2	0.8	0.5
agent time horizon (s)	0.1	5	2
obstacle time horizon (s)	0.1	5	2

Table 2.1: Initial motion model parameters for optimization.

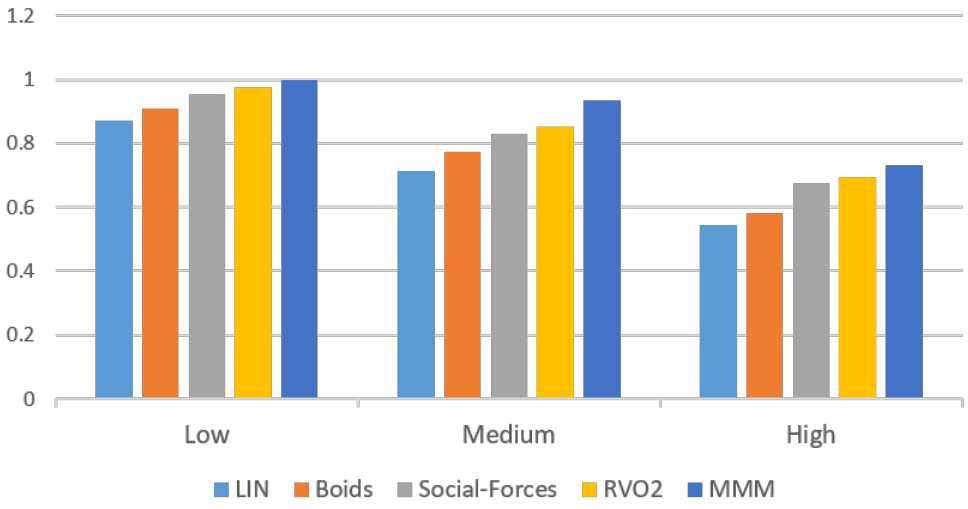


Figure 2.9: We visually analyze the data in Table 2.3 normalized MMM (Low Density) as a baseline.

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
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 2.2: Crowd Scenes used as Benchmarks. We highlight many attributes of crowd videos including density and the number of pedestrians tracked. We use the following abbreviations about the underlying scene: Background Variations(BV), Partial Occlusion(PO), Complete Occlusion(CO), and Illumination Changes(IC).

	LIN			Boids			Social-Forces			RVO2			**MMM**		
	LD	MD	HD	LD	MD	HD	LD	MD	HD	LD	MD	HD	**LD**	**MD**	**HD**
MOTP	64.42%	52.82%	40.31%	67.24%	57.10%	43.14%	70.52%	61.33%	49.88%	72.19%	63.17%	51.31%	**73.98%**	**69.23%**	**54.29%**
MOTA	49.42%	35.3%	31.37%	50.59%	26.42%	30.88%	53.28%	44.19%	33.51%	53.95%	48.81%	35.83%	**54.18%**	**50.16%**	**38.83%**

Table 2.3: We compare the MOTA and MOTP values across the density groups and the different motion models.

2.4.3 Implementation Details

We tested these algorithms on an IntelHaswell, Corei7-4771 Processor (4 Cores) with an 8MB Cache, 3.90 GHz and IntelHD Graphics 4600. Our work is implemented in C++, and some components use OpenMP and OpenCL toexploit multiple cores.We adopted a agent-level parallelism: individual pedestrian computations are distributed across the CPU cores (except for the motion-model computations, where pedestrian behavior is interlinked and tasks are highly sequential)

	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
Social Forces	56	16	66	26	52	33	62	15	74	11	68	23	41	19	59	31	75	18	72	14
RVO2	57	14	69	20	53	29	64	13	71	10	64	26	42	18	53	32	72	20	74	16
MeanShift	27	32	31	38	23	52	34	29	39	36	41	31	22	33	39	45	31	28	45	28
MMM	63	12	73	19	57	27	67	10	77	7	71	20	44	16	63	28	79	17	78	14

Table 2.4: We compare the percentage of successful tracks (ST) and ID switches (IS) of our Mixture Motion Model algorithm (MMM) with homogeneous motion models - LIN, Boids, Social Force, LTA, RVO2, and a baseline mean-shift tracker. Our method provides higher accuracy compared to homogeneous motion models and lesser ID switches. The benefits of our approach are higher, as the crowd density increases. These datasets are publicly available at http://gamma.cs.unc.edu/RCrowdT/.

2.5 Limitations, Conclusions, and Future Work

We present a real time algorithm for pedestrian tracking in crowded scenes that are needed for next generation shared-environment human-robot collaboration systems as well as design of architectural models and urban environments. Our algorithm provides a good balance between accuracy and runtime performance. We highlight its performance on many pedestrian datasets, showing that it can track crowded scenes in real time on a PC with a multi-core CPU. Furthermore, we highlight the improved accuracy and the performance in complex benchmarks with low to medium density crowds.

Our approach has some limitations related to our motion model set. Our motion model set does not take into account physiological and psychological pedestrian traits. All pedestrians are modeled with the same sensitivity towards gender and density; our model set does not take into account heterogeneous agent characteristics, which affect the final behavior. These behavior characteristics can introduce additional errors in our confidence estimation. In practice, the performance of the algorithm can vary based on various other attributes of the input video.

As part of our future work, we would like to incorporate the personality characteristics of the pedestrians, along with other characteristics, such as "fundamental diagrams" from pedestrian dynamics. We would like to parallelize the approach on a GPU to handle more complex pedestrian datasets in realtime. Finally, we would like to evaluate their performance with robots, e.g. service robots or autonomous vehicles, navigating through pedestrians.

CHAPTER 3

Realtime Pedestrian Path Prediction using Global and Local Movement Patterns

3.1 Introduction

In Chapter 2, we discussed how we model different crowds and improve the motion model for tracking. In this chapter we will present algorithms to improve pedestrian prediction which builds those tracking algorithms. In everyday life, the use of robots is increasing. More and more robots are being introduced into human surroundings. Thus, it becomes increasingly important to develop safe and reliable techniques for human-robot interaction. It is necessary for robots working around humans to be able to successfully navigate to their goal positions in dynamic environments with multiple people moving around them. A robot in a dynamic environment thus needs the ability to sense, track, and predict the position of all people moving in its workspace to navigate without collision in complicated environments.

As mobile robots are increasingly used for service tasks, it is important for these robots to perceive the intent and trajectory of humans for collaborative collision avoidance. In the context of autonomous driving, it is important to compute precise estimates of the current and future positions of each pedestrian with respect to the moving vehicle for collision-free navigation. In computer vision and multimedia applications, pedestrian movement detection and prediction is used for detecting abnormal activities or behaviors.

Pedestrian path prediction from videos or other sensor data is regarded as a challenging problem. In general, pedestrians have varying behaviors and can change their speed to avoid collisions with the obstacles in the scene and other pedestrians. In high density or crowded scenarios, the pairwise interactions between the pedestrians tend to increase significantly. As a result, the highly dynamic nature of pedestrian movement makes it hard to estimate their current or future positions. Furthermore, many applications need real-time prediction capabilities to estimate the positions of large number of pedestrians in a short time.

Some of the commonly used algorithms for predicting the path of pedestrians are based on tracking filters. These include Kalman filter, particle filter, and their variants. Other approaches are based on hidden Markov models. Many of these trackers also use motion models for pedestrian movement to improve the

Figure 3.1: **Improved Prediction** We demonstrate the improved accuracy of our pedestrian path prediction algorithm (GLMP) over prior real-time prediction algorithms (BRVO, Const Vel, Const Accel) and compare them with the ground truth. We observe upto 18% improvement in accuracy.

prediction accuracy. The simplest motion models are based on constant velocity or constant acceleration assumptions (Del Bimbo and Dini, 2011). Other algorithms are based on sophisticated motion models based on social forces (Helbing and Molnar, 1995a), reciprocal velocity obstacles (Van Den Berg et al., 2011a), dynamic social behaviors (Pellegrini et al., 2009a), etc. to model pairwise interactions between the pedestrians, or combine Bayesian statistical inference with velocity-space reasoning (Kim et al., 2014) for computing individualized motion model for each pedestrian. In practice, all these methods only capture local interactions and movements, which are mostly useful for short-term deviations from goal-directed paths. However, they may not work well in dense situations where the pedestrians make frequent stops or long-term predictions.

Main Results: In this chapter, we present a novel algorithm to learn pedestrian local and global movement patterns from sparse 2D pedestrian trajectory data using Bayesian Inference. Our approach is general, makes no assumption about pedestrian movement or density, and performs no pre-computation. We use the trajectory data information over a sequence of frames to predict the future pedestrian states using Ensemble Kalman

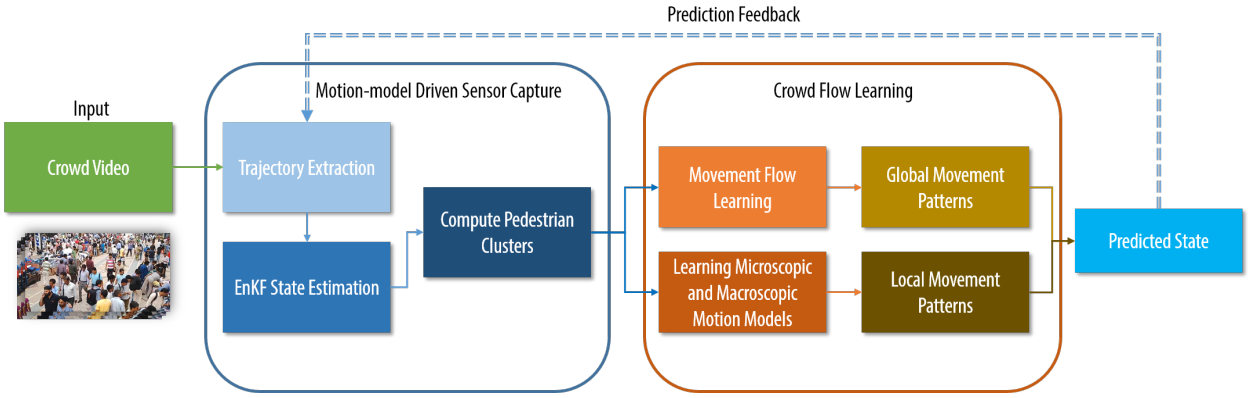


Figure 3.2: We highlight various components of our real pedestrian path prediction algorithm. Our approach computes both local and global movement patterns using Bayesian inference from 2D trajectory data and combines them to improve prediction accuracy.

Filters (EnKF) and Expectation Maximization (EM). The state information is used to compute movement-flow information of individual pedestrians and coherent pedestrian clusters using a mixture of motion models. The global movement features are combined with local motion patterns computed using Bayesian reciprocal velocity obstacles to compute the predicted state of each pedestrian. The combination of **G**lobal and **L**ocal **M**ovement **P**atterns (i.e. GLMP) corresponds to computing dynamically varying individualized motion model for each pedestrian. Overall, our approach offers the following benefits:

- Our algorithm is general and can compute global and local movement patterns in real-time with no prior learning.
- We can robustly handle sparse and noisy trajectory data generated using current online pedestrian trackers.
- We observe upto 18% increase in prediction accuracy as compared to prior real-time methods that are based on simple filters or only local movement patterns.

We highlight the performance of GLMP to predict the positions of pedestrians using trajectory data extracted from a variety of video datasets consisting of 30-400 pedestrians. Our approach can predict the positions of tens of pedestrians in around 40-50 milliseconds over long-intervals. Furthermore, we demonstrate its benefits over prior real-time prediction algorithms.

The rest of our chapter is organized as follows. Section II gives a brief overview of prior work in pedestrian path prediction, tracking, and motion models. We present our movement flow learning algorithm

in Section III and use that for path prediction. We highlight its performance on different crowd video datasets in Section IV and compare its performance with prior methods.

3.2 Related Work

In this section, we give a brief overview of prior work on motion models and pedestrian path prediction.

3.2.1 Path Prediction

Fulgenzi et al. (Fulgenzi et al., 2007) use a probabilistic velocity-obstacle approach combined with the dynamic occupancy grid; this method assumes constant linear velocity motion of the obstacles. DuToit et al. (Du Toit and Burdick, 2010) present a robot planning framework that takes into account pedestrians' anticipated future location information to reduce the uncertainty of the predicted belief states. Other techniques use potential-based approaches for robot path planning in dynamic environments (Pradhan et al., 2011). Some methods learn the trajectories from collected data. Ziebart et al. (Ziebart et al., 2009a) use pedestrian trajectories collected in the environment for prediction using Hidden Markov Models. Bennewitz et al. (Bennewitz et al., 2005) apply Expectation Maximization clustering to learn typical motion patterns from pedestrian trajectories, before using Hidden Markov Models to predict future pedestrian motion. Henry et al. (Henry et al., 2010) use reinforced learning from example traces, estimating pedestrian density and flow with a Gaussian process. Kretzschmar et al. (Kretzschmar et al., 2014) consider pedestrian trajectories as a mixture probability distribution of a discrete as well as a continuous distribution and then use Hamiltonian Markov chain Monte Carlo sampling for prediction. Kuderer et al. (Kuderer et al., 2012) use maximum entropy based learning to learn pedestrian trajectories and use a hierarchical optimization scheme to predict future trajectories. Many of these methods involve a priori learning, and may not work in new or unknown environments.

Trautman et al. (Trautman et al., 2013) have developed a probabilistic predictive model of cooperative collision avoidance and goal-oriented behavior for robot navigation in dense crowds. Guzzi et al. (Guzzi et al., 2013) present a distributed method for multi-robot human like local navigation. Variations of the Bayesian filters for pedestrian path prediction have been studied in (Schneider and Gavrila, 2013; Mogelmose et al., 2015). Some of these methods are not suitable for real-time applications or may not work well for dense crowds.

Figure 3.3: **Prediction Outputs** We test our approach on a variety of crowd datasets with varying density. Our approach had a benefit of upto 18% better prediction at a 5 second time horizon for some high-density datasets. Yellow lines represent past tracked trajectories whereas the red dots represent predicted motion.

3.3 Real-time Pedestrian Path Prediction

In this section, we present our real-time algorithm that learns movement flows from real-world pedestrian 2D trajectories that are extracted from video. Our approach involves no pre-computation or learning, and can be combined with real-time pedestrian trackers.

Fig. 2 gives an overview of our approach, including computation of movement flows and using them for pedestrian prediction. The input to our method consists of a live or streaming crowd video. We extract the initial set of trajectories using an online particle-filter based pedestrian tracker. These trajectories are time-series observations of the positions of each pedestrian in the crowd. The various components used in our algorithm are shown in the figure and explained below. The output is the predicted state of each agent that is based on learning the local and global pedestrian motion patterns.

3.3.1 Pedestrian State Estimation

The trajectories extracted from a real-world video tend to be noisy and may have incomplete tracks (Enzweiler and Gavrila, 2009); thus, we use Bayesian-inference technique to compensate for any errors and to compute the state of each pedestrian (Kim et al., 2014). At each time step, the observation of a pedestrian computed by a tracking algorithm corresponds to the position of each pedestrian on the 2D plane, denoted as $\mathbf{z}^t \in \mathbb{R}^2$. The observation function $h()$ provides $\mathbf{z}^t$ of each pedestrian's true state $\hat{\mathbf{x}}^\mathbf{t}$ with sensor error $\mathbf{r} \in \mathbb{R}^2$, which is assumed to follow a zero-mean Gaussian distribution with covariance Σ_r:

$$z^t = h(\hat{\mathbf{x}}^t) + \mathbf{r}, \mathbf{r} \sim N(0, \Sigma_r). \tag{3.1}$$

$h()$ is the tracking sensor output.

Figure 3.4: **Global vs Local Movement Patterns** The blue trajectories indicate prior tracked data. The red dots indicate local predicted patterns retrieved from learning macro and microscopic simulation models, The shaded (green-blue) path represent the global movement patterns learned from the path data in that cluster

We use the notion of a state-transition model $f()$ which is an approximation of true real-world pedestrian dynamics with prediction error $\mathbf{q} \in \mathbb{R}^6$, which is represented as a zero-mean Gaussian distribution with covariance Σ_q:

$$\mathbf{x}^{t+1} = f(\mathbf{x}^t) + \mathbf{q},\ \mathbf{q} \sim N(0, \Sigma_q). \tag{3.2}$$

We can use any local navigation algorithm or motion model for function $f()$, which computes the local collision-free paths for the pedestrians in the scene. More details are given below. Our algorithm uses an Ensemble Kalman Filter (EnKF) and Expectation Maximization (EM) with the observation model $h()$ and the state transition model $f()$ to estimate the most likely state $\mathbf{x}$ of each pedestrian (Kim et al., 2014). In particular, EnKF predicts the next state based on the transition model and the covariance matrix Σ_q and updates them whenever a new observation is available. The EM step computes Σ_q to maximize the likelihood of the state estimation.

Pedestrian Clusters Our approach is targeted towards computing the movement flows of pedestrians in dense settings. It is not uncommon for some nearby pedestrians to have similar flows. As a result, we compute clusters of pedestrians in a crowd based their positions, velocity, inter-pedestrian distance, orientations, etc. In particular, we use a bottom-up hierarchical clustering approach, as they tend to work better for small clusters. Initially, we assign each pedestrian to a separate cluster that consists of a single pedestrian. Next, we merge these clusters based on computing the distance between various features highlighted above.

Our approach is based on group-expand procedure (McPhail and Wohlstein, 1982) and we include many pedestrian movement related features to compute the clusters. We compute a connectivity graph among the pedestrians and measure the graph density based on intra-cluster proximity 2. Eventually, we use a macroscopic model to estimate the movement of each cluster and use this model to predict their global movement.

3.3.2 Global Movement Pattern

A key aspect of our approach is to compute global movement patterns that can be used to predict the state of each pedestrian. These movement patterns describe the trajectory-level motion or behavior at a certain position at time frame t. The patterns include the movement of the pedestrian during the past w frames, which we call *time window*, and the intended direction of the movement (preferred velocity) at this position.

In our formulation, we represent each movement feature vector as a six-dimensional vector:

$$\mathbf{b} = [\mathbf{p}\ \mathbf{v}^{avg}\ \mathbf{v}^{pref}]^T, \tag{3.3}$$

where $\mathbf{p}$, $\mathbf{v}^{avg}$, and $\mathbf{v}^{pref}$ are each two-dimensional vectors representing the current position, average velocity during past w frames, and estimated preferred velocity computed as part of state estimation, respectively. $\mathbf{v}^{avg}$ can be computed from $(\mathbf{p}^t - \mathbf{p}^{t-w})/w * dt$, where dt is the time step.

We use the notion of average velocity over the last w frames as that provides a better estimate of pedestrian movement. In a dense setting, some pedestrians may suddenly stop or change their local directions as they interact with other pedestrians. As a result, the duration of the time window, w, is set based on the characteristics of a scene. If we use small time windows, the movement flows will be able to capture the details in dynamically changing scenes. On the other hand, larger time windows tend to smooth out abrupt changes in pedestrian motion and are more suitable for scenes that have little change in pedestrians' movement.

At every w steps, we compute the new trajectory features for each pedestrian in the scene, using Equation 4.1. Moreover, we group the similar features and find K most common trajectory patterns, which we call *global movement patterns*. We use recently observed behavior features to learn the time-varying

movement flow. This set of K global movement patterns $B = \{B_1, B_2, ..., B_K\}$ is computed as follows:

$$\underset{B}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{b_i \in B_k} dist(b_i, \mu_k), \tag{3.4}$$

where b_i is a movement feature vector, μ_k is a centroid of each flow movement pattern, and $dist(b_i, \mu_k)$ is a distance measure between the arguments. In our case, the distance between two feature vectors is computed as

$$\begin{aligned} dist(b_i, b_j) &= c_1 \left\| \mathbf{p}_i - \mathbf{p}_j \right\| \\ &+ c_2 \left\| (\mathbf{p}_i - \mathbf{v}_i^{avg} w\ dt) - (\mathbf{p}_j - \mathbf{v}_j^{avg} w\ dt) \right\| \\ &+ c_3 \left\| (\mathbf{p}_i + \mathbf{v}_i^{pref} w\ dt) - (\mathbf{p}_j - \mathbf{v}_j^{pref} w\ dt) \right\|, \end{aligned} \tag{3.5}$$

which corresponds to the weighted sum of the distance among three points: current positions, previous positions and estimated future positions that are extrapolated using v^{pref}, c_1, c_2, and c_3 as the weight values. Comparing the distance between the positions rather than mixing the points and the vectors eliminates the need to normalize or standardize the data. We use the movement feature of the cluster to compute the predicted state at time t, $\mathbf{S_t^g}$.

3.3.3 Local Movement Pattern

During each frame, some of the pedestrians are modeled as discrete agents, while the clusters are treated using macroscopic techniques. Based on the observations and state information, we estimate the motion model for these discrete agents and pedestrian clusters. For each individual pedestrian represented as a discrete agent, we compute the motion model that best fits its position as tracked over recent frames i.e. we compute the features per-agent and predict motion patterns locally. We choose the "best" local motion model from a fixed set of choices. The common choices are based on social forces, reciprocal velocity obstacles or Boids. In our case, the "best" motion model is the one that most accurately matches the immediate past states based on a given error metric. This "best" motion model is computed using a local optimization algorithm 2, which automatically finds the motion model parameters that minimize that error metric.

A motion model (microscopic or macroscopic) is defined as an algorithm f (defined in Equation 3.2) that starts with a collection of agent states $\mathbf{X}_t$, and computes the new states $\mathbf{X}_{t+1}$ for these agents. It represents

their motion over a timestep towards the agents' immediate goals $\mathbf{G}$:

$$\mathbf{X}_{t+1} = f(\mathbf{X}_t, \mathbf{G}, \mathbf{P}), \tag{3.6}$$

where $\mathbf{P}$ denotes the individual pedestrian parameters. Formally, at any timestep t, we define the agents' (k+1)-states (as computed by the tracker and state estimation) $\mathbf{S}_{t-k:t}$:

$$\mathbf{S}_{t-k:t} = \bigcup_{i=t-k}^{t} \mathbf{S}_i. \tag{3.7}$$

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

$$f(\mathbf{S}_{t-k:t}, \mathbf{P}) = \bigcup_{i=t-k}^{t} f(\mathbf{X}_i, \mathbf{G}, \mathbf{P}), \tag{3.8}$$

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:

$$\mathbf{P}_t^{opt,f} = \underset{\mathbf{P}}{\operatorname{argmin}}\, error(f(\mathbf{S}_{t-k:t}, \mathbf{P}), \mathbf{S}_{t-k:t}), \tag{3.9}$$

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.

3.3.4 Prediction Output

For every pedestrian, we compute both the global and local movement patterns separately. In practice, we observed that for lower density scenarios, local movement patterns are more useful than global patterns and vice-versa. Our final predicted state is a weighted average of the individual predicted states generate from the local and global patterns as:

$$\mathbf{S_t^p} = (1 - w) * \mathbf{S_t^l} + w * \mathbf{S_t^g}, \tag{3.10}$$

where $\mathbf{S_t^p}$ is the final predicted state at time t, $\mathbf{S_t^l}$ is the state predicted from the local patterns and the $\mathbf{S_t^g}$ is the state predicted from global patterns or from the movement flows. As a general rule of thumb, w varies

Figure 3.5: **Improved Prediction** We demonstrate the improved accuracy of our pedestrian path prediction algorithm (GLMP) over prior real-time prediction algorithms (BRVO, Const Vel, Const Accel).

from 0 to 1 and is computed based on the pedestrian density. We use a larger weight for higher density. In order to perform long-term predictions (5-6 seconds or even longer), we tend to increase w as the global movement patterns provide better estimates for pedestrian position.

3.4 Analysis

In this section, we highlight the prediction results using GLMP algorithm and compare its performance with prior method. We have applied it to the 2D trajectories generated from different crowd videos and compared the prediction accuracy with the ground truth data, that was also generated using a pedestrian tracker. The underlying crowd videos have different pedestrian density corresponding to low (i.e. less than 1 pedestrian per squared meter), medium (1-2 pedestrians per squared meter), and high (more than 2 pedestrians per squared meter). We highlight the datasets, their crowd characteristics, and the prediction accuracy of different real-time algorithms for short-term and long-term prediction in Table 1. We also analyze the accuracy of our appraoch based on varying the pedestrian density (Fig. 7) and the frame sampling rate

(Fig. 8). The performance of the method with noisy data (i.e sensor noise) is also analyzed. Finally, we perform a qualitative and quantitative comparison to other real-time pedestrian path prediction algorithms.

Dataset	Challenges	Density	# Tracked	ConstVelocity		Kalman Filter		BRVO		GLMP	
				1 sec	*5 secs*	*1 sec*	*5 secs*	*1 sec*	*5 secs*	*1 sec*	*5 secs*
NDLS-1	BV, PO, IC	High	131	55.3%	32.0%	53.1%	37.9%	56.5%	42.0%	60.2%	51.2%
IITF-1	BV, PO, IC, CO	High	167	63.5%	33.4%	63.9%	39.1%	65.3%	41.8%	71.2%	50.5%
IITF-3	BV, PO, IC, CO	High	189	61.1%	29.1%	63.6%	31.0%	67.6%	37.5%	68.4%	45.7%
IITF-5	BV, PO, IC, CO	High	71	59.2%	28.8%	61.7%	29.1%	62.9%	30.1%	64.6%	40.0%
NPLC-1	BV, PO, IC	Medium	79	76.1%	63.9%	78.2%	65.8%	79.9%	69.0%	82.3%	72.5%
NPLC-3	BV, PO, IC, CO	Medium	144	77.9%	70.1%	79.1%	71.9%	80.8%	74.4%	84.3%	78.1%
Students	BV, IC, PO	Medium	65	65.0%	58.2%	66.9%	61.0%	69.1%	63.6%	72.2%	66.8%
Campus	BV, IC, PO	Medium	78	62.4%	57.1%	63.5%	59.0%	66.4%	59.1%	69.6%	59.5%
seq_hotel	IC, PO	Low	390	74.7%	67.8%	76.7%	68.3%	76.9%	69.2%	79.5%	70.1%
Street	IC, PO	Low	34	78.1%	70.9%	78.9%	71.0%	81.4%	71.2%	83.8%	72.7%

Table 3.1: Crowd Scene Benchmarks: We highlight many attributes of these crowd videos, including density and the number of tracked pedestrians. 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). We highlight the results for short-term prediction (1 sec) and long term prediction (5 sec). We notice that our GLMP algorithm results in higher accuracy for long-term prediction and dense scenarios. More details are given in Section IV(B).

We include comparisons to constant velocity (ConstVelocity) and constant acceleration (ConstAccel) motion models, which are widely used for pedestrian tracking and prediction in robotics and computer vision (Del Bimbo and Dini, 2011). We also compare the accuracy with recent methods that use more sophisticated motion models (LTA and ATTR) to compute local movement patterns (Pellegrini et al., 2009a; Yamaguchi et al., 2011a). Finally, we also compare the accuracy with the Bayesian reciprocal velocity obstacle (BRVO) algorithm (Kim et al., 2014) that computes a more individualized motion model for estimating local movement patterns.

3.4.1 Noisy Data

Sensor noise is an important concern in pedestrian prediction algorithms. In order to evaluate the impact of noise, we add synthetic noise to the datasets and compare the performance of GLMP vs. other algorithms on these benchmarks: IITF (Bera and Manocha, 2014b), ETH and Campus (Pellegrini et al., 2009b) datasets.

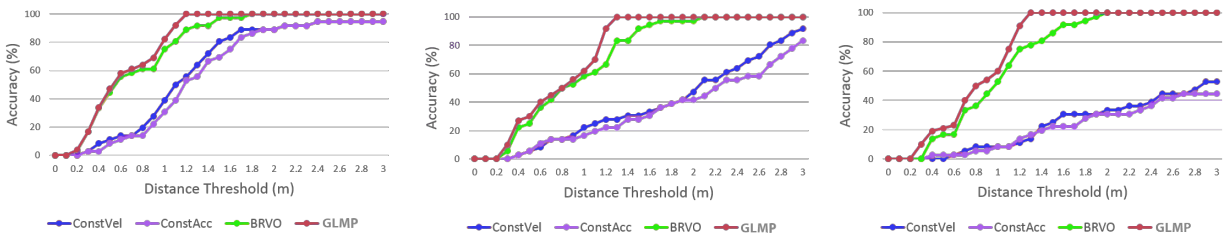
Fig. 6 compares the prediction accuracy of GLMP, constant velocity, constant acceleration and BRVO, by comparing the predicted positions to the actual ground truth data extracted using pedestrian trackers. We use these noise levels, 0.05m, 0.1m, and 0.15m to simulate different sensor variations. During the prediction step, we assume that no further information is given when we are predicting the future state, and our best guess is

that the pedestrians move according to their preferred velocity computed using the movement patterns. For GLMP, the pedestrian's movement direction changes when there is any interaction with obstacles or other pedestrians as observed based on local and global movement patterns. Fig. 7 shows the fraction of correctly predicted paths within varying accuracy thresholds. At an accuracy threshold of 0.5m, GLMP has higher accuracy than BRVO and offers considerable benefits over constant velocity, constant acceleration models even with little noise. As the noise increases, the benefit in prediction accuracy using GLMP also increases.

3.4.2 Long-term Prediction Accuracy

Being able to predict a trajectory over a longer time-horizon is important for service robots and autonomous vehicles. Our approach is able to perform long-term prediction (5-6 seconds) with much higher accuracy than prior methods (see Table 1).

We use a simple prediction metric to evaluate the accuracy of both, long and short term prediction. A prediction is counted as "successful" when the estimated mean error between the prediction result and the ground-truth value at that time instance is less than 0.8 meter in ground space coordinates. The average human stride length is about 0.8 meter and we consider the prediction to be incorrect if the mean error is more than this value. We define prediction accuracy as the ratio of the number of "successful" predictions and total number of tracked pedestrians in the scene. We use our algorithm for long and short term prediction across a large number of datasets, highlighted in Table 1.



(a) 0.05m (b) 0.1m (c) 0.15m

Figure 3.6: **Prediction Accuracy vs. Sensor Error (higher is better)** We increase the sensor noise (Gaussian) from left to right and highlight the prediction accuracy across various distance thresholds. The X-axis represents the percentage of correctly predicted paths within varying accuracy thresholds. In this GLMP results in more accurate predictions, as compared to BRVO, Constant Velocity, Constant Acceleration. As the sensor noise increases (c), we observe more significant benefit.

3.4.3 Varying the Pedestrian Density

We use a variation of crowd videos with different densities (Low, Medium and High) and compare GLMP's error to that of BRVO, constant velocity and constant acceleration models (see Fig. 3.7). Both the constant velocity and constant acceleration models have large variations in error for different regions of the scenario with varying densities. In contrast, the GLMP approach performs well across all densities because it can dynamically adapt the parameters for each agent for each frame and learn global as well as local motion patterns. We observe higher accuracy benefits in high density scenarios due to the computation of global movement patterns.

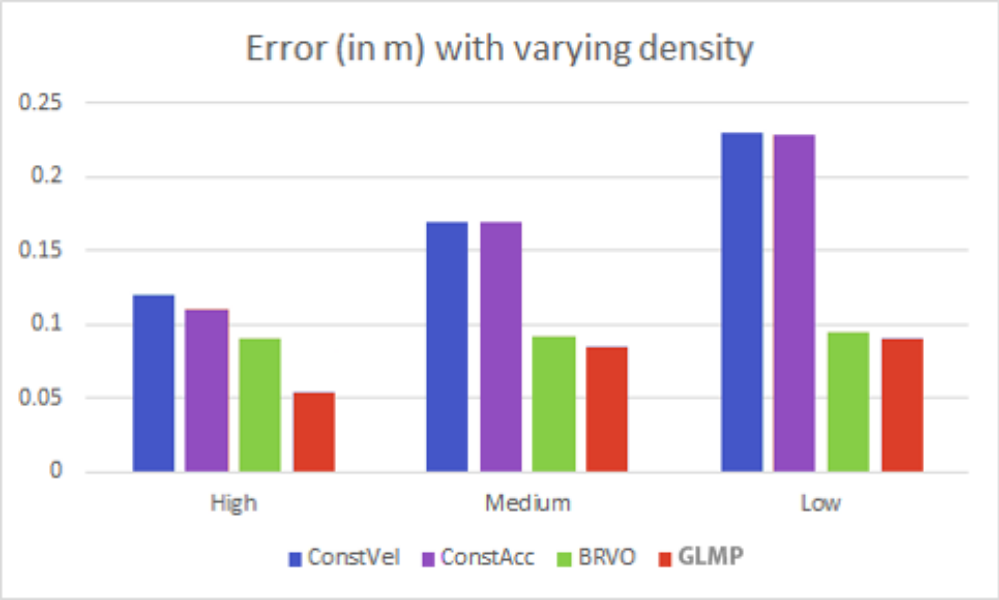


Figure 3.7: **Errors for varying Pedestrian Densities (lower is better).** In low-density scenarios, local movement patterns (e.g BRVO) are able to predict the positions well, but are more accurate than const. velocity and const. acceleration. We observe improved accuracy with GLMP, as the pedestrian density increases.

3.4.4 Varying the Sampling Rate

GLMP works very well on data with very low frame-rate, when the video or data is sampled at large intervals. In low FPS videos, there is less temporal direct pixel overlap between frames. In a way, this is also a metric for evaluating the accuracy of our prediction algorithm. In order to evaluate the performance, we evaluated the accuracy on the Street, seq_hotel, seq_eth and IITF datasets at varying frame-rates. Fig. 3.8 shows the graph of the mean error versus the sampling interval. The graph in the figure shows that our method

performs very well compared to BRVO, constant velocity and constant acceleration model across all the sampling rates.

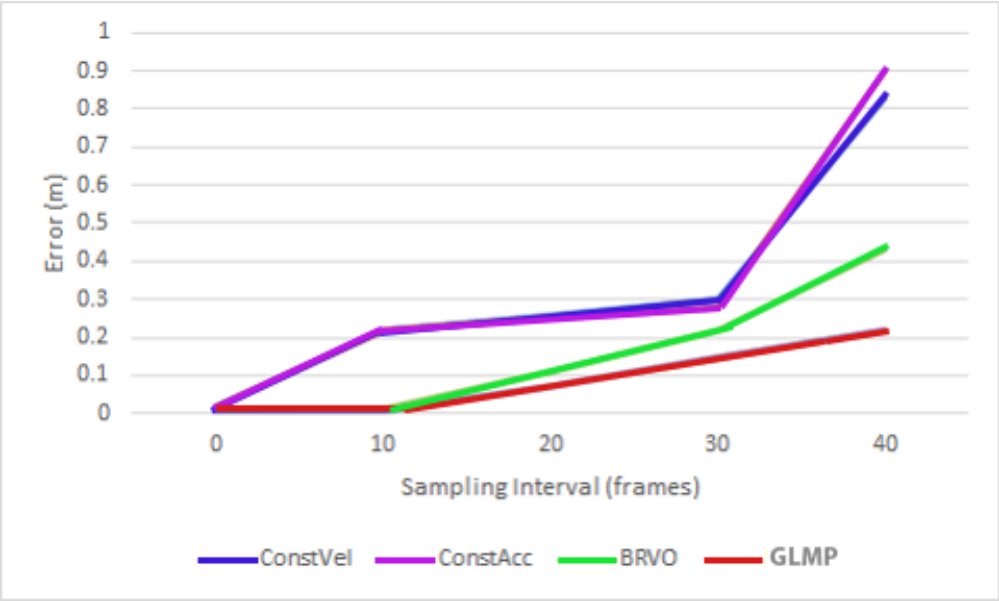


Figure 3.8: **Error vs Sampling Interval** As the sampling interval increases, the error from Constant Velocity, Constant Acceleration and BRVO grows much larger than that from GLMP.

3.4.5 Comparison with Prior Methods

We directly compare our results with the prediction results of LTA (Pellegrini et al., 2009b) and ATTR (Yamaguchi et al., 2011a), which report performance numbers for some of the same benchmarks. Unlike GLMP, both these methods require offline preprocessing or annotation. We also compare GLMP with BRVO along with LTA and ATTR on Street, NDLS, IITF, seq_hotel and seq_eth datasets, all sampled every 1.5 seconds, and measure mean prediction error for every agent in the scene during the entire video sequence.

The metric used was error reduction comparison, and is measured as improvement in percentage of error reduction over the LIN model for different algorithms. The results are shown in Fig. 3.9. Our method outperforms LTA and ATTR with 24-47% error reduction rate across the three different scenarios. LTA and ATTR use the ground truth destinations for prediction; LTA+D and ATTR+D use destinations learned offline, as explained in (Yamaguchi et al., 2011a); ATTR+D uses grouping information learned offline. Even though GLMP is an online and real-time method, it shows significant improvement in prediction accuracy on all the datasets, producing less error than other approaches.

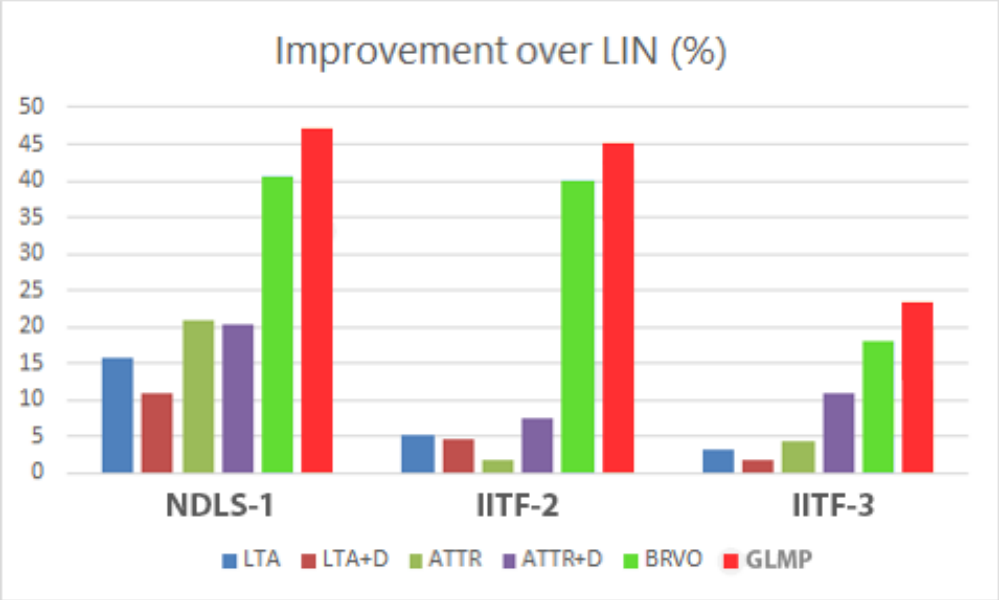


Figure 3.9: **Error Reduction Comparison** We compare the improvements of our method, LTA, ATTR and BRVO over LIN (linear velocity) model. Our method (GLMP) outperforms LTA, ATTR with 24-47% error reduction rate in all three different scenarios.

3.4.6 Implementation Details

We tested these algorithms on an IntelHaswell, Corei7-4771 Processor (4 Cores) with an 8MB Cache, 3.90 GHz and IntelHD Graphics 4600. Our work is implemented in C++, and some components use OpenMP and OpenCL toexploit multiple cores. We adopted a agent-level parallelism: individual pedestrian computations are distributed across the CPU cores (except for the motion-model computations, where pedestrian behavior is interlinked and tasks are highly sequential)

3.5 Limitations, Conclusions, and Future Work

We present a novel real-time algorithm for pedestrian path prediction. The main idea is to learn the local and global movement patterns using Bayesian inference. Our approach can handle low as well as high density videos and is useful for short-term and long-term prediction. We have highlighted its performance on many benchmarks and demonstrate the improvements in accuracy over prior real-time algorithms.

Our approach has some limitations. The underlying formulation does not model many other aspects of pedestrian behavior, including physiological and psychological pedestrian traits as well as age, gender or external environmental factors. The estimation techniques relies on Bayesian inferences and that may not work

well in some cases. In terms of future work, we would like to overcome these limitations. Furthermore, we would like to evaluate their performance with robots, e.g. service robots or autonomous vehicles, navigating through pedestrians.

CHAPTER 4
Learning Pedestrian Behaviors

4.1 Introduction

Modeling and classifying the behavior of different pedestrians in a crowd is an important problem in various domains including psychology, robotics, pedestrian dynamics, and behavior learning. Even simple tasks like walking towards a goal position involve several complex decisions such as figuring out the most efficient path or route, and choosing between the various available paths to avoid collisions. According to *Convergence Theory* (Turner and Killian, 1987), a well-known approach used in sociology and economics, crowd behavior is not a sole product of the crowd itself; rather, it is defined by the individual pedestrians in that crowd. As a result, it is important to accurately predict the behavior of individuals and their interactions with the environment to capture realistic, heterogeneous crowd behaviors.

Recent advances in sensor technologies have made it easier to capture high resolution videos of pedestrians and crowds. Moreover, surveillance cameras are frequently used in public places and buildings for monitoring human behaviors. In this chapter, we address the problem of classifying the behaviors of different pedestrians in a crowd video based on their movement patterns and use these patterns for crowd behavior prediction. Besides surveillance, these techniques are also useful for architectural design and collision-free navigation of robots or autonomous vehicles in crowded scenarios.

Figure 4.1: Crowd Behavior Learning/Prediction: Our approach can automatically classify the behavior of each pedestrian in a large crowd. We highlight its application for the 2017 Presidential Inauguration crowd video at the National Mall at Washington, DC (courtesy PBS): (1) original aerial video footage of the dense crowd; (2) a synthetic rendering of pedestrians in the red square based on their personality classification: aggressive (**orange**), shy (**black**), active (**blue**), tense (**purple**), etc; (3) a predicted simulation of 1M pedestrians in the space with a similar distribution of personality traits.

Many factors including biological, developmental, and situational variations, along with individual personalities, govern people's overall behavior. We mainly focus on capturing the variations in behavior that arise as humans navigate the physical world and avoid collisions. In general, categorizing the variety of personalities that humans exhibit can be hard. Psychologists have proposed different models to represent these variations, but they have some limitations (Harvey et al., 1995). Therefore, we base our classification

on *Personality Trait Theory*, which proposes that a wide range of variations in behavior is primarily the result of a small number of underlying traits. It is also important to model many external or environmental factors, including surrounding pedestrians and the crowd's movement flow for estimation and prediction.

Main Results: We present a novel learning algorithm to classify pedestrian behaviors based on their movement patterns. We extract the trajectory of each pedestrian in a video and use a combination of Bayesian learning and pedestrian dynamics techniques to compute the local and global characteristics at interactive rates. The local characteristics include the time-varying motion model that is used to compute the personality traits. We also present new statistical algorithms to learn high-level characteristics and global movement patterns. We combine these characteristics with Eysenck's 3-factor PEN model (Eysenck and Eysenck, 1985) and characterize the personality into six weighted behavior classes: *aggressive, assertive, shy, active, tense,* and *impulsive*. We also use the individual personalities to predict the state of the crowd under different environmental scenarios.

To the best of our knowledge, this is the first approach that can automatically identify the behavior of each pedestrian in a crowd. We have evaluated its accuracy with a user study (88.48%) and evaluated its performance on different videos with tens of pedestrians. One example is the large crowd gathered in Washington, DC for the Presidential Inauguration (January 2017) using *PBS HD* video footage (see Figure 1). We also want to be clear and mention that our definition for behavior is restricted to specific pedestrian level "motion patterns" observed in crowds. Our approach offers many benefits:

1. Robust: Our approach is robust, can account for noise in the pedestrian trajectories, and classifies the behavior using time-varying pedestrian movement dynamics.

2. General: Our approach is applicable to indoor and outdoor crowd videos and makes no assumption about their size or density.

3. Crowd Analysis and Prediction: Our approach can be used to analyze and estimate the future movement or behavior of the crowd. Furthermore, it can be used to predict different scenarios based on the behaviors and global characteristics, e.g., the distribution and density of a large crowd at the National Mall in Figure 4.1.

The rest of the chapter is organized as follows. Section 2 provides an overview of related work in video-based crowd analysis and personality models. We introduce the terminology and present our algorithm for computing the local and global characteristics in Section 3. We highlight the performance on challenging benchmarks and describe results from our user evaluation in Section 4.

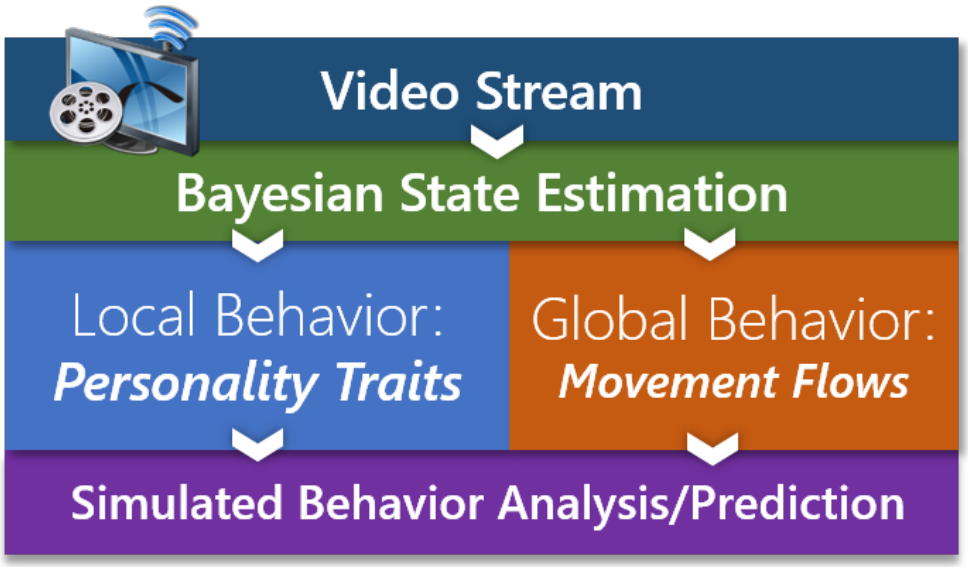


Figure 4.2: Our method takes a streaming crowd video as an input. We compute the state of pedestrians in the crowd, as explained in Section 3. Based on the state information, we learn local and global behavior properties, which are combined for behavior classification and prediction.

4.2 Related Work

In this section, we give a brief overview of prior work on video-based crowd analysis, behavior classification, and personality models.

4.2.1 Video-Based Crowd Analysis

There is extensive work in computer vision, multimedia, and robotics that analyzes the behaviors and movement patterns in crowd videos, as surveyed in (Li et al., 2015; Borges et al., 2013). The main objectives of these work include human behavior understanding and crowd activity recognition for detecting abnormal behaviors (Hu et al., 2004). Many of these methods use a large number of training videos to learn the patterns offline (Zen and Ricci, 2011; Solmaz et al., 2012). Other methods utilize motion models to learn crowd behaviors (Pellegrini et al., 2012) or use machine learning algorithms (Zhou et al., 2012b; Cheung et al., 2016). Some techniques focus on classifying the most common behavior patterns in a given scene using offline learning. These include activity prototypes using a convex learning algorithm (Zen and Ricci, 2011) and detection of popular behavior patterns like bottlenecks, fountainheads, lanes, arches, and blocking (Solmaz et al., 2012).

Crowd behavior learning using motion or simulation has been used for different applications. Parameter learning has been used to predict pedestrian motion for tracking (Pellegrini et al., 2012). However, these

techniques use either manual selection or offline learning techniques to estimate the goal positions. Other researchers have used low-density tracking data to learn agent intentions (Musse et al., 2007) or use online Bayesian motion-prediction methods for human-robot interactions, data-driven crowd simulation (Kim et al., 2016), and offline training (Zhong et al., 2015).

4.2.2 Pedestrian Behavior Modeling

Different approaches have been used to model pedestrian behavior. (Funge et al., 1999) use cognitive modeling to empower agents to plan and perform high-level tasks (Godoy et al., 2016). Other approaches use personality models to simulate the behavior of pedestrians and crowds including the OCEAN model (Durupinar et al., 2011), the MBTI model (Salvit and Sklar, 2011), and Personality Trait theory (Guy et al., 2011) and General Adaptation Syndrome Theory (Kim et al., 2012). However, these methods only take into account local motion models, not the global characteristics.

4.3 Pedestrian Behavior Computation

In this section, we present our interactive algorithm, which classifies pedestrian behavior using 2D, real-world trajectories that are extracted from video. Our approach can be combined with almost any real-time pedestrian tracker that works on dense crowd videos. Figure 4.2 gives an overview of our approach. Our method takes a live or streaming crowd video as an input. We extract the initial set of pedestrian trajectories using an online pedestrian tracker. We learn the pedestrian motion model parameters using statistical methods and learn global and local behavior characteristics. These can be used to predict the future state of the pedestrian or the overall crowd.

Motion Model: $\mathbf{P} \in \mathbb{R}^5$ denotes the set of parameters for the motion model. The motion model corresponds to the local navigation rule or scheme that each pedestrian uses to avoid collisions with other pedestrians or obstacles. Our formulation is based on the RVO velocity-based motion model (van den Berg et al., 2008a). In this model, the motion of each pedestrian is governed by these five characteristics: *Neighbor Dist, Maximum Neighbors, Planning Horizon, Radius, and Preferred Speed*. Our approach can also be combined with other models based on social forces or Boids.

4.3.1 Personality Trait Classification

Psychologists have proposed various ways of characterizing the personalities exhibited by pedestrians. Our work builds on Trait Theories of Personalities, a theory that categorizes people's behavior based on a small number of personality traits (Pervin, 2003). The overall goal is to automatically classify every pedestrian in a crowd. In particular, we characterize each pedestrian behavior based on a weighted combination of different personality traits that are inferred based on his or her movement pattern and interactions with other pedestrians and obstacles in the environment. We use the well-known Personality Trait Theory from psychology and the Eysenck 3-factor model (Eysenck and Eysenck, 1985) to classify such behaviors. This model identifies three major factors that characterize the personality: *Psychoticism*, *Extraversion*, and *Neuroticism* (commonly referred to as **PEN**). Each of these three traits has been linked to a biological basis such as the levels of testosterone, serotonin, and dopamine present in ones body.

In our case, each pedestrian's personality is identified based on how they exhibit each of these three traits. Each pedestrian is then classified into one of six weighted behavior classes: aggressive, assertive, shy, active, tense, and impulsive (see Fig. 4.3). We chose these six particular behavior characteristics because they are useful in describing pedestrians' behaviors, and span the space covered by the PEN model, with at least two adjectives for each PEN trait (Pervin, 2003). We classify each pedestrian's behavior using these traits, and our fundamental assumption is that these behaviors are captured by a weighted combination of these six traits.

A key issue in this formulation is defining a mapping between the five motion model parameters for the RVO model described in Section 3.1 and these six traits. We make use of the data-driven mapping presented in (Guy et al., 2011) to derive such a mapping that adopts the results of a user study and derives a linear model of the mapping as:

$$\mathbf{B} = \begin{pmatrix} Aggressive \\ Assertive \\ Shy \\ Active \\ Tense \\ Impulsive \end{pmatrix} = \mathbf{M}_{mat} * \begin{pmatrix} \frac{1}{13.5}(Neighbor\ Dist - 15) \\ \frac{1}{49.5}(Max.\ Neighbors - 10) \\ \frac{1}{14.5}(Planning\ Horiz. - 30) \\ \frac{1}{0.85}(Radius - 0.8) \\ \frac{1}{0.5}(Pref.\ Speed - 1.4) \end{pmatrix}$$

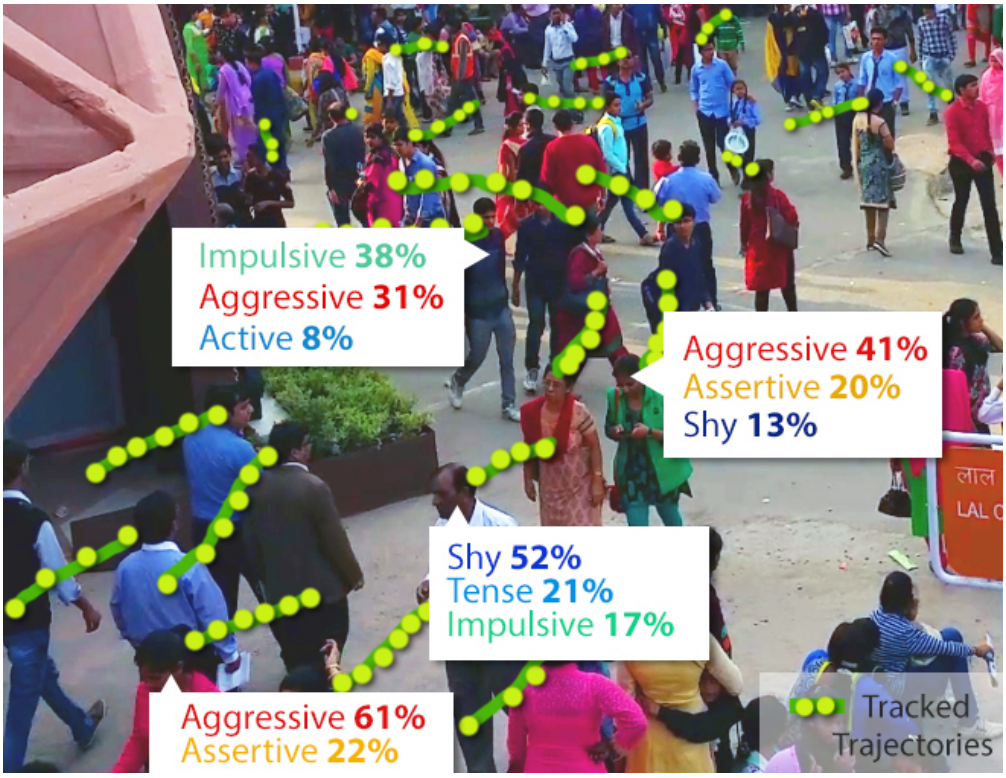


Figure 4.3: Personality Classification: We identify the personality of each tracked pedestrian based on pedestrian dynamics and the motion model. Each pedestrian is automatically classified using a weighted combination of different personality traits.

where,

$$\mathbf{M_{mat}} = \begin{pmatrix} -0.02 & 0.32 & 0.13 & -0.41 & 1.02 \\ 0.03 & 0.22 & 0.11 & -0.28 & 1.05 \\ -0.04 & -0.08 & 0.02 & 0.58 & -0.88 \\ -0.06 & 0.04 & 0.04 & -0.16 & 1.07 \\ 0.10 & 0.07 & -0.08 & 0.19 & 0.15 \\ 0.03 & -0.15 & 0.03 & -0.23 & 0.23 \end{pmatrix}.$$

Even though our approach is general, the mapping (M_{mat}) is specific to the RVO motion model and the user study described in (Guy et al., 2011).

4.3.2 Global Characteristics

We use an Ensemble Kalman Filter (EnKF) and Expectation Maximization (EM) to estimate the most likely state $\mathbf{x}$ of each pedestrian (as in Section 3.3.1).

Our approach extends the method presented in (Kim et al., 2016). The global dynamics consist of factors that govern pedestrians' trajectory behaviors in a group or crowd, i.e., the factors that influence a pedestrian's overall movement or flow. We use two main components to describe the global dynamics: start point of each pedestrian in the scenario and movement flows. We then use them to analyze the global behavior of each pedestrian. Formally, we represent these dynamic characteristics for each pedestrian with a vector-valued function, $f()$, whose initial value is determined by the function, $E()$:

$$\mathbf{x}^{t+1} = f(t, \mathbf{x}^t) = [P(\mathbf{x}^t)\ I(\mathbf{x}^t)\ G(t, \mathbf{x}^t)]^{\mathbf{T}}; \quad \mathbf{x}^0 = E(t^0).$$

For each pedestrian in the crowd, the function $G : \mathbb{R} \times \mathbb{R}^6 \times \mathbb{S} \rightarrow \mathbb{R}^2$ maps time t, the current state of the pedestrian $\mathbf{x} \in \mathbf{X}$, and the current state of the environment $\mathbf{S} \in \mathbb{S}$ to a preferred velocity $\mathbf{v}^{pref}$. Function $I : \mathbb{R}^6 \times \mathbb{S} \rightarrow \mathbb{R}^2$ represents the RVO motion model that is used to compute the current velocity $\mathbf{v}^c$ for collision-free interactions with other pedestrians and obstacles. The function $P : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ computes the position given $\mathbf{v}^c$ and $E : \mathbb{R} \rightarrow \mathbb{R}^2$ computes the initial position for time t_0, which is the time at which a particular pedestrian enters the environment. The three components of the pedestrian dynamics (start point, movement flow, and local collision-free navigation) are mapped to the functions $E()$, $G()$, and $I()$, respectively. We use Bayesian inference to compute $E()$ and $G()$ from the 2D trajectory data of the pedestrians.

Movement Feature The movement features describe the characteristics of the trajectory behavior at a certain position at time frame t. These movement features of different pedestrians are grouped together and form a cluster that represents the movement flow. The characteristics include the movement of the pedestrians during the past w frames, which we call the *time window*, and the intended direction of the movement (i.e. the preferred velocity) at this position. In our case, the movement feature vector is represented as a six-dimensional vector:

$$\mathbf{g} = [\mathbf{p}\ \mathbf{v}^{avg}\ \mathbf{v}^{pref}]^T, \tag{4.1}$$

where $\mathbf{p}$, $\mathbf{v}^{avg}$, and $\mathbf{v}^{pref}$ are each two-dimensional vectors that correspond to the current position, average velocity during past w frames, and estimated preferred velocity computed as part of state estimation, respectively. $\mathbf{v}^{avg}$ can be computed from $(\mathbf{p}^t - \mathbf{p}^{t-w})/w * dt$, where dt is the time-step.

At every w steps, we compute the new behavior features for each pedestrian using Equation 4.1. We group similar features and find K most common behavior patterns, which correspond to the *movement flow clusters*. We use recently observed behavior features to learn the time-varying movement flow. We use the k-means data clustering algorithm to classify these features into K movement flow clusters. In our case, K and N^f are user defined values that represent the total number of the clusters and the total number of collected behavior features, respectively, and $K \leq N^f$. A set of movement-flow clusters $G = \{G_1, G_2, ..., G_K\}$ is computed as follows:

$$\underset{G}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{g_i \in G_k} dist(b_i, \mu_k), \tag{4.2}$$

where g_i is a movement feature vector, μ_k is a centroid of each flow cluster, and $dist(g_i, \mu_k)$ is a distance measure between the arguments. In our case, the distance between two feature vectors is computed as

$$\begin{aligned} dist(g_i, g_j) = c_1 \left\| \mathbf{p}_i - \mathbf{p}_j \right\| \\ + c_2 \left\| (\mathbf{p}_i - \mathbf{v}_i^{avg} w\ dt) - (\mathbf{p}_j - \mathbf{v}_j^{avg} w\ dt) \right\| \\ + c_3 \left\| (\mathbf{p}_i + \mathbf{v}_i^{pref} w\ dt) - (\mathbf{p}_j - \mathbf{v}_j^{pref} w\ dt) \right\|, \end{aligned}$$

which corresponds to the weighted sum of the distance among three points: current positions, previous positions, and estimated future positions.

Estimation of Start Points Start points for each point correspond to the estimated position when that pedestrian enters the scene. These starting positions and timings for each position are very important and are used to compute the global behavior. We use a multivariate Gaussian mixture model to learn the time-varying distribution of the start points. We define $E()$ as the function that provides a position sampled from the learned distributions. We assume that the distribution of start points, e, from which the function $E()$ samples, is a mixture of J components and that each of the components is a multivariate Gaussian distribution of a two-dimensional random variable, $\mathbf{p}$, with a set of parameters $\Theta = (\alpha_1, \cdots, \alpha_J, \theta_1, \cdots, \theta_J)$:

$$e(\mathbf{p}|\Theta) = \sum_{j=1}^{J} \alpha_j e_j(\mathbf{p}|\mu_j, \theta_j), \tag{4.3}$$

$$e_j(\mathbf{p}; \theta_j) = \frac{1}{2\pi|\Sigma_j|^{1/2}} exp(-\frac{1}{2}(\mathbf{p} - \mu_j)^T \Sigma_j^{-1}(\mathbf{p} - \mu_j)). \tag{4.4}$$

Each component e_j is a Gaussian distribution given by the parameters $\theta_j = (\mu_j, \Sigma_j)$, where μ_j is the mean of the component j and Σ_j is a 2×2 covariance matrix. α_j is a mixture weight, which is the probability that a point $\mathbf{p}$ belongs to the component j. $\alpha_j \in [0, 1]$ for all i and the sum of α_js are constrained to 1 ($1 = \sum_{j=1}^{J} \alpha_j$). From an initial guess of the parameters θ_j, we perform EM to learn these parameters $\theta_j = (\mu_j, \Sigma_j)$ from the given start points collected from the real pedestrian trajectories.

4.3.3 Simulated Crowd Behavior Analysis & Prediction

Given a video, we can classify the personality trait, B, of each pedestrian (PTC) and compute the global features (GMD) corresponding to the start points, e, and the movement flows, G. We can combine this local and global information to predict the future position or state of each pedestrian and learn the shape, distribution, or behavior of the crowd in different environmental conditions. These different conditions may correspond to a change in the obstacle locations, an increase or decrease in the number of pedestrians in the scene, or a change in crowd density, but still maintain the original distribution of pedestrian behaviors and movement flows. We demonstrate our approach's performance on the *PBS* video stream from the 2017 Presidential Inauguration ceremony at Washington, DC, USA. We extract a representative sample of the crowd by selecting 130 pedestrians from a camera angle and learn their behaviors. As part of our crowd prediction, we changed the number of pedestrians in the scenario (e.g., 1 million pedestrians), and estimated the distribution and shape of the resulting crowd at the National Mall, as shown in Figure 1. The resulting crowd of 1M pedestrians has the same behavior classification as the original 130 representative pedestrians.

4.4 Performance and User Evaluation

In this section, we highlight the performance of our algorithms on different crowd videos. Furthermore, we evaluate the accuracy of our personality classification algorithm with a user study.

4.4.1 Performance Evaluation

We have applied our novel algorithms to the 2D pedestrian trajectories generated and extracted from different crowd videos, as shown in Table 1. The pedestrian density in these crowd videos varies from low-density (less than 1 pedestrian per square meter) to medium-density (1-2 pedestrians per square meter), to high-density (more than 2 pedestrians per square meter). We have implemented our system on a Windows

10 desktop PC with Intel Xeon E5-1620 v3 with 16 GB of memory and we use four cores for PTC and GMD computations.

Scenario	Analysed Pedestrians	Input Frames	Avg. time PTC	Avg. time GMD
Manko	42	373	0.034	3.72e-01
Marathon	18	450	0.041	0.98E-04
Explosion	19	238	0.033	3.12E-06
Street	147	9014	0.022	4.13E-08
TrainStation	200	999	0.061	5.11E-08
ATC Mall	50	7199	0.037	2.28E-01
IITF-1	18	450	0.041	3.11E-03
IITF-3	19	238	0.046	2.74E-04
IITF-5	18	450	0.056	2.98E-03
NPLC-1	19	238	0.039	1.31E-04
NPLC-3	18	450	0.031	4.16E-03
NDLS-2	19	238	0.049	3.00E-04
2017 Presidential Inauguration	130*	1927	0.87	0.38

Table 4.1: Performance of PTC (Personality Trait computation) and GMD (Global Movement Dynamics) algorithms on different crowd videos. We highlight the number of pedestrians used for personality classification, the number of video frames used for extracted trajectories, and the running time (in seconds). In the PBS Presidential Inauguration video, we chose around 130 representative pedestrians in the video for analysis and prediction.

4.4.2 User Evaluation

To evaluate the performance of our personality classification, we compare the results of PTC with a user study on the same set of videos and pedestrians. We present the details and highlight the results.

Study Goals and Expectations The aim of the study was to compare the results of our personality trait computation algorithm (PTC) to the perception of personality by human users for the given videos.

Experimental Design The participants of our web-based user study were recruited from students and staff in a university. Each participant was shown 10 different videos of pedestrians walking among crowds and asked to label them based on personality adjectives. At the beginning of each video, a specific pedestrian was highlighted by a yellow circle and, over the next few seconds, that pedestrian would navigate the crowded scene. Participants were given written instructions to carefully observe the highlighted pedestrian's trajectory and its interactions with the other pedestrians. They were advised not to take into account other factors

corresponding to facial expressions, visual appearance, etc. After observing the pedestrian trajectory in each video, participants reported the most appropriate personality trait from the given set: *Aggressive, Shy, Active, Tense, Impulsive, and Assertive*.

Results and Discussion Figure 4.4 shows the responses of 31 participants for 10 videos. Even though we model an individual's personality as a combination of six personality traits (6-D), evaluating a nominal variable with such a high number of categories is difficult. From the data, we observed that combining two personality traits to reduce the six factor model to a three factor model increases the agreement between participant responses. This three factor model is the **PEN** model (3-D) (explained in Section 3.3) and previous studies have shown that it can also offer sufficiently rich dimensions to characterize personality traits in crowd navigation. We therefore map the six personality traits to the 3-factor PEN model (Figure 4.5) using a linear mapping as described below.

Trait	Adjectives
Psychoticism	Aggressive, Impulsive
Extraversion	Assertive, Active
Neuroticism	Shy, Tense

Table 4.2: Correspondence between six personality traits and the PEN model (Pervin, 2003).

Some of the videos still show disagreement between the participants; we attribute this to the inherent features of the pedestrian in the videos and do not take these videos (Videos 2, 5, and 7) into account for further analysis.

Figure 4.6 shows the comparison of participants' responses to the personality traits predicted by our PTC algorithm. Over the 7 crowd videos, we observed an overall accuracy of 76.96% for the most dominant personality trait given by 31 participants. The accuracy increased to 88.48% if we also included the second most dominant personality trait. We also computed a statistical measure, Fleiss' kappa (κ), to assess the reliability of agreement between the participants. A value of $\kappa = 0.4578$ indicated a moderate agreement between the participants' responses based on (Landis and Koch, 1977). Error analysis of κ revealed that the observed agreement was not accidental ($Z = 35.6444, p < 0.0001$).

4.5 Conclusions, Limitations, and Future Work

We present a novel algorithm to classify the personalities of pedestrians in a crowd video. We compute the time-varying motion model of each pedestrian using Bayesian inference and combine it with Personality

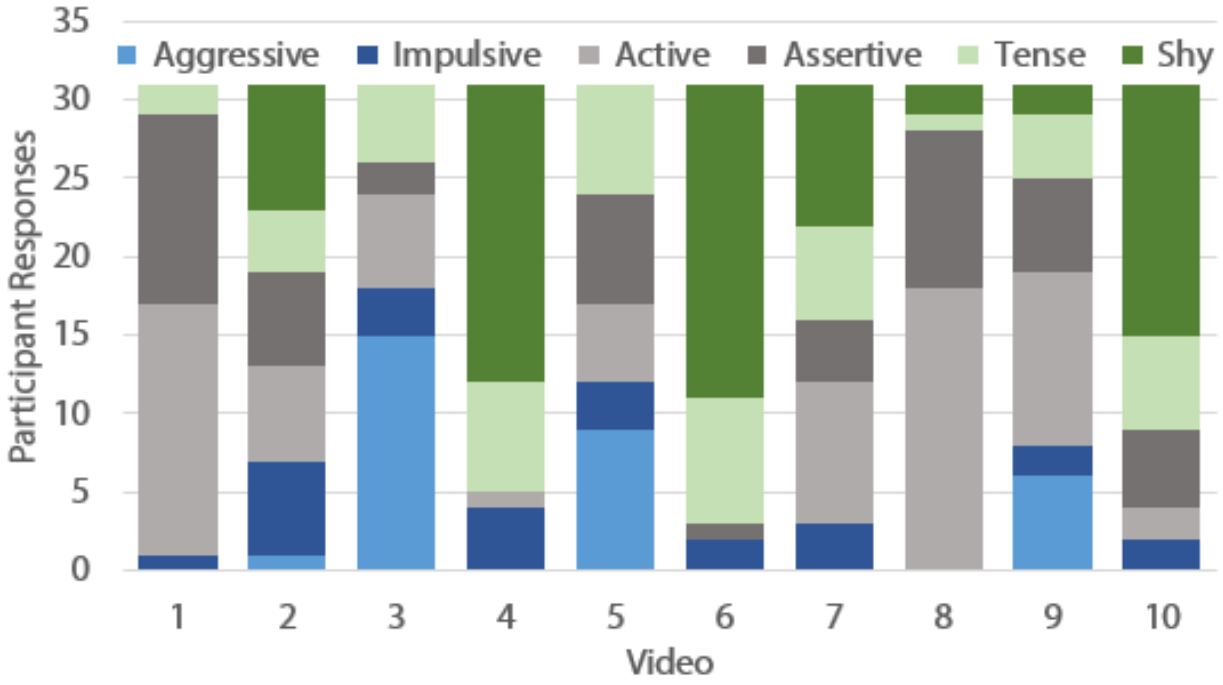


Figure 4.4: Participant Responses Using the Six Factor Personality Model: Participants were shown 10 different videos of pedestrians walking among crowds. In each video, a single pedestrian was highlighted and participants were asked to report the most appropriate personality trait for that pedestrian. 31 participants reported the most appropriate personality trait from the given set: *Aggressive, Impulsive, Active, Assertive, Tense, and Shy*.

Trait Theory. We also compute the global movement features and use them to analyze and simulate the crowd movements or distributions. We evaluate the accuracy with a user study and our results are promising. To the best of our knowledge, this is the first approach for automatic pedestrian personality classification based on their movements in a video.

Our approach has some limitations. The behavior classification is based on personality models and PEN, and may not be sufficient to capture all observed behaviors. We assume that it is possible to extract the trajectory of every pedestrian in a crowd. The global algorithm assumes that the relative distribution of pedestrian behaviors is about the same.

As part of future work, we would like to overcome these limitations and extend our algorithm for anomaly detection and surveillance applications. We would further like to adopt the same data-driven techniques to build mappings from simulation parameters to other personality trait theories, such as the OCEAN model. We would also like to investigate the extent to which our proposed model is appropriate for different cultures. Finally, we would like to use it for autonomous navigation of robots among crowded scenes.

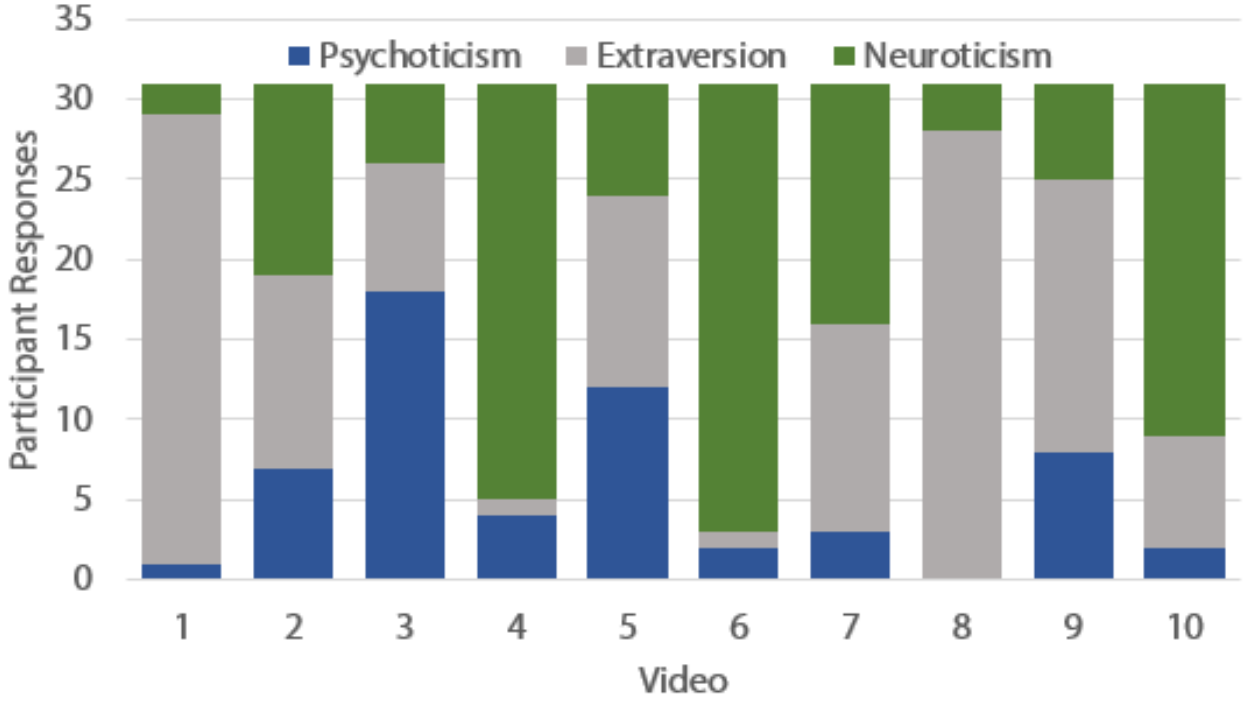


Figure 4.5: Participant Responses Using the PEN Model: We converted the participant responses to the three factor PEN model to reduce the disagreement. Participant responses were converted to three PEN factors (*Psychoticism, Extraversion, and Neuroticism*) using Table 4.2.

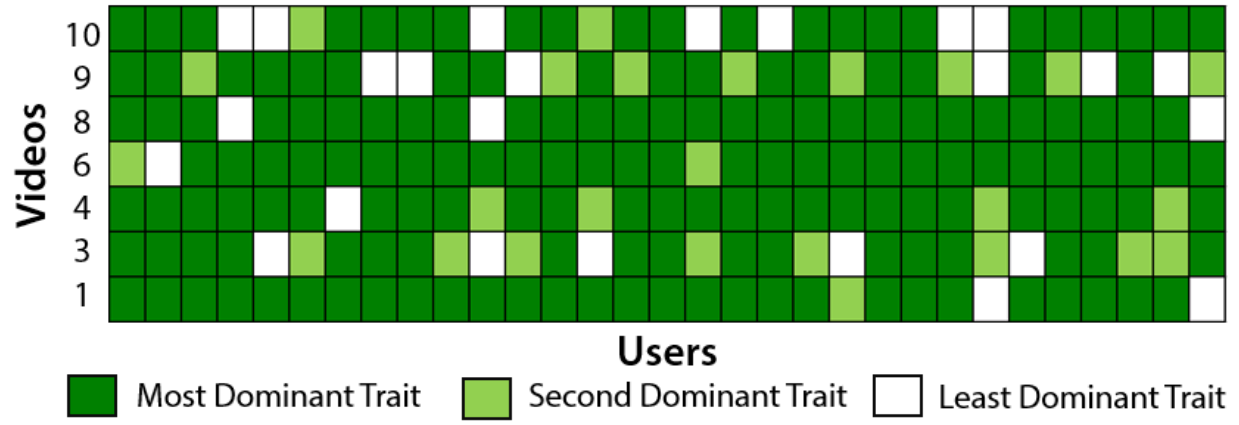


Figure 4.6: Accuracy of our PTC Algorithm: Our PTC algorithm predicted the most dominant trait for each of the 7 videos. In 76.96% of the cases, participants also chose the most dominant trait as denoted by the dark green color. If we also add the second most dominant trait (denoted by light green), the accuracy increased to 88.48%. This accuracy indicates that our PTC algorithm was able to correctly identify the personality traits as perceived by human participants.

CHAPTER 5
Applications

In this chapter, we demonstrate three applications from graphics, robotics and crowd surveillance based on our novel algorithms for tracking, prediction and behavior learning.

5.1 Data-driven Crowd Simulation

We present a new data-driven technique that can be used to generate crowd simulations based on real-world videos. Our method is built on top of an improved multi-person tracking algorithm. Furthermore, we integrate an online smoothing technique with our tracking algorithm so that we can directly use the trajectories for interactive applications (e.g. games, virtual environments) or animation.

Current techniques for automatic tracking are either limited to sparse crowds or are limited to offline simulation. Most of the online tracking algorithms use a probabilistic basis. Therefore, human-agent detection and the trajectories computed tend to be noisy, inaccurate and can have issues with agent orientations. This leads to incorrect or low-fidelity simulation or rendering of resulting agents in interactive applications. Even state-of-the-art multi-person trackers can result in lower accuracy in crowded scenes; seemingly small problems, such as occlusion or changes in illumination, can cause major issues in tracking, such as ID switches (when a tracker erroneously targets another pedestrian) or loss of the tracking target. Even though pedestrian tracking is well studied, there are many challenging issues that arise due to the following reasons: restricted visibility due to inter-pedestrian occlusion (one pedestrian blocking another), changes in lighting and pedestrian appearance, and the difficulty of modeling human behavior or the intent of each pedestrian.

We apply our tracking and parameter-learning algorithms for data-driven crowd simulation. We demonstrate their performance for crowd replication as well as a few applications.

We use our improved tracking algorithm for two applications: crowd replication and mixing crowd streams.

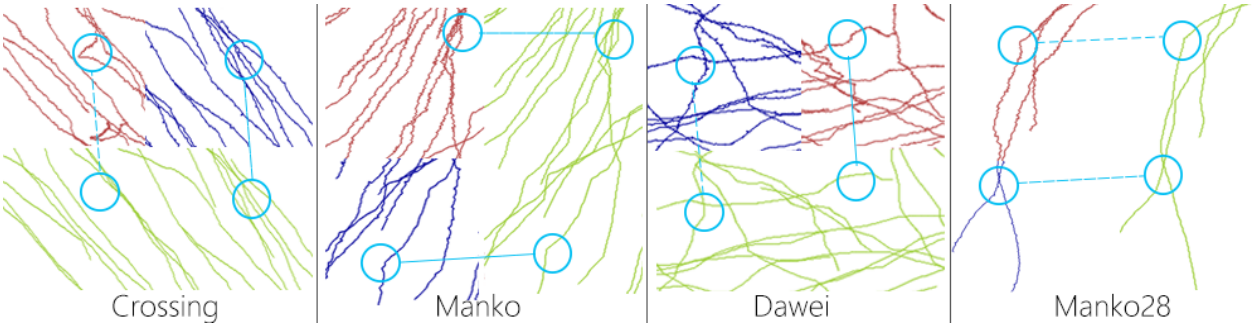


Figure 5.1: Noisy vs. Smooth trajectories: The red trajectories were tracked using LIN (constant velocity) as the motion model and the blue trajectories are results from our baseline tracking algorithm (Chapter 2). We display the improved trajectories extracted by our algorithm in green, which are smoother. Our motion model iteratively refines pedestrian behavior and produces smooth trajectories. The blue circles highlight the improvement using our method. (For clarity, these trajectories are just a cropped section of the entire scene.)

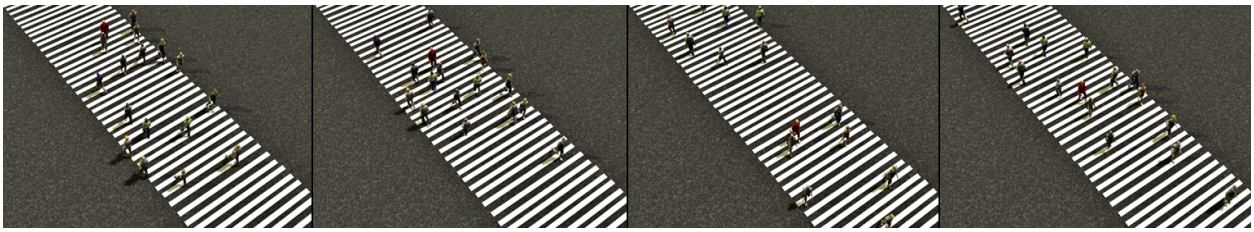
(a) Background extracted (b) Pedestrians extracted (c) Virtual+Real Crowd (d) Final Result

Figure 5.2: **Augmented Crowd Video:** Our approach can be used to augment a crowd video with additional virtual agents. (a) We use an outdoor crowd scene (Crossing benchmark (Shao et al., 2014)) as a background. (b) We extract pedestrian trajectories from a different video benchmark (Manko benchmark (Shao et al., 2014)) (c) We add computer-simulated virtual pedestrians following behaviors observed in the Manko video dataset. (d) The resulting trajectories are rendered and overlaid upon the Crossing video dataset. We use chroma keying to insert these new virtual agents into the original video, which includes additional environmental features, such as moving cars.

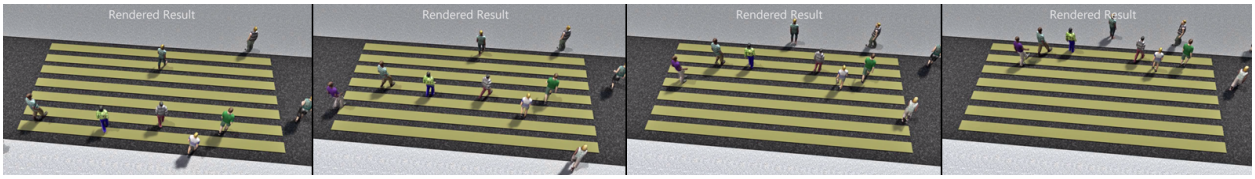
5.1.0.1 Crowd Replication

One important application of data-driven crowd simulation is crowd replication. The goal is to faithfully reproduce real human crowd trajectories or movements using virtual characters in a simulation. Crowd replication is used extensively in movies and games, but in almost all cases, the pedestrian trajectories are either drawn manually or the tracked data is subject to extensive post-production work to make it usable.

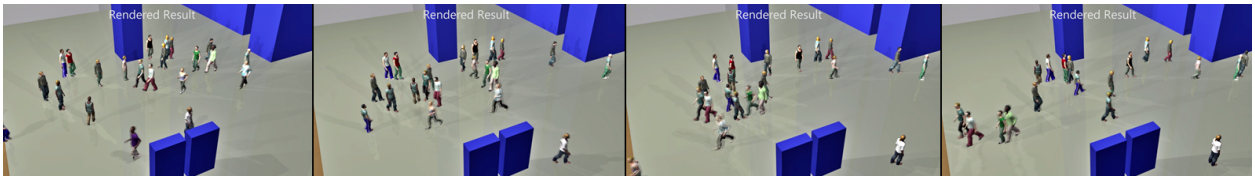
Some of the problems which beset pedestrian-tracking algorithms include trajectory noise, incorrect pedestrian orientation (pedestrian orientation is computed using past states but due to noise, this computation can be incorrect leading to frequent orientation changes), and ID switches; our algorithm produces smooth trajectories (which also resolves orientation issues), and displays fewer ID switches and higher accuracy overall.



(a) Crossing



(b) Manko28



(c) Dawei

(d) Manko

Figure 5.3: **Replicated Crowds.** We improve the quality of the rendered crowd behaviors by adding online smoothing step to the tracker. (a) to (d) shows the replicated crowds rendered directly using the tracking results without any pre-processing, corresponding to each benchmark video.

We feed the smooth trajectories from our pipeline to *Golaem*, a commercially available crowd-rendering platform.

5.1.0.2 Mixing Crowd Streams

Another application is to combine the pedestrian trajectories extracted from two or more different videos. Each video may only capture some aspects of crowd behavior, and we want to combine them for an application. In this case, our mixing algorithm takes as input the smooth trajectories extracted by our algorithm from two videos (those shown on the left and middle figures in Fig. 5.4). The mixing algorithm (Zhang et al., 2010) uses the two sets of extracted trajectories and performs a simple local collision avoidance between them.

5.1.1 Crowd Content Generation

Generating crowd behaviors for multimedia content has received considerable attention in many fields, including animation, gaming, virtual reality, education, computer-aided design of architectural structures, etc. In this section, we use the trajectories of real-world agents and their behavior parameters to generate or mix visual crowd content. This process involves augmenting real pedestrians in a video using virtual characters or agents that have similar behaviors and that do not collide with one other agents or with obstacles. We demonstrate that our approach can generate crowd movements or trajectories that are similar to those observed for pedestrians in the original video.

Traditionally, crowd behavior is generated by designers or animators using editing tools (e.g., by drawing the trajectory of each agent on a 2D screen). In our case, we use widely available crowd video datasets to extract the trajectory-level behavior of characters and automatically insert them into other crowd videos.

5.1.1.1 Augmented Crowds

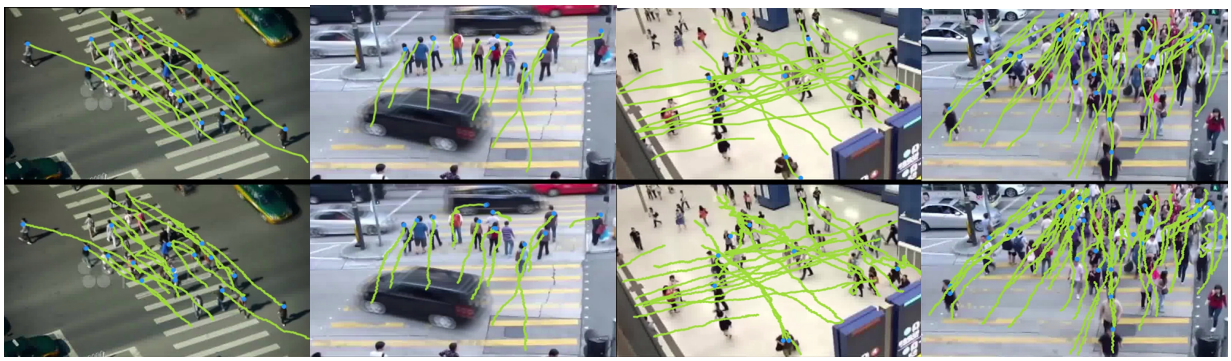
Our main goal is to add more agents into a real-world crowd video to increase the number or density of pedestrians or agents engaged in other behaviors or movement. Different methods have been proposed to insert virtual crowds as an overlay onto a real-world video in computer graphics and augmented reality. These methods include the coupling of camera-tracked humans with virtual agents (Rivalcoba et al., 2014) based on a precomputed human detector. Grid-based density fields have been used to overlay several simulated virtual agents onto a real-world video (Ren et al., 2013) that is based on a generalized steering model utilizing linear

Figure 5.4: Mixing Crowd Streams: The agents on the brown (left) and blue (central) floors exhibit varied behaviors, generated from different videos. The final mixed video (green floor) has behaviors combined from both the video streams. Pedestrians marked with brown are from the left video, and those marked blue are from the central video. The overall mixing algorithm uses the two sets of extracted trajectories and performs local collision avoidance between them.

(a) Pedestrian trajectories extracted from two different videos

(b) Mixed trajectories

Figure 5.5: **Mixing trajectories from multiple input videos.** We can use multiple videos as inputs to a single, more complex scenario. In the left two videos of (a), pedestrians move in an image-space from bottom to top or from top to bottom. In the right two videos of (a), pedestrians move in a uniform direction: right to left in a slightly tilted way in the image-space. There are, therefore, three different main directions of pedestrian movements. (b) Using these trajectories with agent-based simulation methods, we can generate crowds with these three different flows, while still achieving local collision avoidance between the agents.



(a) Crossing (b) Manko28 (c) Dawei (d) Manko

Figure 5.6: **Comparison of improved tracking (our method, above) to prior methods based on particle filter + LIN (below).** We compare the quality of the extracted trajectories on four different real crowd videos. As compared to prior methods, our approach results in smoother trajectories and improved accuracy for each benchmark. Our method runs at interactive rates (24-26fps)

(a) (b) (c)

Figure 5.7: **Augmented Crowd** We use pedestrian tracking results for crowd content generation. In this scenario, we add virtual pedestrians to the scene and generate collision-free trajectories for them at interactive rates. (a) Trajectories of 23 real pedestrians in an open space; (b) adding 50 virtual pedestrians; (c) adding 300 virtual pedestrians;

velocity prediction. Other methods simulate virtual agents automatically by adjusting the behavior of such virtual agents to mimic pre-learned behaviors (Pellegrini et al., 2012).

The trajectories and behaviors extracted from a real-world video can be used to create augmented crowds containing rendered real pedestrians as well as computer-generated virtual pedestrians. At each time step, our tracker computes a position for each pedestrian in the video, denoted as $\mathbf{z}_t \in \mathbb{R}^2$. The first step is to compute appropriate trajectories for virtual pedestrians based on the trajectory behavior features learned from the input video. We use the behavior features extracted from real pedestrians to generate salient movement patterns by grouping them based on their similarities to one another. Finally, we use this information to steer virtual agents' short-term behavior. Overall, we compute the combined state of real and virtual pedestrians, $\mathbf{X}$, during each time step as follows:

$$\mathbf{X} = \mathbf{X}_R \cup \mathbf{X}_V, \tag{5.1}$$

where $\mathbf{X}_R$ is the real crowd state and $\mathbf{X}_V$ is the virtual crowd state. These states are used to generate the final augmented crowd content.

Crossing Scenario: One scenario with which we can highlight the benefit of our approach is pedestrian crossing. In this case, the goal is to use the same background environment and traffic vehicles and augment these with different types of pedestrian behaviors or densities in the crossing video (See Fig. 5.2(a)). In this case, we extract trajectory-level behavior characteristics from the Manko benchmark video (See Fig. 5.2(b)) and generate virtual pedestrians with similar characteristics. The final video contains both real pedestrians

(from the original Crossing video) and virtual agents (from the Manko video) and demonstrates that our approach can easily *mix* the pedestrian behaviors from two or more real-world videos.

Note that pedestrians in the final video are rendered in a 3D environment. The camera parameters are often provided with the video dataset to compute the transformation from image space coordinates to world-space coordinates. Otherwise, the parameters can be estimated with one frame of the input video. The advantage of rendering the crowds in 3D is that we can visualize or evaluate the results for scenes with different viewpoints. After rendering the pedestrians and their trajectories using a crowd rendering system, we use chroma keying – a well-known method in video and multimedia editing – to remove the background. We color correct and add motion and camera effects to the virtual crowd before overlaying it on top of the real video (after masking the real people to remove them from the video).

Crowds in 3D Virtual Environment We can easily insert our mixed pedestrians into a virtual 3D scene. Visual effects artists can create their own 3D scenes in any rendering/modeling software and easily import our mixed trajectories to render crowds.

(a) (b) (c)

Figure 5.8: **Augmented Crowd Rendered in a 3D Environment** (a) Original Dawei video dataset with tracked trajectories, from which 27 trajectories are extracted; (b) Rendered scene with real and randomly inserted virtual pedestrians with similar trajectory-level behaviors, in which 100 virtual pedestrians have been added; and (c) Rendered scene with real and virtual agents using our behavior-learning approach.

Dawei: In this benchmark, we use the trajectory-level behavior feature of real pedestrians to generate 3D crowd animation content. By augmenting real-world trajectories with virtual crowds or agents, we can generate different scenarios (see Fig. 5.8). (a) We used 27 real pedestrian trajectories extracted from the Dawei video. (b) The real pedestrians are rendered by using the extracted trajectories and are augmented with virtual agents simulated with uniformly sampled initial and goal positions around the scene boundaries. (c) The intermediate goal positions are selected based on the behavior features learned by our system.

Square: We used trajectories of 23 pedestrians from the original video of a crowd walking in an open space (see Figure 5.7 (a)). During each frame, we estimate their state and compute the pedestrian behavior

feature. We perform mixed crowd simulation by adding virtual pedestrians and increasing their number (see Figure 5.7 (b)-(c)). Whereas the original video is relatively sparse, the crowd density in the mixed simulation with 300 virtual pedestrians becomes about 4.5peds$/m^2$

5.2 SocioSense: Socially-Aware Robot Navigation

Robots are increasingly used in households, offices and public places and frequently navigate amongst humans or pedestrians. As humans are dynamic agents, these scenarios result in many new challenges related to *human-aware navigation and interaction*. Robots must move through crowds of people while preventing collisions with each other and with humans. In such scenarios, the robots need to interface with not only the physical environment, but also the social environment, and should interact well with humans.

People have clear social norms about interpersonal space or acceptable behaviors (Burgoon and Jones, 1976). A robot that impinges on someone's personal space could make them feel uncomfortable (Takayama and Pantofaru, 2009). There is considerable work on socially-aware navigation in motion planning and HRI (human-robot interaction) literature. These techniques tend to predict the movement or actions of human pedestrians and use them to develop appropriate navigation algorithms (Kruse et al., 2013; Okal and Arras, 2016; Ferrer et al., 2013; Kuderer et al., 2012). The resulting paths account for social norms and conventions, such as personal space and yielding the right of way.

We address the problem of a robot navigating through low- and medium-density crowds. In addition to social constraints, we also take into account the personality or time-varying behaviors of different pedestrians. It is known in psychology that even a simple task such as walking towards a destination involves several complex human-centric decisions, such as which route to take and the various ways to avoid collision with the robot, other pedestrians, and the obstacles in the environment. In many scenarios, different pedestrians will accomplish the same goal in different ways and their resulting paths are governed by their underlying personality (e.g. aggressive, shy, or active). Therefore, it is important to capture the pedestrians' personality traits, and use them to predict their future positions for socially-aware navigation. In many dense crowd scenarios, the speed and movement pattern of a pedestrian may change in response to environmental factors, such as crowd density and overall flow. A key issue is therefore modeling variations in behavior that arise as humans navigate in the physical world and avoid collisions. Furthermore, it is important to capture and compute these time-varying behaviors at interactive rates, so that the robot can navigate accordingly.

We present a real-time planning algorithm, *SocioSense*, that takes into account psychological constraints of each pedestrian in the environment to perform socially-aware navigation. We extract the trajectory of each pedestrian from the video and use Bayesian learning algorithms to compute his/her motion model and the personality characteristics of the pedestrian. Our behavior classification is based on Personality Trait Theory

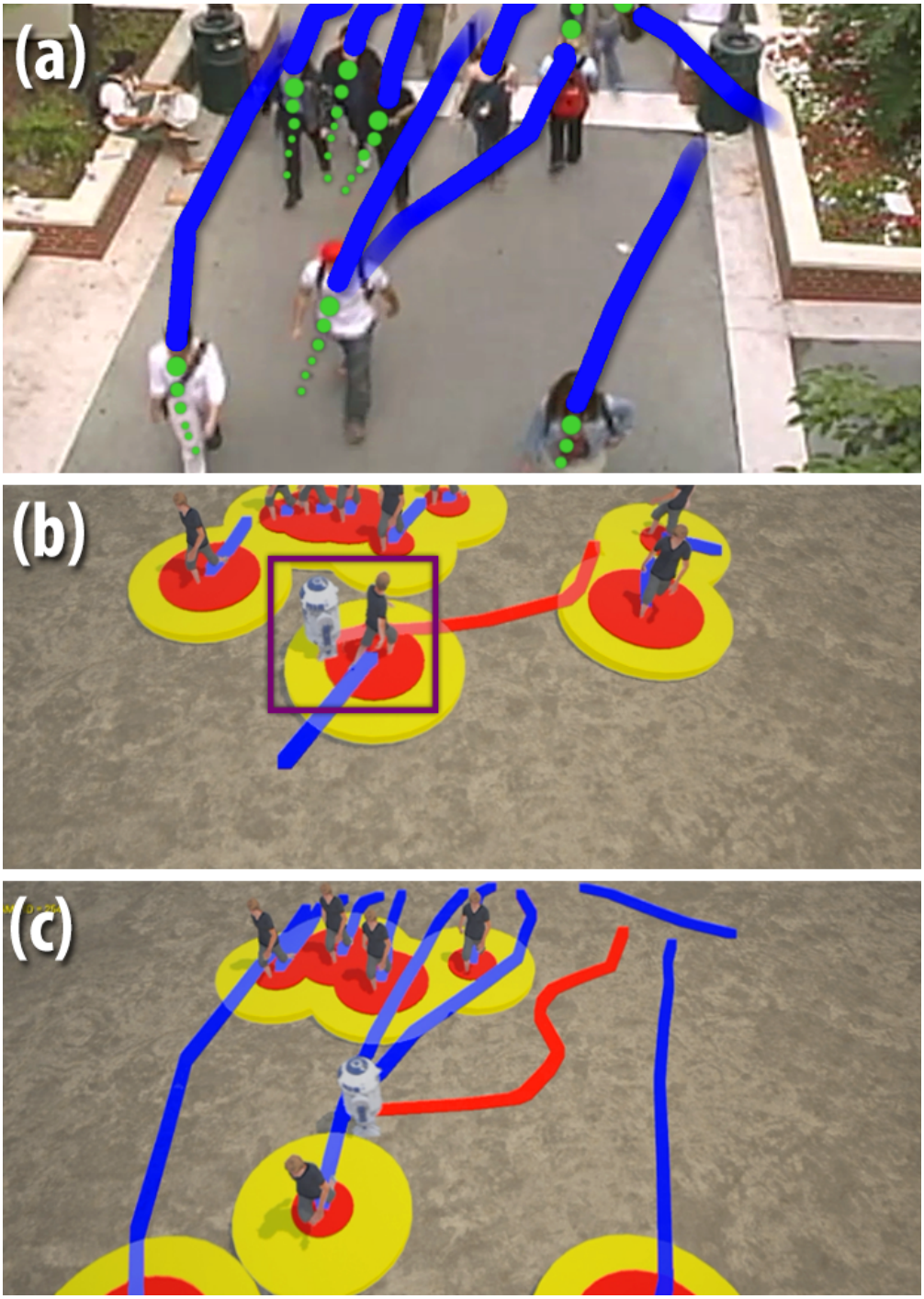


Figure 5.9: Improved Navigation using **SocioSense:** (a) shows a real-world crowd video and the extracted pedestrian trajectories in blue. The green markers are the predicted positions of each pedestrian computed by our algorithm that are used for collision-free navigation. The red and yellow circles around each pedestrian in (b) and (c) highlight their personal and social spaces respectively, computed using their personality traits. We highlight the benefits of our navigation algorithm that accounts for psychological and social constraints in (c) vs. an algorithm that does not account for those constraints in (b). The red trajectory of the white robot maintains these interpersonal spaces in (c), while the robot navigates close to the pedestrians in (b) and violates the social norms.

in psychology literature. This theory assumes that the variations in behavior are governed by a small number of underlying traits.

We combine the time-varying behavior classifications for each pedestrian with local and global learning methods to perform long-term path prediction for collision-free, socially normative robot navigation. A key contribution of this work is mapping the learned personalities to a set of social distances based on proxemics and interpersonal distances (Hall, 1966). These distances constrain the robot navigation to avoid passing through people's personal and social spaces. Using the combination of psychological and social constraints improves both the prediction and the navigation. We have evaluated the performance of our algorithm in real-world captured videos consisting of tens of pedestrians, including dense scenarios

5.2.1 Psychological Constraints

In this section, we introduce the notation and present our approach to model psychological and social constraints. In human environments, it is important to compute robot trajectories that follow the social norms about interpersonal distance and acceptable behaviors. Most of the earlier work is limited to collision-free trajectory planning or only takes into account physical constraints such as kinematics and dynamics. The psychological constraints are complimentary to these physical constraints and depend on the personality or behavior of a pedestrian in the crowd.

Fig. 5.9 gives an overview of our approach, including computation of the different constraints and their use in navigation. Our real-time algorithm learns pedestrians' psychological and social cues from trajectories extracted from real-world crowd videos. Our approach is generic and can be combined with almost any real-time pedestrian tracker that works well on low- to medium- density crowds. We learn a pedestrian's personality traits (psychological constraints) from the trajectories using Bayesian learning. We use these time-varying psychological constraints to compute appropriate interpersonal distances for each pedestrian for robot navigation (social constraints). The personality classification is also used for long-term path prediction, which is also used by the socially-aware robot navigation algorithm.

5.2.2 Personality Traits and Psychological Cues

A key issue in our approach is classifying the personality of each pedestrian in the crowd. We first obtain the motion model parameters from our algorithm in chapter 2 and then e use our behavior learning algorithm from chapter 4 to determine the values of the personality traits, $\mathbf{b} = \{Aggressive, Assertive, Shy, Active, Tense, Impulsi$

The only difference is that we weigh, all the personalities in the matrix $\mathbf{b}$ with $\{\mathbf{w1}$, $\mathbf{w2}$, $\mathbf{w3}$, $\mathbf{w4}$, $\mathbf{w5}$ and $\mathbf{w6}\}$. We use the default $\mathbf{w}$ values ($w_i = 1$) for this step. Even though our approach is general, the mapping ($\mathbf{RVO_{mat}}$) is specific to the RVO motion model and the user study described in (Guy et al., 2011). However, we can use different mappings or other forms of regression to compute such a mapping between the personality characteristics and the motion model. Next, we use these personality characteristics to compute proxemic distances for socially-aware navigation.

5.2.3 Pedestrian Path Prediction

The key aspect of any real-time prediction algorithm is estimating the motion parameters of a pedestrian as they provide the best estimator of their movement in a dense setting. We represent the motion parameters of a pedestrian i by his/her state $\mathbf{x_i}$. Given the state of the crowd $\mathbf{X} = \bigcup_i \mathbf{x_i}$ for the previous n frames at time t, our pedestrian path prediction algorithm predicts the motion parameters of a pedestrian i for a future time, $\mathbf{x_i^{t+\Delta t}}$. We compute these motion parameters from the personality trait learning module described above in Section 5.2.2. We use these estimated motion parameters to extend the accuracy of the real-time prediction algorithm GLMP 3.

Pedestrian behavior may have slight variations during the course of the motion. To capture these variations, we find an upper bound $\mathbf{M}_{ub}$ and a lower bound $\mathbf{M}_{lb}$ on the motion parameters. From the computed values of six personality traits ($\mathbf{b}$) in Section 5.2.2, we compute the trait with the largest value, the most dominant trait, b_d. $\mathbf{M}_{ub}$ is calculated by adding $y\%$ to b_d and adding $(y/3)\%$ to the other traits ($\mathbf{b} \setminus b_d$). Similarly, we subtract $y\%$ and $(y/3)\%$ to compute $\mathbf{M}_{lb}$, where y is a user-defined variable. A value of y = 5% is able to capture noise and the natural variance in the pedestrian behavior.

If the individual pedestrian parameters are within the bounds $\mathbf{M}_{lb}$ and $\mathbf{M}_{ub}$, we use them directly for prediction; otherwise, we clamp $\mathbf{M}$ to the corresponding boundary value of the motion parameters, ($\mathbf{M}_{lb}$, $\mathbf{M}_{ub}$). Having updated the motion model parameters, we recompute the personality weights $\mathbf{w}$. Our formulation assumes that each pedestrian behavior generally remains constant and lies within a range of variance; if the change in pedestrian behavior is substantial between successive frames, there is a possibility of error in prediction. Since the behaviour of a pedestrian might changes considerably over a long period of time, we re-sample the behavior every few frames.

$$\mathbf{M} = \begin{cases} \mathbf{M}_{lb}, & \text{if } \mathbf{M_i} \leq \mathbf{M_{lb_i}}, \forall i \\ \mathbf{M_{ub}}, & \text{if } \mathbf{M_i} \geq \mathbf{M_{ub_i}}, \forall i \\ \mathbf{M}, & \text{otherwise} \end{cases}$$

Next, we combine these motion parameters $\mathbf{M}$ with the local movement patterns from to perform long-term path prediction for each pedestrian.

5.2.4 Proxemic Distances (Social Cues)

A key component of socially-aware navigation is computation of the appropriate distance between the robot and each pedestrian, based on the notion of proxemics (Hall, 1966). In this context, we use the notions of *public distance* and *social distance* to perform socially-aware navigation. In particular, public distance refers to the distance at which people can give a speech and social distance characterizes the distance at which people can talk to each other. These distances have been known to vary according to cultural norms, environment, or an individual's personality. In our formulation, we mainly focus on variations in these distances originating from the differences in personality. It has been shown that traits like *Extraversion* affect interpersonal distances. Though social and public distances do not vary significantly, extroverts can have a smaller personal distance than introverts (Williams, 1971). Based on the formulation described in (Williams, 1971), we compute the limits on an individual's personal distance (Table 5.1). We obtain the personal distance of an individual by taking a weighted average of these limits, weighted by a computed value of *Extraversion* (PEN_e) for that individual.

	Personal Distance	**Social Distance**
Extrovert	179.58	267.97
Introvert	88.9	233.17

Table 5.1: **Extraversion vs Personal/Social Distances:** The personal distance indicates the minimum distance before the pedestrian feels uncomfortable with the robot. All distances are given in cms.

In order to determine the *Extraversion* of each pedestrian, we compute the mapping of the six personality traits ($\mathbf{b}$, computed earlier) to a 3-factor **PEN** model based on the function described below:

$$\begin{pmatrix} \mathbf{P}sychoticism \\ \mathbf{E}xtraversion \\ \mathbf{N}euroticism \end{pmatrix} = \mathbf{PEN_{mat}} * \mathbf{B}$$

where,

$$\mathbf{PEN_{mat}} = \begin{pmatrix} 0.22 & 0.28 & -0.09 & -0.01 & -0.17 & 0.31 \\ 0.16 & 0.33 & 0.07 & 0.53 & 0.05 & 0.10 \\ -0.15 & 0.16 & 0.47 & -0.01 & 0.42 & -0.08 \end{pmatrix}.$$

Currently, we use a single scalar quantity, *Extraversion* (PEN_e), from the PEN model. Based on the proposed distances given in Table 5.1, we establish a linear mapping between the pedestrians' personality traits and their personal distance to the robot. After normalizing PEN_e by dividing it by the magnitude of the **PEN** vector, we compute the personal distance d_p of the pedestrian:

$$d_p = 179.58(PEN_e) + 88.9(1 - PEN_e)$$

For the other distance metrics,, like social distance d_s, we select a distance from Table 5.1. If $PEN_e < 0.5$ then $d_s = 233.17$ else $d_s = 267.97$.

5.2.5 Socially-Aware Robot Navigation

We use the distances, d_s and d_p, computed using the psychological constraints to enable socially-aware collision-free robot navigation through a crowd of pedestrians. Our navigation method is based on Generalized Velocity Obstacles (GVO) (Wilkie et al., 2009), which uses a combination of local and global methods. The global metric is based on a roadmap of the environment. The local method computes a new velocity for the robot and takes these distances into account. Moreover, we also take into account the dynamic constraints of the robot in this formulation.

Figure 5.11 illustrates how a robot avoids an approaching pedestrian based on these distances. At a given time instant, the pedestrian is located at $\mathbf{p}_{human}^{curr}$, and has two proxemic distances: a personal distance of d_p (red) and a social distance of d_p (yellow). At the same time instance, the robot is located at $\mathbf{p}_{robot}^{curr}$ and has a preferred velocity $\mathbf{v}_{robot}^{pref}$ that is computed based on global navigation module. This is the velocity that it would have for navigating to its goal position, in the absence of any static or dynamic obstacles. The robot

predicts that during the next time frame, the pedestrian will move to the position $\mathbf{p}_{human}^{pred}$ using the path prediction described in Section III(D), and computes its new velocity to avoid a collision with the pedestrian. However this path prediction is not sufficient for socially-aware navigation, since the robot fails to take into account the pedestrian's proxemic distances. Based on these distances, the robot alters its goal position to $\mathbf{p}_{robot}^{pred+soc}$ and its velocity to $\mathbf{v}_{robot}^{pred+soc}$ to accommodate both social and psychological constraints. Notice that the velocity $\mathbf{v}_{robot}^{pred}$ causes the robot to intrude on the pedestrian's personal distance, shown by the red circle centered around the pedestrian, whereas the updated velocity $\mathbf{p}_{robot}^{pred+soc}$ successfully accounts for the pedestrian's personal distance and as well as its social distance.

We assume that the pedestrians in the environment are non-cooperative and may not actively avoid collisions in a reciprocal manner with the robot. Thus, the robot must assume 100% responsibility to avoid collisions and keep safe distances. During the navigation, our *SocioSense* algorithm predicts the pedestrian's future motion and avoids any steering inputs that result in a collision with the predicted pedestrian positions (Figure 5.10). Our approach is agnostic to the underlying navigation algorithm (e.g. GVO) and can be combined with other methods like potential field methods.

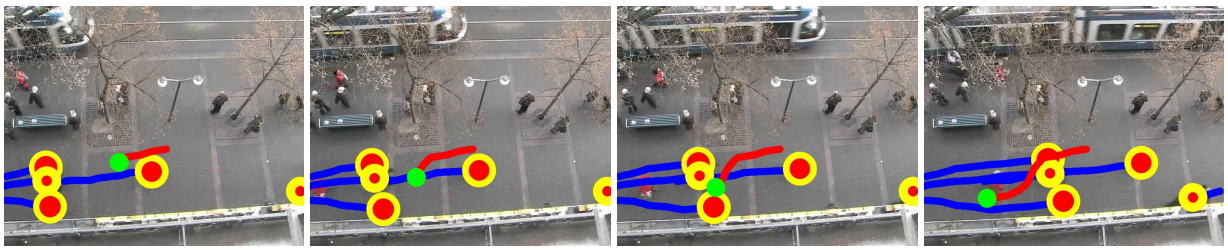


Figure 5.10: **Example robot trajectory** navigating through the crowd in **Hotel** dataset. Red/yellow circles represent current pedestrian positions(personal/social distance), green circles are the current position of the robot.

5.2.6 Performance and Analysis

We have evaluated the accuracy of our real-time prediction algorithm against other state of the art real-time methods. We applied our algorithm to the 2D pedestrian trajectories tracked from different video datasets, and calculated the accuracy of our predicted positions relative to the ground truth using an error metric. Because of error accumulation over time, trajectory prediction algorithms tend to be more accurate over shorter time windows as compared to longer time windows. Therefore, we measure the accuracy results for two time windows for each algorithm - short (1 second) and long (5 seconds). In terms of video

benchmarks, we chose different crowd video datasets corresponding to indoor and outdoor scenes with varying pedestrian density and cultural background. The density varied considerably: from low (less than 1 pedestrians/m^2) to medium (1-2 pedestrians/m^2) to high (more than 2 pedestrians/m^2). Table II summarizes the crowd video datasets, their crowd characteristics, and the accuracies of the predicted trajectories computed by different real-time algorithms for both time windows. The last columns highlight the improved prediction results computed using our approach.

Dataset	**Challenges**	**Density**	**# Tracked**	**ConstVelocity**		**Kalman Filter**		**GLMP (Baseline)**		**SocioSense**	
				1 sec	*5 sec*	*1 sec*	*5 sec*	*1 sec*	*5 sec*	*1 sec*	*5 sec*
UCSD-Peds1	BV, PO, IC	Low	43	75.2%	54.1%	76.4%	56.4%	79.1%	61.6%	84.3%	74.3%
Marathon	BV, PO, IC, CO	High	76	32.1%	10.4%	35.2%	11.3%	41.3%	17.2%	46.8%	21.1%
NDLS-2	BV, PO, IC	High	121	57.8%	31.0%	52.0%	34.1%	59.7%	39.6%	68.2%	43.0%
MANKO	BV, PO, IC, CO	High	87	43.7%	19.4%	44.4%	19.7%	49.4%	24.0%	59.2%	28.9%
879-38	BV, PO, IC, CO	High	43	38.1%	29.1%	38.7%	31.0%	41.2%	35.6%	48.1%	42.9%
Crossing	BV, PO, IC	High	37	75.3%	52.0%	76.1%	53.1%	79.0%	59.3%	86.6%	59.3%
IITF-1	BV, PO, IC, CO	High	167	63.5%	33.4%	63.9%	39.1%	65.3%	41.8%	65.3%	41.8%
IITF-3	BV, PO, IC, CO	High	189	61.1%	29.1%	63.6%	31.0%	67.6%	37.5%	78.2%	42.9%
IITF-5	BV, PO, IC, CO	High	71	59.2%	28.8%	61.7%	29.1%	62.9%	30.1%	68.1%	34.0%
NPLC-1	BV, PO, IC	Medium	79	76.1%	63.9%	78.2%	65.8%	79.9%	69.0%	81.2%	72.6%
NPLC-3	BV, PO, IC, CO	Medium	144	77.9%	70.1%	79.1%	71.9%	80.8%	74.4%	88.1%	74.6%
Students	BV, IC, PO	Medium	65	65.0%	58.2%	66.9%	61.0%	69.1%	63.6%	69.1%	63.6%
Campus	BV, IC, PO	Medium	78	62.4%	57.1%	63.5%	59.0%	66.4%	59.1%	66.4%	59.1%
seq_hotel	IC, PO	Low	390	74.7%	67.8%	76.7%	68.3%	76.9%	69.2%	81.2%	73.6%
Street	IC, PO	Low	34	78.1%	70.9%	78.9%	71.0%	81.4%	71.2%	81.4%	71.2%

Table 5.2: Accuracy Benchmarks: We compare our path prediction algorithm with state of the art real-time algorithms, on crowd video datasets with varying densities and numbers of tracked pedestrians, and time windows of 1 sec and 5 sec. Our approach, SocioSense, consistently outperforms the other methods, even for challenging datasets like Marathon. Abbreviations used for scene characteristics: BV: Background Variations, PO: Partial Occlusion, CO: Complete Occlusion, IC: Illumination Changes.

5.2.7 Prediction Accuracy

We use a simple metric to evaluate the accuracy of our predicted trajectories. The average human stride length is about 0.8 meters (Reynolds, 1987). For a given time instant, the predicted pedestrian position is counted as successful when the estimated mean error between the predicted position and the ground truth value is less than this constant. We define prediction accuracy at a time instant as the ratio of the number of "successful" predictions and the total number of tracked pedestrians in the video.

A prediction algorithm should be able to predict trajectories over a long time horizon. All the algorithms listed in Table II have reduced accuracy scores when a longer time window is used. Even in these situation, our approach outperforms (or does as well as) the other methods in these crowd video datasets. In particular,

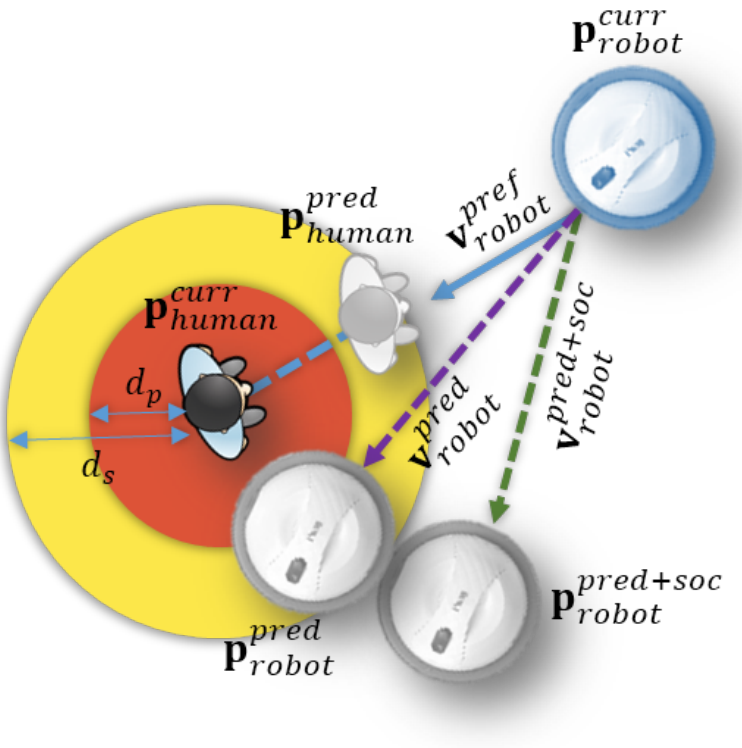


Figure 5.11: Our robot navigation algorithm satisfies the proxemic distance constraints, including personal space (red) and social space (yellow). The trajetory computed by our SocioSense navigation algorithm (green trajectory) does not intrude on the personal/social space of the pedestrian, whereas a robot that fails to take into account the social constraints (purple trajectory) may cause discomfort to the pedestrians.

our algorithm is significantly more accurate than competing real-time methods in predicting trajectories over a *long time window* in *high-density* crowd datasets like *Marathon* and *IITF-5*.

5.2.8 Socially-aware Navigation

We evaluate the performance of our socially-aware navigation algorithm, *SocioSense* with other algorithms without that do not take into account proxemic or social constraints. We compute the number of times the non-social robot intrudes on the personal space of the pedestrians, and thereby results in discomfort for some of the pedestrians. We also measure the additional time a robot with our *SocioSense* algorithm takes to reach the goal position, without any intrusions of pedestrians' personal/social spaces. Our results (Table 5.3) demonstrate that in $< 30\%$ additional time *SocioSense* can reach its goal while ensuring that the personal/social space of any pedestrian is not intruded. Table 5.3 also lists the time taken to compute proxemic social constraints.

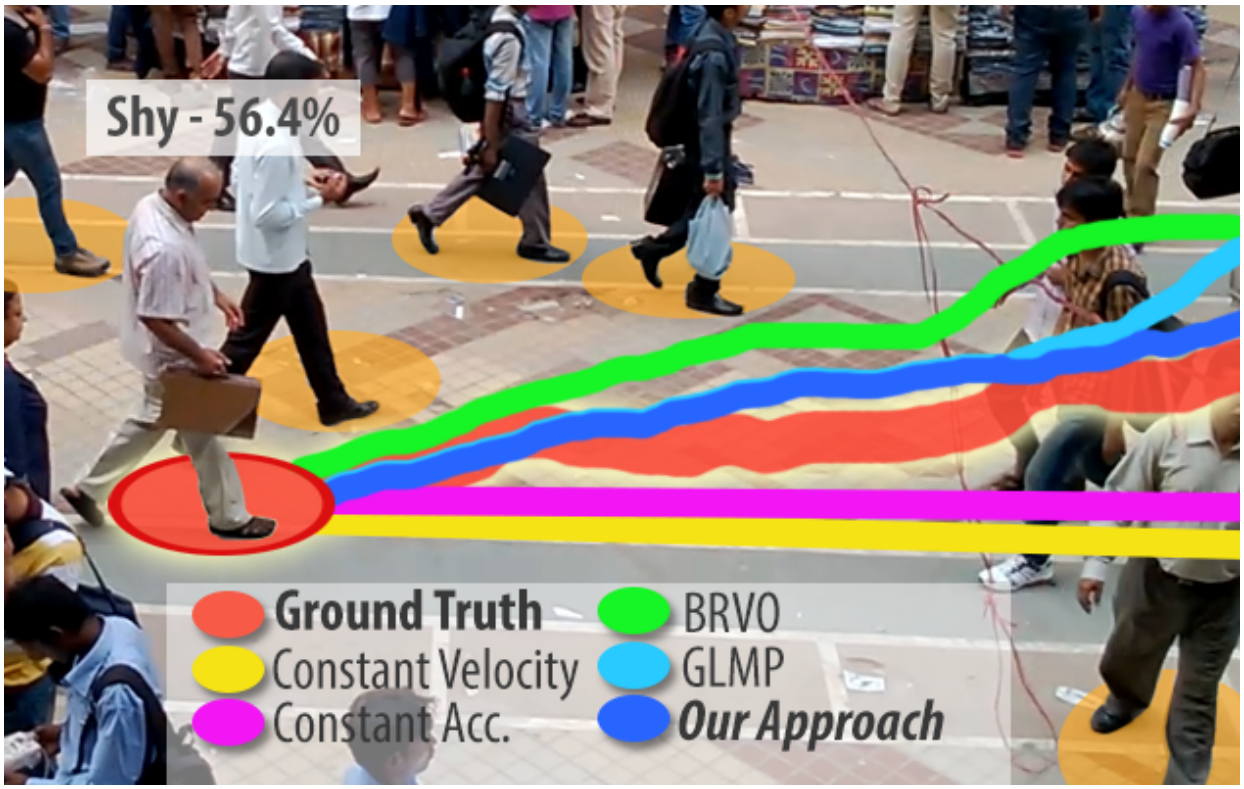


Figure 5.12: Our approach automatically classifies pedestrian behaviors in real-time (e.g. Shy behavior of a pedestrian). This behavior information is used to dynamically compute motion parameters and improve the performance of our long-term prediction algorithm (shown in dark blue) and compute proxemic distances. Our results are very close to ground truth (shown in red) and offer up to 21% improvement over prior real-time algorithms, whose predicted trajectories are shown in different colors. This demonstrates the benefits of using the psychological constraints for prediction and navigation.

Dataset	***Additional Time***	***Performance***	***Intrusions Avoided***
UCSD-Peds1	27%	3.00E-04 ms	11
NDLS-2	13%	2.74E-04 ms	24
Students	11%	0.72E-04 ms	17
seq_hotel	17%	0.98E-04 ms	31

Table 5.3: **Navigation Performance:** A robot using our *SocioSense* navigation algorithm can reach its goal position, while ensuring that the personal/social space of any pedestrian is not intruded with $< 30\%$ overhead. We evaluated this performance in a simulated environment, though the pedestrian trajectories were extracted from the original video.

5.3 Anomaly Detection

There has been a growing interest in developing computational methodologies for simulating and analyzing the movements and behaviors of crowds in real-world videos. This include simulation of large crowds composed of a large number of pedestrians or agents, moving in a shared space, and interacting with each other. Some of the driving applications include surveillance, training systems, robotics, navigation, computer games, and urban planning.

In this work, we deal with the problem of interactive anomaly detection in crowd videos and develop approaches that perform no precomputation or offline learning. Our research is motivated by the widespread use of commodity cameras that are increasingly used for surveillance and monitoring, including sporting events, public places, religious and political gatherings, etc. One of the key challenges is to devise methods that can automatically analyze the behavior and movement patterns in crowd videos to detect anomalous or atypical behaviors (Li et al., 2015). Furthermore, many of these applications desire interactive or realtime performance, and do not rely on apriori learning or labeling. Many algorithms have been designed to track individual agents and/or to recognize their behavior and movements and detect abnormal behaviors) (Junior et al., 2010). However, current methods are typically limited to sparse crowds or are designed for offline or non-realtime applications.

We present an algorithm for realtime anomaly detection in low to medium density crowd videos. Our approach uses online methods to track each pedestrian and learn the trajectory-level behaviors for each agent by combining non-linear motion models and Bayesian learning. Given a video stream, we extract the trajectory of each agent using a realtime multi-person tracking algorithm that can model different interactions between the pedestrians and the obstacles. Next, we use a Bayesian inference technique to compute the trajectory behavior feature for each agent. These trajectory behavior features are used for anomaly detection in terms of pedestrian movement or behaviors. Our approach involves no offline learning and can be used for interactive surveillance and any crowd videos. We have implemented our system on a multi-core PC and have applied it to both indoor and outdoor crowd videos containing up to tens of pedestrians. We are able to compute crowd agents' trajectories and behavioral features in less than a tenth of a second.

5.3.1 Related Work

There is extensive research in computer vision and multimedia analyzing crowd behaviors and movements from videos (Li et al., 2015). Most of the work has focused on extracting useful information including behavior patterns and situations for surveillance analysis through activity recognition and abnormal behavior detection. Certain methods focus on classifying the most common, simple behavior patterns (linear, radial, etc.) in a given scene. However, most of these methods are designed for offline applications and tend to use a large number of training videos for offline learning of patterns for detecting common crowd behavior patterns (Solmaz et al., 2012), normal and abnormal interactions (Mahadevan et al., 2010), human group activities (Ni et al., 2009). Other methods are designed for crowd analysis using a large number of web videos (Rodriguez et al., 2011). However, these techniques employ either manual selection methods or offline learning techniques for behavior analysis and therefore, cannot be used for interactive applications. Other methods are based on low-density tracking data to learn agent intentions (Musse et al., 2007) or pre-processing techniques that decompose crowd scenes into main agents and background agents (Sun et al., 2013). All of these methods perform offline computations, and it is not clear whether they can be directly used for interactive applications.

5.3.2 Approach

To capture the essence of a pedestrian behavior in a crowd, we need to capture both the individual level movement features and also the dynamics of the group or cluster of which it is a part. Our approach computes global movement flows of pedestrians in semi-dense to dense settings (the importance of global features increases when the density increases). It is not uncommon for some nearby pedestrians to have similar trajectories. As a result, we compute clusters of pedestrians in a crowd based on their positions, velocity, inter-pedestrian distance, orientations, etc. 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 compute a connectivity graph among the pedestrians. There is an edge between vertices of this graph if and only if the two pedestrians 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. Eventually, all the behavior features vectors of every cluster are computed in the same way they are computed per agent. This corresponds to the global pedestrian features $\mathbf{b}^g$ to predict their global movement. This clustering divides the crowd into subsections with similar characteristics.

In the last section, we presented an algorithm to compute the trajectory level behavior features at each time step from a given video. These trajectory level features can be used for different applications related to crowd scene analysis. In this section, we highlight the use of these features and characteristics to detect anomalies based on motion segmentation. Our approach does not use any offline learning methods, and is based on unsupervised classification methods. We highlight their performance on many challenging scenarios.

5.3.2.1 Motion segmentation

The goal behind motion segmentation is to clearly classify variations of pedestrian behaviors in a crowd. Our trajectory-level behavior features can also be used for motion pattern segmentation. Typically, motion pattern segmentation techniques segment spatial regions on an image/video based on the similarity of the pedestrians' movement patterns.

Flow-based methods are often used to segment crowd movements in videos (Ali and Shah, 2007). These techniques mostly work well for structured scenes. Coherent filtering (Zhou et al., 2012a) uses tracklets instead of trajectories; thus, it can accommodate unstructured scenarios. Meta-tracking (Jodoin et al., 2013) tracks sets of particles and is effective for unstructured scenarios with high density crowds. See, for example, (Li et al., 2015). In terms of segmentation results, our method yields similar results as meta-tracking in terms of handling both structured and unstructured scenarios with low or high densities.

We use the K-means data-clustering algorithm to group the trajectories' behavior features observed during a certain time window. Because we are focused on temporal local behavior analysis, we discard the data observed before a particular threshold time or earlier frames. We classify these features into K groups of flows, which we call behavior clusters. K and N are user-defined values that represent the total number of the clusters and the total number of collected behavior features, respectively, and $K \leq N$. A set of behavior clusters $B = \{B_1, B_2, ..., B_K\}$ is computed as follows:

$$\underset{B}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{\mathbf{b}_i \in B_k} dist(\mathbf{b}_i, \mu_k), \tag{5.2}$$

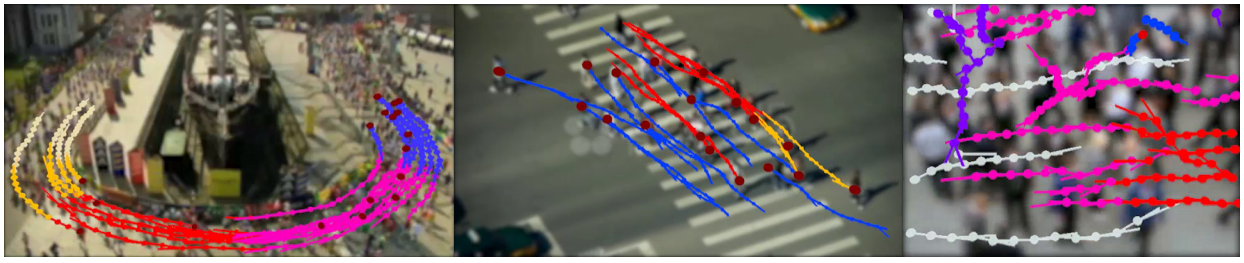
where $\mathbf{b}_i$ is a behavior feature vector, μ_k is a centroid of each cluster, and $dist(b_i, \mu_k)$ is a distance measured between the arguments. Further details about the behavior feature extraction and classification can be found in (Kim et al., 2015).

In our case, the distance between two pedestrian feature vectors is computed as

$$\begin{aligned} dist(\mathbf{b}_i, \mathbf{b}_j) = c_1 \left\| \mathbf{p}_i - \mathbf{p}_j \right\| \\ + c_2 \left\| (\mathbf{p}_i - \mathbf{v}_i^{avg} wdt) - (\mathbf{p}_j - \mathbf{v}_j^{avg} wdt) \right\| \\ + c_3 \left\| \mathbf{g}_i - \mathbf{g}_j \right\|, \end{aligned} \tag{5.3}$$

which corresponds to the weighted sum of the distance between three points: current positions, previous positions, and future positions. c_1, c_2, and c_3 are the weights.

Each behavior cluster is visualized with eight different colors based on the direction of the velocity components of its centroid. Fig. 5.13 shows the segmentation examples in structured, unstructured, and highly unstructured videos. For the Marathon video, we show that the segmentation from the sparse samples matches the behavior patterns of entire crowds. In terms of computation, our algorithm takes only tens of milliseconds for clustering computation during each frame.



(a) Marathon (Ali and Shah, 2007) (b) Crossing (Shao et al., 2014) (c) 879-38 (Rodriguez et al., 2011)

Figure 5.13: **Motion segmentation of structured and unstructured scenarios:** Different colors indicate clusters grouped by similarity of behavior or movement features at interactive rates. We use eight discrete colors for visualization of the results in these benchmarks.

5.3.2.2 Anomaly detection

Anomaly detection is an important problem that has been the focus of research in diverse research areas and applications. It corresponds to the identification of pedestrians, events, or observations that do not conform to an expected pattern or to other pedestrians in a crowd dataset. Typically, the detection

of anomalous items or agents can lead to improved automatic surveillance. Anomaly detection can be categorized into two classes based on the scale of the behavior that is being extracted (Kratz and Nishino, 2009): global anomaly detection and local anomaly detection. A global anomaly typically affects a large portion of, if not the entire, crowd and local anomaly is limited to an individual scale (for example, individuals moving against the crowd flow). We primarily use our trajectory-based behavior characteristics for local anomaly detection. In other words, we detect a few behaviors that are rare and are only observed in the video during certain periods. These periods can be as long as the length of the video or as short as a few hundred frames. In other words, we classify an anomaly as temporally uncommon behavior. For example, a person's behavior going against the flow of crowds may be detected as an anomaly at one point, but the same motion may not be detected as an anomaly later in the frame if many other pedestrians are moving in the same direction.

For anomaly detection we compare the distance between the local and global pedestrian features of every pedestrian (computed using equation 5.3). When an anomaly appears in a scene, the anomaly features typically tend to be isolated in the cluster of which it is a part. In other words, the pedestrian's motion will be different from that of the surrounding crowd. If the Euclidean distance between the *global* and *local* feature is more than a threshold value, we classify it as an anomaly.

$$dist(\mathbf{b}^l, \mathbf{b}^g) > Threshold \tag{5.4}$$

This threshold is a user-tunable parameter. If this threshold is set low, the sensitivity of the anomaly detection will increase and vice-versa.

5.3.3 Quantitative Results

We compare the accuracy of our motion segmentation and anomaly detection methods using the quantitative metrics presented in Table 1 and Table V, as described in Li et al. (Li et al., 2015). Table 1 in (Li et al., 2015) provides a true detection rate for motion pattern segmentation. It is based on the criterion that the approach successfully detected the regions containing the moving pedestrians. Although we cannot directly compare the numbers with pixel-based performance measures, MOTP values (Table 1) can be an indirect measure for the true detection rate motion segmentation. Compared to the values range of **0.4-1.0** in [15], the

corresponding values computed by our approach are in the range of **0.7-0.8** in terms of detecting moving pedestrians, even for unstructured videos. These numbers indicate that the performance of our method is comparable to the state of the art.

(a) (b)

Figure 5.14: **Anomaly Detection**. We also evaluate other datasets like UCSD. Trajectories of 63 real pedestrians are extracted from a video. One person in the middle walks against the flow of crowd. Our method can capture the anomaly of this pedestrian's behavior or movement by comparing the behavior features with those of other pedestrians.

Fig. 5.14 shows the results of anomaly detection in different crowd videos. **879-38 video dataset (Rodriguez et al., 2011)**: The trajectories of 63 pedestrians are extracted from the video. One person in the middle is walking against the flow of pedestrians through a dense crowd. Our method can distinguish the unique behavior of this pedestrian by comparing its behavior features with those found by methods. In **UCSD-Peds1-Biker** and **UCSD-Peds1-Cart** benchmarks, our method is able to distinguish parts of the trajectories of the biker and the cart because their speeds were noticeably different from those of other pedestrians.

Apart from ARENA, we evaluated the accuracy of the anomaly detection algorithm on the UCSD PEDS1 dataset (Mahadevan et al., 2010) and compared it with Table V in Li et al. (Li et al., 2015) in Table 2.

Our method successfully detected the following anomalies in the ARENA - *Person checking vehicle*, *different motion pattern*, *person on a bike*, *push and run*, *abnormal motion near vehicle*, *man touching vehicle*, *hit and run*, *suddenly people running* and *possible mugging*.

Video	Density	Total Frames	BLT
ARENA (01_01)	Low	1060	0.002
ARENA (01_02)	Low	890	0.004
ARENA (03_05)	Low	1440	0.002
ARENA (03_06)	Low	1174	0.002
ARENA (06_01)	Low	2941	0.001
ARENA (06_04)	Low	1582	0.002
ARENA (08_02)	Low	792	0.002
ARENA (08_03)	Low	746	0.006
ARENA (10_03)	Low	1173	0.002
ARENA (10_04)	Low	1188	0.005
ARENA (10_05)	Low	894	0.004
ARENA (11_03)	Low	329	0.001
ARENA (11_04)	Low	729	0.002
ARENA (11_05)	Low	666	0.002
ARENA (14_01)	Low	1081	0.001
ARENA (14_03)	Low	1242	0.004
ARENA (14_05)	Low	1509	0.001
ARENA (14_06)	Low	857	0.002
ARENA (14_07)	Low	1312	0.004
ARENA (15_02)	Low	917	0.004
ARENA (15_05)	Low	903	0.004
ARENA (15_06)	Low	660	0.001
ARENA (22_01)	Low	2079	0.002
ARENA (22_02)	Low	1006	0.001
ARENA (23_01)	Low	712	0.001
Crossing	Medium	238	0.03
Marathon	High	450	0.02
879-38	High	349	0.01
UCSD-Peds1-Cart	Low	200	0.004
UCSD-Peds1-Biker	Low	200	0.009
IITF-5	High	876	0.0512
NPLC-1	Low	775	0.012
NDLS-1	High	941	0.049

Table 5.4: Performance of trajectory level behavior learning on a single core for different benchmarks: We highlight the number of frames of extracted trajectories, the time spent in learning pedestrian behaviors (BLT - Behavior Learning Time (in sec)). Our learning and trajectory computation algorithms demonstrate interactive performance on these complex crowd scene analysis scenarios.

Reference	Dataset	Performance				
		Area under ROC Curve	*Accuracy*	*DR*	*Equal Error Rate*	*Online/Offline*
Our Method	UCSD	**0.873**	**85%**	**-**	**20%**	**Online**
Wang 2012		0.9	-	85%	-	Offline
Cong 2013		0.86	-	-	23.9	Offline
Cong 2012		0.98-0.47	46%	46%	20%	Offline
Thida 2013		0.977	-		17.8%	Offline
Our Method	879-44	**0.97**	**80%**	**-**	**13%**	**Online**
Our Method	ARENA	**0.91**	**76%**	**-**	-	**Online**

Table 5.5: Comparison of Anomaly Detection techniques. All the reference methods have been explained in detail in (Li et al., 2015). Our method has comparable results with the state of the art offline methods in anomaly detection.

Video Name	Camera ID	Threat Level
11_03	TRK_RGB_1	High
15_02	TRK_RGB_1	High
22_02	ENV_RGB_3	High
14_06	TRK_RGB_1	Medium
15_06	TRK_RGB_1	Medium
14_07	TRK_RGB_1	Low
10_04	TRK_RGB_1	Low
06_01	TRK_RGB_1	Low
10_05	TRK_RGB_1	Low

Table 5.6: Details of the anomalies detected in the ARENA Dataset.

CHAPTER 6
Conclusion and Future Work

In this thesis, we have presented interactive algorithms to **track** (Chapter 2), **predict** (Chapter 3) and **learn** pedestrian behaviors(Chapter 4). A large part of our research borrows ideas related to understanding and observing human-like behaviors and their interactions from other fields including psychology, physics, and machine learning. As a result, both short-term, local interaction and long-term, high-level behavior models are improved. Moreover, we highlight applications of our pedestrian tracking, prediction and behavior learning algorithms to many different areas, including computer animation, computer vision, and robotics (Chapter 5). Specifically, we have presented following approaches:

Pedestrian Tracking: We proposed a real time algorithm for pedestrian tracking in crowded scenes. Our approach has a hybrid formulation that combines discrete (microscopic) and continuum (macroscopic) models. The continuum method is used to model the flow of homogeneous clusters within a crowd. The discrete model is used to predict the local interactions and collision avoidance behaviors of each pedestrian. Our discrete formulation computes the best-fit mixture motion model (for microscopic clusters). The motion model parameter estimation is formulated as an optimization problem, and we use an approach that solves this combinatorial optimization problem in a model-independent manner and that is hence scalable to include any multi-agent pedestrian motion model.

Pedestrian Prediction: We proposed a novel real-time algorithm for pedestrian path prediction. Our algorithm is general and can compute global and local movement patterns in real-time with no prior learning and can handle low as well as high density videos and is useful for short-term and long-term prediction. Our approach makes no assumption about pedestrian movement or density, and performs no pre-computation. We observe up to 18% increase in prediction accuracy as compared to prior real-time methods that are based on simple filters or only local movement patterns.

Behavior Learning: We proposed a novel algorithm to classify the personalities of pedestrians in a crowd video. We computed the time-varying motion model of each pedestrian using Bayesian inference and combine it with Personality Trait Theory. We also compute the global movement features and use them to

analyze and simulate different crowd movements or distributions. We evaluated the accuracy with a user study and our results are promising. To the best of our knowledge, this is the first approach for automatic pedestrian personality classification based on their movements in a video.

There are many avenues for future work, in addition to overcoming the limitations of our work discussed in each chapter. Currently, our methods only compute the predictions and behaviors derived from trajectories. We would like to incorporate other pedestrian features like facial expressions, full body motions etc. for learning different aspects of pedestrians interactions. We also want to look into improvements using deep learning and extending our methods to handle higher density crowds.

Our behavior learning approach has some limitations. The underlying formulation does not model many other aspects of pedestrian behavior, including physiological and psychological pedestrian traits as well as age, gender or external environmental factors. The estimation techniques relies on Bayesian inferences and that may not work well in some cases. In terms of future work, we would like to overcome these limitations. The behavior classification is based on personality models and PEN, and may not be sufficient to capture all observed behaviors. We assume that it is possible to extract the trajectory of every pedestrian in a crowd. The global prediction algorithm assumes that the relative distribution of pedestrian behaviors is about the same. One possibility is to combine the classification scheme with other personality models like the Myers-Briggs Type Indicator (Myers et al., 1999). Furthermore, a behavior representation based on six traits may not be sufficient. We would like to integrate these algorithms with different robots and evaluate their performance in crowded indoor and outdoor scenes. We would also like to take into account cultural norms and group behaviors during social-aware navigation.

BIBLIOGRAPHY

Ali, S. and Shah, M. (2007). A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *Computer Vision and Pattern Recognition. IEEE Conference on*, pages 1–6.

Ali, S. and Shah, M. (2008). Floor fields for tracking in high density crowd scenes. In *ECCV*, pages 1–14.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, pages 174–188.

Badler, N. (1997a). Virtual humans for animation, ergonomics, and simulation. In *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, pages 28–36. IEEE.

Badler, N. I. (1997b). Real-time virtual humans. In *Computer Graphics and Applications, 1997. Proceedings., The Fifth Pacific Conference on*, pages 4–13. IEEE.

Bainbridge, W. S. (2007). The scientific research potential of virtual worlds. *science*, 317(5837):472–476.

Bandini, S., Federici, M., Manzoni, S., and Vizzari, G. (2006). Towards a methodology for situated cellular agent based crowd simulations. In Dikenelli, O., Gleizes, M.-P., and Ricci, A., editors, *Engineering Societies in the Agents World VI*, volume 3963 of *Lecture Notes in Computer Science*, pages 203–220. Springer Berlin Heidelberg.

Baylor, A. L. (2009). Promoting motivation with virtual agents and avatars: role of visual presence and appearance. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 364(1535):3559–3565.

Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S. (2005). Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31–48.

Bera, A. and Manocha, D. (2014a). Realtime multilevel crowd tracking using reciprocal velocity obstacles. In *Proceedings of Conference on Pattern Recognition, Sweden*.

Bera, A. and Manocha, D. (2014b). Realtime multilevel crowd tracking using reciprocal velocity obstacles. In *ICPR*.

Borges, P., Conci, N., and Cavallaro, A. (2013). Video-based human behavior understanding: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(11):1993–2008.

Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., and Van Gool, L. (2011). Online multiperson tracking-by-detection from a single, uncalibrated camera. *PAMI*, pages 1820–1833.

Bruce, A. and Gordon, G. (2004). Better motion prediction for people-tracking. In *Proc. of the International Conference on Robotics and Automation (ICRA), New Orleans, USA*.

Burgoon, J. K. and Jones, S. B. (1976). Toward a theory of personal space expectations and their violations. *Human Communication Research*, 2(2):131–146.

Chattaraj, U., Seyfried, A., and Chakroborty, P. (2009). Comparison of pedestrian fundamental diagram across cultures. In *Advances in Complex Systems*, pages 393–405.

Cheung, E., Wong, T. K., Bera, A., Wang, X., and Manocha, D. (2016). Lcrowdv: Generating labeled videos for simulation-based crowd behavior learning. In *European Conference on Computer Vision*, pages 709–727. Springer International Publishing.

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *CVPR*, pages 142–149.

Cristani, M., Raghavendra, R., Del Bue, A., and Murino, V. (2013). Human behavior analysis in video surveillance: A social signal processing perspective. *Neurocomputing*, 100:86–97.

Cui, J., Zha, H., Zhao, H., and Shibasaki, R. (2005). Tracking multiple people using laser and vision. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2116–2121. IEEE.

Curtis, S. and Manocha, D. (2012). Pedestrian simulation using geometric reasoning in velocity space. In *Pedestrian and Evacuation Dynamics (PEDS)*.

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

Du Toit, N. and Burdick, J. (2010). Robotic motion planning in dynamic, cluttered, uncertain environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 966–973.

Durupinar, F., Pelechano, N., Allbeck, J., Gü anddü andkbay, U., and Badler, N. (2011). How the ocean personality model affects the perception of crowds. *Computer Graphics and Applications, IEEE*, 31(3):22 –31.

Enzweiler, M. and Gavrila, D. M. (2009). Monocular pedestrian detection: Survey and experiments. *PAMI*, pages 2179–2195.

Eysenck, H. and Eysenck, M. (1985). *Personality and individual differences: A natural science approach*. Plenum Press New York.

Ferrer, G., Garrell, A., and Sanfeliu, A. (2013). Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *Intelligent robots and systems (IROS), 2013 IEEE/RSJ international conference on*, pages 1688–1694. IEEE.

Fulgenzi, C., Spalanzani, A., and Laugier, C. (2007). Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1610–1616.

Funge, J., Tu, X., and Terzopoulos, D. (1999). Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *SIGGRAPH*, pages 29–38. ACM Press.

Ge, W., Collins, R. T., and Ruback, B. (2009). Automatically detecting the small group structure of a crowd. In *Applications of computer vision (wacv), 2009 workshop on*, pages 1–8. IEEE.

Godoy, J., Karamouzas, I., Guy, S. J., and Gini, M. (2016). Moving in a crowd: Safe and efficient navigation among heterogeneous agents. In *Proc. Int. Joint Conf. on Artificial Intelligence*.

Gong, H., Sim, J., Likhachev, M., and Shi, J. (2011). Multi-hypothesis motion planning for visual object tracking.

Guy, S. J., Chhugani, J., Curtis, S., Dubey, P., Lin, M., and Manocha, D. (2010). PLEdestrians: a least-effort approach to crowd simulation. In *Symp. on Computer Animation*, page 119128.

Guy, S. J., Kim, S., Lin, M. C., and Manocha, D. (2011). Simulating heterogeneous crowd behaviors using personality trait theory. In *Symposium on Computer Animation*, pages 43–52. ACM.

Guzzi, J., Giusti, A., Gambardella, L., Theraulaz, G., and Di Caro, G. (2013). Human-friendly robot navigation in dynamic environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 423–430.

Hall, E. T. (1966). *The hidden dimension*. Doubleday & Co.

Harvey, R. J., Murry, W. D., and Stamoulis, D. T. (1995). Unresolved issues in the dimensionality of the myers-briggs type indicator. *Educational and Psych. Measurement*.

Helbing, D. and Molnar, P. (1995a). Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282.

Helbing, D. and Molnar, P. (1995b). Social force model for pedestrian dynamics. *Physical review E*.

Henry, P., Vollmer, C., Ferris, B., and Fox, D. (2010). Learning to navigate through crowded environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 981–986.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–72.

Hu, W., Tan, T., Wang, L., and Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352.

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

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

Jodoin, P.-M., Benezeth, Y., and Wang, Y. (2013). Meta-tracking for video scene understanding. In *Advanced Video and Signal Based Surveillance, IEEE International Conference on*, pages 1–6.

Junior, S. J. et al. (2010). Crowd analysis using computer vision techniques. *IEEE Signal Processing Magazine*, 27(5):66–77.

Karamouzas, I., Heil, P., van Beek, P., and Overmars, M. (2009). A predictive collision avoidance model for pedestrian simulation. *Motion in Games*, pages 41–52.

Karamouzas, I. and Overmars, M. (2010). A velocity-based approach for simulating human collision avoidance. In *Intelligent Virtual Agents*, pages 180–186. Springer.

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

Khan, Z., Balch, T., and Dellaert, F. (2004). An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV*, pages 279–290.

Kim, S., Bera, A., Best, A., Chabra, R., and Manocha, D. (2016). Interactive and adaptive data-driven crowd simulation. In *IEEE Virtual Reality (VR)*, pages 29–38. IEEE.

Kim, S., Bera, A., and Manocha, D. (2015). Interactive crowd content generation and analysis using trajectory-level behavior learning. In *International Symposium on Multimedia*. IEEE.

Kim, S., Guy, S. J., Liu, W., Wilkie, D., Lau, R. W., Lin, M. C., and Manocha, D. (2014). Brvo: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research*, page 0278364914555543.

Kim, S., Guy, S. J., Manocha, D., and Lin, M. C. (2012). Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

Kirchner, A. and Schadschneider, A. (2002). Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A: statistical mechanics and its applications*, 312(1):260–276.

Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. In *ACM transactions on graphics (TOG)*, volume 21, pages 473–482. ACM.

Kratz, L. and Nishino, K. (2009). Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *Computer Vision and Pattern Recognition. IEEE Conference on*, pages 1446–1453. IEEE.

Kratz, L. and Nishino, K. (2011). Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (99):11.

Kratz, L. and Nishino, K. (2012). Going with the flow: pedestrian efficiency in crowded scenes. In *ECCV*.

Kretzschmar, H., Kuderer, M., and Burgard, W. (2014). Learning to predict trajectories of cooperatively navigating agents. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4015–4020. IEEE.

Kruse, T., Pandey, A. K., Alami, R., and Kirsch, A. (2013). Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743.

Kuderer, M., Kretzschmar, H., Sprunk, C., and Burgard, W. (2012). Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: science and systems*.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Li, T., Chang, H., Wang, M., Ni, B., Hong, R., and Yan, S. (2015). Crowded scene analysis: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 25(3):367–386.

Li, Y., Ai, H., Yamashita, T., Lao, S., and Kawade, M. (2008a). Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *PAMI*, pages 1728–1740.

Li, Z., Tang, Q., and Sang, N. (2008b). Improved mean shift algorithm for occlusion pedestrian tracking. *Electronics Letters*, pages 622–623.

Liao, L., Fox, D., Hightower, J., Kautz, H., and Schulz, D. (2003). Voronoi tracking: Location estimation using sparse and noisy sensor data. In *Proc. of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 723–728. IEEE.

Luber, M., Stork, J., Tipaldi, G., and Arras, K. (2010). People tracking with human motion predictions from social forces. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 464–469.

Mahadevan, V., Li, W., Bhalodia, V., and Vasconcelos, N. (2010). Anomaly detection in crowded scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1975–1981.

McPhail, C. and Wohlstein, R. T. (1982). Using film to analyze pedestrian behavior. *Sociological Methods & Research*, 10(3):347–375.

Mogelmose, A., Trivedi, M. M., and Moeslund, T. B. (2015). Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 330–335.

Musse, S. R., Jung, C. R., Jacques, J. C. S., and Braun, A. (2007). Using computer vision to simulate the motion of virtual agents. *Computer Animation and Virtual Worlds*, 18(2):83–93.

Myers, I., McCaulley, M., Quenk, N., and Hammer, A. (1999). *MBTI manual*. Consulting Psychologists Press.

Ni, B., Yan, S., and Kassim, A. (2009). Recognizing human group activities with localized causalities. In *Computer Vision and Pattern Recognition, 2009. IEEE Conference on*, pages 1470–1477. IEEE.

Okal, B. and Arras, K. O. (2016). Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2889–2895. IEEE.

Paris, S. and Donikian, S. (2009). Activity-driven populace: A cognitive approach to crowd simulation. *IEEE Computer Graphics and Applications*, 29(4):34–43.

Pellegrini, S., Ess, A., Schindler, K., and Van Gool, L. (2009a). You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, pages 261–268.

Pellegrini, S., Ess, A., Schindler, K., and Van Gool, L. (2009b). You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proc. of the IEEE 12th International Conference on Computer Vision*, page 261268.

Pellegrini, S., Gall, J., Sigal, L., and Gool, L. (2012). Destination flow for crowd simulation. In Fusiello, A., Murino, V., and Cucchiara, R., editors, *ECCV 2012. Workshops and Demonstrations*, volume 7585 of *Lecture Notes in Computer Science*, pages 162–171. Springer Berlin Heidelberg.

Pervin, L. (2003). *The Science of Personality*. Oxford University Press, Oxford.

Pettré, J., Ondřej, J., Olivier, A. H., Cretual, A., and Donikian, S. (2009). Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Symp. on Computer Animation*, page 189198.

Pradhan, N., Burg, T., and Birchfield, S. (2011). Robot crowd navigation using predictive position fields in the potential function framework. In *American Control Conference (ACC), 2011*, pages 4628–4633.

Ren, Z., Gai, W., Zhong, F., Pettré, J., and Peng, Q. (2013). Inserting virtual pedestrians into pedestrian groups video with behavior consistency. *The Visual Computer*, 29(9):927–936.

Reynolds, C. W. (1999a). Steering behaviors for autonomous characters. In *Game Developers Conference. http://www. red3d. com/cwr/steer/gdc99*.

Reynolds, C. W. (1999b). Steering behaviors for autonomous characters. In *Game Developers Conference 1999*.

Reynolds, T. R. (1987). Stride length and its determinants in humans, early hominids, primates, and mammals. *American Journal of Physical Anthropology*.

Rivalcoba, J., Gyves, O., Rudomin, I., and Pelechano, N. (2014). Coupling pedestrians with a simulated virtual crowd. In *Proc. of the International Conference on Computer Graphics and Applications (GRAPP'2014)*.

Rodriguez, M., Sivic, J., Laptev, I., and Audibert, J.-Y. (2011). Data-driven crowd analysis in videos. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1235–1242.

Rodriguez, M. and Sivic, J. e. a. (2011). Data-driven crowd analysis in videos. In *ICCV*, pages 1235–1242.

Salvit, J. and Sklar, E. (2011). Toward a Myers-Briggs Type Indicator Model of Agent Behavior in Multiagent Teams. *Multi-Agent-Based Simulation XI*, pages 28–43.

SanMiguel, J. C., Cavallaro, A., and Martínez, J. M. (2012). Standalone evaluation of deterministic video tracking. In *ICIP*, pages 1353–1356.

Schneider, N. and Gavrila, D. M. (2013). Pedestrian path prediction with recursive bayesian filters: A comparative study. In *Pattern Recognition - 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013. Proceedings*, pages 174–183.

Sean Curtis, Andrew Best, D. M. (2013). Menge: A modular framework for simulating crowd movement. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill.

Shao, J., Loy, C., and Wang, X. (2014). Scene-independent group profiling in crowd. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2227–2234.

Shao, W. and Terzopoulos, D. (2005). Autonomous pedestrians. In *Symposium on Computer animation*, pages 19–28.

Snape, J., van den Berg, J., Guy, S., and Manocha, D. (2011). The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics (TRO)*, 27(4):696–706.

Solmaz, B., Moore, B. E., and Shah, M. (2012). Identifying behaviors in crowd scenes using stability analysis for dynamical systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10):2064–2070.

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

Sun, L., Li, X., and Qin, W. (2013). Simulating realistic crowd based on agent trajectories. *Computer Animation and Virtual Worlds*, 24(3-4):165–172.

Takayama, L. and Pantofaru, C. (2009). Influences on proxemic behaviors in human-robot interaction. In *Intelligent robots and systems, 2009. IROS 2009. IEEE/RSJ international conference on*, pages 5495–5502. IEEE.

Trautman, P., Ma, J., Murray, R., and Krause, A. (2013). Robot navigation in dense human crowds: the case for cooperation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2153–2160.

Treuille, A., Cooper, S., and Popovi, Z. (2006). Continuum crowds. In *ACM SIGGRAPH 2006*, pages 1160–1168.

Turner, R. H. and Killian, L. M. (1987). *Collective Behavior*. Prentice Hall.

Ulicny, B. and Thalmann, D. (2002). Towards interactive real-time crowd behavior simulation. In *Computer Graphics Forum*, volume 21, pages 767–775. Wiley Online Library.

Van Den Berg, J., Guy, S., Lin, M., and Manocha, D. (2011a). Reciprocal n-body collision avoidance. *Robotics Research*, page 319.

Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011b). Reciprocal n-body collision avoidance. In *Robotics Research*.

van den Berg, J., Lin, M., and Manocha, D. (2008a). Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928 –1935.

van den Berg, J., Patil, S., Sewall, J., Manocha, D., and Lin, M. (2008b). Interactive navigation of individual agents in crowded environments. In *Symp. on Interactive 3D Graphics and Games (I3D 2008)*.

Wang, H., Ondřej, J., and O'Sullivan, C. (2016). Path patterns: Analyzing and comparing real and simulated crowds. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 49–57. ACM.

Wang, H., Ondřej, J., and OSullivan, C. (2017). Trending paths: A new semantic-level metric for comparing simulated and real crowd data. *IEEE transactions on visualization and computer graphics*, 23(5):1454–1464.

Wilkie, D., van den Berg, J., and Manocha, D. (2009). Generalized velocity obstacles. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 5573–5578, Piscataway, NJ, USA. IEEE Press.

Williams, J. L. (1971). Personal space and its relation to extraversion-introversion. *Canadian Journal of Behavioural Science/Revue canadienne des sciences du comportement*, 3(2):156.

Wolinski, D., Guy, S. J., Olivier, A.-H., Lin, M. C., Manocha, D., and Pettré, J. (2014). Parameter estimation and comparative evaluation of crowd simulations. In *Eurographics*.

Wu, S., Moore, B. E., and Shah, M. (2010). Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2054–2060. IEEE.

Yamaguchi, K., Berg, A., Ortiz, L., and Berg, T. (2011a). Who are you with and where are you going? In *Proc. of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1345–1352.

Yamaguchi, K., Berg, A. C., Ortiz, L. E., and Berg, T. L. (2011b). Who are you with and where are you going? In *CVPR*.

Yi, S., Li, H., and Wang, X. (2015). Understanding pedestrian behaviors from stationary crowd groups. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3488–3496.

Zen, G. and Ricci, E. (2011). Earth mover's prototypes: A convex learning approach for discovering activity patterns in dynamic scenes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3225–3232.

Zhang, Y., Qin, X., Pettre, J., Peng, Q., et al. (2010). Real-time inserting virtual characters into dynamic video scene. In *Proceedings of the Chinese Conference on Computer Graphics (CHINAGRAPH 2010)*.

Zhao, X., Gong, D., and Medioni, G. (2012). Tracking using motion patterns for very crowded scenes. In *ECCV*, pages 315–328.

Zhong, J., Cai, W., Luo, L., and Yin, H. (2015). Learning behavior patterns from video: A data-driven framework for agent-based crowd modeling. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 801–809, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Zhou, B., Tang, X., and Wang, X. (2012a). Coherent filtering: Detecting coherent motions from crowd clutters. In *Proc. of the European Conference on Computer Vision - Part II*, pages 857–871.

Zhou, B., Wang, X., and Tang, X. (2012b). Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2871–2878.

Ziebart, B. D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J. A., Hebert, M., Dey, A. K., and Srinivasa, S. (2009a). Planning-based prediction for pedestrians. In *Proc. of the 2009 IEEE/RSJ international conference on Intelligent robots and systems (IROS)*, IROS'09, pages 3931–3936, Piscataway, NJ, USA. IEEE Press.

Ziebart, B. D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J. A., Hebert, M., Dey, A. K., and Srinivasa, S. (2009b). Planning-based prediction for pedestrians. In *IROS*, pages 3931–3936.