DATA-DRIVEN TECHNIQUES FOR VIRTUAL CROWDS

Panayiotis Charalambous

University of Cyprus, 2013

Virtual crowds are important in a variety of applications such as computer games, movies, training simulations and safety modeling. Increasing processing power enables designers and programmers to add multitudes of virtual characters in real-time applications. Despite these advances, there is a significant gap between rendered appearance and simulated behaviour of crowds. This thesis is addressing some of the shortcomings of data driven techniques for simulating and evaluating crowd behaviours.

Firstly, the *Perception Action Graph (PAG)* framework is proposed for efficient data-driven crowd simulation. By employing this framework using as input data from videos of real world crowds, fast, consistent and believable steering behaviours for human crowds can be generated.

Secondly, we propose a multi-objective data-driven framework for crowd evaluation. This method employs recent methods in Machine Learning for novelty/outlier detection under multiple criteria. Using as input well behaved crowds, the proposed framework identifies abnormal parts of the simulation and pinpoints them for further examination.

Finally, we believe that by gaining a higher level understanding of crowd behaviours is a step towards better crowd simulation, evaluation and authoring. A method that annotates pedestrian trajectory segments into higher level descriptors is presented that uses both local and global knowledge of crowd trajectories. This method successfully identifies behaviours such as wandering around and group formations.

DATA-DRIVEN TECHNIQUES FOR VIRTUAL CROWDS

Panayiotis Charalambous

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Cyprus

Recommended for Acceptance

by the Department of Computer Science

December, 2013

APPROVAL PAGE

Doctor of Philosophy Dissertation

DATA-DRIVEN TECHNIQUES FOR VIRTUAL CROWDS

Presented by

Panayiotis Charalambous

Research Supervisor ______________________________
Yiorgos Chrysanthou, Associate Professor, UCY, Cyprus

Committee Member ______________________________
Chris Christodoulou, Associate Professor, UCY, Cyprus

Committee Member ______________________________
Yianos Sazeides, Associate Professor, UCY, Cyprus

Committee Member ______________________________
Julien Pettré, Research Scientist INRIA, France

Committee Member ______________________________
Nuria Pelechano, Assistant Professor, UPC, Spain

University of Cyprus

December, 2013

ACKNOWLEDGEMENTS

"As you set out for Ithaka
hope the voyage is a long one,
full of adventure, full of discovery.
Laistrygonians and Cyclops,
angry Poseidon–don't be afraid of them:
you'll never find things like that on your way
as long as you keep your thoughts raised high,
as long as a rare excitement
stirs your spirit and your body.
Laistrygonians and Cyclops,
wild Poseidon–you won't encounter them
unless you bring them along inside your soul,
unless your soul sets them up in front of you."
(Ithaka by Constantine P. Cavafy)

The past few years have been full of very intense, interesting, beautiful, often frustrating experiences. I dived into a completely novel for me area just because I thought the whole idea of simulating something that looks and feels intelligent on a computer screen was cool! It's during these past few years that I felt humbled by the research performed by other researchers and understood what it means to dedicate yourself to reaching an often seemingly endless road.

I wouldn't be able to achieve much without the help of other people. I would like to first give my sincere thanks to Yiorgos, the captain of my trip during the past few years, who has been a very valuable and inspiring advisor and friend. He helped me stay on course (well, at least he tried to) and gave me guidance and support when I needed it. Secondly, I would like to thank my co-travellers in the lab: Stathis, Nicolas, Marios, Andreas, Despina, Haris and all the students that were part of our research team over the years. I wouldn't be able to achieve my goals without the support and love of Christiana, who was with me for a large portion of my travels both mentally and physically. And finally, I would like to dedicate this work to my family; my beloved mother

Chrystalla who shaped me since childhood and taught me to be strong, my father Savvas and my awesome brother Telis and his family.

Reaching to Ithaka has been a long trip indeed, full of discovery, frustration, endless fights with monsters (compilers, bad data, bad mood, you name it) but it was worth it. I am now a completely different man ready to set out for new journeys and nothing can stop me enjoying the trips. Thank you all, from the bottom of my heart and may your journeys be long and beautiful.

CREDITS

As part of this thesis, a number of papers and tools have been developed. More specifically, a paper describing the crowd simulation framework that is presented in Chapter 4 has been conditionally accepted at Computer Graphics Forum [7] and is currently under final production. Additionally, papers that discuss different data-driven crowd simulation approaches proposed during the development of this thesis have been written and published in various conferences [4, 5]. The proposed crowd simulation framework is integrated in a cultural heritage project for which a number of papers have already been published [8, 32]. A paper is being written for the multi-criteria anomaly detection algorithm proposed in Chapter 5 to be submitted at *Transactions on Graphics*. Finally, initial work for the crowd annotation framework presented in Chapter 6 has been presented at a conference [6] and a paper covering all of the proposed work is currently under preparation.

List of Publications

- CHARALAMBOUS, P., AND CHRYSANTHOU, Y. The PAG Crowd: A Graph Based Approach for Efficient Data-Driven Crowd Simulation. *Computer Graphics Forum* (2014). Under print
- CHARALAMBOUS, P., KARAMOUZAS, I., GUY, S., AND CHRYSANTHOU, Y. Data-Driven Crowd Analysis using Pareto Optimality. In preparation, 2014
- CHARALAMBOUS, P., AND CHRYSANTHOU, Y. Classifying Pedestrian Behaviour using Random Forests, 2013. Poster presented at the 6th International Conference on Motion in Games 2013

- Charalambous, P., Iliadou, H., Apostolou, C., and Chrysanthou, Y. Reconstruction of Everyday Life in 19^{th} Century Nicosia. In *Proceedings of the 4th International Conference on Progress in Cultural Heritage Preservation* (Berlin, Heidelberg, 2012), EuroMed'12, Springer-Verlag, pp. 568–577

- Iliadou, H., Charalambous, P., Apostolou, C., and Chrysanthou, Y. Reviving Nicosia of the XIXth Century. In *11th International Conference on Urban History* (2012), EAUH'12

- Charalambous, P., and Chrysanthou, Y. Learning Crowd Behavior. In *Workshop on Crowd Simulation, Collocated with the 23rd International Conference on Computer Animation and Social Agents (CASA2010)* (2010)

- Charalambous, P., and Chrysanthou, Y. Learning Crowd Steering Behaviors from Examples. In *Proceedings of the Third International Conference on Motion in Games* (Berlin, Heidelberg, 2010), MIG'10, Springer-Verlag, pp. 35–35

List of Software Tools

In the following paragraphs we give a brief description of the software tools that have been developed as part of this thesis. Most of these software tools are planned to be released to the community.

Chapter 4: Data-Driven Simulation Tools

1. PAG Crowd Simulator Framework: The Data-Driven System responsible for simulating crowds based on data of real crowds.

2. Crowd Simulator based on Decision Trees: Initial implementation of a data-driven crowd simulator based on state-action decision trees. This idea was abandoned after some time for the PAG framework that relied on a completely new approach.
3. Crowd simulator similar to Lerner et al. [51]: This was implemented for comparison purposes.
4. Crowd Tracker: A tool to manually track crowds from videos to be used as input for data-driven crowd simulators (Developed with the help of an intern student).
5. TPP generator and pre-processor: A tool to generate TPPs from data of crowds based on different state representations (Chapter 3). Currently two state representations have been implemented (Visibility based and Lee et al. [48]) based.
6. Crowd Data Explorer: A tool to explore data from crowds that include: TPPs, similarity maps, distance matrices, the PAG graph, the TPPs in nodes and comparison between people TPPs in the same input datasets.
7. PAG Server: A multi-threaded server based implementation of the PAG for use by different rendering clients. This is multi-threaded in the sense that it can handle multiple clients – simulations are still single threaded.
8. Batch Crowd Simulator: A tool that runs batch simulations based on different parameters, stores simulations on disk and extracts data such as crowd positions over time, collisions, run times, PAG statistics, etc. Simulations can be run multiple times to get average and standard deviation values of statistics.
9. Crowd Statistics Gatherer: A tool to analyze statistically each individual in the crowd both locally and globally. Statistics can be divided into individual and interpersonal.

10. Unity 3D Crowd Viewer: A real-time crowd rendering tool based on the Unity 3D Game Engine [101].

11. Unity 3D Crowd Client: A real-time crowd client based on the Unity 3D Game Engine [101], to be used in a real time environment.

12. OpenGL Crowd Renderer: A simple real-time crowd rendering application based in OpenGL for visualization purposes and debugging.

Chapter 5: Data-Driven Evaluation

Crowd Outlier Detector and Visualizer: A data-driven outlier/novelty detection tool based on the work described in Chapter 5. This tool integrates 4 different outlier detection algorithms: PDA k-LPE, One Class SVM and various k-NN based approaches such as distance to the k-th neighbor and weighted distance to all k-NN neighbors. This tool supports a variety of individual and inter-personal statistics over segments or whole trajectories. Local and global statistics can be intermixed.

Chapter 6: Higher Level Annotation of Crowds

RanForest: Higher level automatic annotation of individuals in crowds based on local and global characteristics of crowds. The system was implemented as part of Chapter 6.

TABLE OF CONTENTS

Chapter 1: Introduction **1**

1.1 Crowds . . . 1

1.2 Virtual Human Crowds . . . 3

1.3 Problem statement / Motivation/ Contributions . . . 10

1.3.1 Data-Driven Crowd Simulation . . . 11

1.3.2 Crowd Evaluation . . . 13

1.3.3 Crowd Behaviour Annotation . . . 15

1.4 Thesis Outline . . . 16

Chapter 2: Related Work **18**

2.1 Virtual Crowds . . . 18

2.2 Crowd Behaviour . . . 19

2.2.1 High Level Behaviour . . . 20

2.2.2 Path Planning . . . 21

2.3 Crowd Steering . . . 22

2.3.1 Macroscopic Simulations . . . 23

2.3.2 Microscopic Simulation . . . 25

2.4 Evaluation and Analysis . . . 31

2.4.1 User Based Evaluation . . . 32

2.4.2 Statistics Based Evaluation . . . 32

2.4.3 Data-Driven Evaluation . . . 33

2.4.4 Automatic Behaviour Annotation . . . 33

2.5 Conclusion . . . 34

Chapter 3: Framework 36

3.1 Agent Based Crowd Simulation 36

3.1.1 A Model for Humans 37

3.1.2 Virtual Humans as Intelligent Agents 40

3.1.3 Basic Simulator 42

3.1.4 State Representation 44

3.2 Discussion 47

Chapter 4: The PAG Graph

Efficient Data Driven Crowd Simulation 49

4.1 Introduction 49

4.2 Overview 51

4.2.1 Data Preprocessing 53

4.2.2 Simulation 54

4.3 Graph Construction 55

4.3.1 Data Acquisition 55

4.3.2 State Representation 56

4.3.3 TPP Similarity 59

4.3.4 TPP Interconnection 62

4.4 Simulation 63

4.4.1 Basic Idea 64

4.4.2 Initialization 65

4.4.3 Interaction Free States 67

4.4.4 Agent Interaction 67

4.5 Results . . . 73

4.5.1 Preprocessing . . . 76

4.5.2 Scenarios . . . 77

4.5.3 Simulation Update Step . . . 85

4.5.4 Speed vs Quality . . . 87

4.6 Discussion . . . 94

Chapter 5: Data Driven Evaluation of Crowds 97

5.1 Introduction . . . 97

5.1.1 Contributions . . . 101

5.2 Method Overview . . . 102

5.2.1 Comparison Metrics . . . 103

5.2.2 Segments . . . 105

5.2.3 Anomaly Detection . . . 107

5.3 Multiple-Criteria Crowd Analysis . . . 113

5.3.1 Pareto Depth Analysis . . . 115

5.4 Results . . . 119

5.4.1 Crowd Analysis . . . 120

5.4.2 Real-time Crowd Analysis . . . 123

5.5 Discussion and Future Work . . . 125

Chapter 6: Behaviour Annotation 128

6.1 Introduction . . . 128

6.2 Methodology . . . 129

6.3 Experiments . . . 133

6.3.1 Evaluating crowd simulations using higher level annotations 137

6.4 Discussion 139

Chapter 7: Conclusions and Future Work **142**

7.1 Contributions 142

7.2 Future Work 144

Bibliography **148**

Appendices **162**

Appendix A: Simulation Performance of PAG **163**

A.1 Pedestrians 1 (lerner-zara) 163

A.2 Pedestrians 2 (eth-hotel) 163

A.3 Small Groups (lee-chat) 164

A.4 Simulated Data (reynolds-pedestrians) 164

LIST OF TABLES

4.1 Input Datasets used for the experiments in this work. 73

5.1 Difference Metrics 105

5.2 Difference Operators 106

5.3 Timing results for evaluating various datasets 126

6.1 Table classes 130

6.2 Table of statistics that were selected as features for classifying 135

6.3 The datasets used on the annotation experiments 135

LIST OF FIGURES

1.1 Human Crowds 2

1.2 Crowds in Nature 3

1.3 Artificial Crowds 4

1.4 Crowds in Movies 5

1.5 Crowds in World War Z added in post-production phase 6

1.6 Various crowds in games, such as The Sims and Assasin's Creed 7

1.7 A typical crowd simulator for virtual humans has to deal with most of these components depending on target applications. 10

1.8 The PAG framework can be used to simulate realistic crowds 12

2.1 Typical crowd simulation layers 19

2.2 Shao et al. [89] crowd simulaton framework 20

2.3 The HiDAC behaviour components 21

2.4 Crowd steering categories 22

2.5 Continuum Crowds algorithm from [99] 23

2.6 Vector field learning from Courty et al. [11]. 24

2.7 Reynolds defined three simple rules to simulate flocks of birds. 26

2.8 Results for flocking behaviours using Reynolds approach. 26

2.9 The data driven framework from Lerner et al. [51] 29

2.10 Various data driven crowd simulation frameworks 30

2.11 Morphable Crowds by Ju et al. [34]. 30

2.12 Li et al. [54] proposed recording small crowds in a controlled environments and then generating larger groups of people using crowd patches. 31

3.1 A human can be considered as an autonomous entity that senses all the time, processes information and acts, while at the same it learns and adapts. 38
3.2 Information theoretic approach to modeling a human by Tor Nørretranders [65]. . 39
3.3 Agent Based Simulation 40
3.4 Agent Based Simulation 45
3.5 Various agent state representations 46
3.6 PAG Simulation Framework 47
3.7 PAG Simulation Framework 48
4.1 PAG Simulation Framework 52
4.2 The PAG framework's preprocessing pipeline 53
4.3 Agents' state diagram. 54
4.4 Perception Pattern 56
4.5 The Perception Action Graph (PAG) 57
4.6 Temporal Perception Pattern 58
4.7 Distance Matrices for 4 different pedestrians 60
4.8 Similarity Maps 61
4.9 Temporal Perception Pattern compression with DCT 62
4.10 State transitions example for three agents (A-C) 65
4.11 Binning agents for performance 66
4.12 Searching the Interaction Free database of trajectories 68
4.13 Action Selection Mechanism for agents in the PAG framework 70
4.14 PAG framework implementation 74
4.15 Frame grabs from the input datasets used in the experiments in this thesis. 74
4.16 Two opposing agents scenario 78

4.17 Four opposing agents scenario 79

4.18 14 agents in a circular scenario using the Reynolds [82] dataset 80

4.19 14 agents in a circular scenario using different datasets 81

4.20 Trajectories for agents moving in a ”butterfly” like formation. 82

4.21 Manhattan like environment simulation results 83

4.22 Agents writing letters on the ground using the eth-hotel dataset 84

4.23 Agents writing letters on the ground using as input the lee-chat dataset 85

4.24 Agents can follow complex paths whilst navigating 86

4.25 Agents can follow complex paths whilst navigating 87

4.26 The effect of changing update frequency of agents 88

4.27 PAG speedup over kNN with different update frequencies 89

4.28 Timing comparison between the PAG and the kNN approach for a simulation of 32 agents in a busy street. 90

4.29 Speedup of the PAG over the *Full kNN* approach – PAG outperforms it by up to $x14$. 91

4.30 Comparison of PAG and kNN for students on the manhattan scenario 95

4.31 Comparison of PAG and kNN for students on the chinese hanzi scenario 96

5.1 Our outlier detection system overview 100

5.2 The anomaly detection tool interface 104

5.3 Personal vs Inter-personal Difference Measures 105

5.4 Space/time trajectories are split into overlapping segments 106

5.5 Decision boundaries as estimated by One Class SVM on random data. 111

5.6 Scalarization performance 113

5.7 Hand-drawn example of outlier detection where PDA is prefered ovel convex combinations for other methods 114
5.8 Pareto Fronts 115
5.9 Pareto Fronts for the Zara01 dataset 117
5.10 Step-by-step Pareto Fronts construction for a random dataset of 100 points. 118
5.11 Source data used for the multi-criteria evaluation algorithms 121
(a) Assasin's Creed 121
(b) Grouping 121
(c) Zara 121
5.12 100 agents crosssing scenario experiment 122
(a) Trajectories 122
(b) Segments 122
5.13 Against the flow experiment 123
(a) Trajectories 123
(b) Dyads 123
5.14 Example outlier detection on the Assasin's Creed game 124
5.15 Randomized PDA comparison to the original PDA approach 125
6.1 Splitting trajectories into smaller overlapping segments 131
6.2 Examples that fall in the same classes 132
6.3 FPerformance score of different classifiers on the input data 133
6.4 Feature importance of the selected features 134
6.5 Example annotations on videos of pedestrians 136
6.6 Classification results for varius datasets 138
6.7 Classification results for PAG simulations 140

A.1 Comparison of PAG and kNN for lerner-zara on the manhattan scenario 166

A.2 Comparison of PAG and kNN for lerner-zara on the chinese hanzi scenario 167

A.3 Comparison of PAG and kNN for eth-hotel on the manhattan scenario 168

A.4 Comparison of PAG and kNN for eth-hotel on the chinese hanzi scenario 169

A.5 Comparison of PAG and kNN for lee-chat on the manhattan scenario 170

A.6 Comparison of PAG and kNN for lee-chat on the chinese hanzi scenario 171

A.7 Comparison of PAG and kNN for reynolds on the manhattan scenario 172

A.8 Comparison of PAG and kNN for reynolds on the chinese hanzi scenario 173

Chapter 1

Introduction

"The intelligence of that creature known as a crowd is the square root of the number of people in it."
(Terry Pratchett, Jingo)

1.1 Crowds

A *crowd* is a number of persons or living beings gathered or considered together. Typically crowds are defined by a common sense of purpose or emotions, such as a crowd of spectators at an event (e.g. a concert or football game) or people walking around and shopping in a mall or busy street (Figures 1.1a and 1.1b). Crowds are everywhere, they can be planned as in the case of crowds in a football game or flashmobs of people dancing or they can emerge such as pedestrians in a street going their ways. Crowds can be destructive or they can be constructive, they can be small or large, they might seem purposeful or completely random, they might be noisy or completely silent. They can be all or some of the above, but one thing is for sure; crowds are everywhere and they constitute a large part of what we know as our world, our environment, our lives and therefore their absence in virtual worlds we thrive on creating is a necessity for the believability of that world.

(a) (b)

Figure 1.1: (a)Two different crowds; a football team with its fans. One can consider both as a crowd with a common purpose; to enjoy and respect the game. (b) People moving around in a shopping mall.

During the past years, with the sudden explosion of computing power, the large dependence of the movie industry on Visual FX and the rise of Computer Games as the reigning king in entertainment, a large interest on creating more believable virtual worlds has risen. A large portion of what makes a world more realistic; aside from realistic lighting, rendering and plausibility is animation not just of individual characters but of large crowds. Additionally, the rise of robotics led to the design and implementation of autonomous robots which have been affected in their design either from nature (such as insect like flying robots or humanoid ones) or they are completely based on the application (e.g. assembly robots).

Crowds in nature Crowds are abundant in nature– even for animals without the cognitive capabilities of human beings, such as ants or bees which are highly social families of insects (Figure 1.2a). Starling birds exhibit incredible collective behaviour such as creating amazing formations without each individual bird having a sense of the whole (Figure 1.2b). This collective kind of behaviour might be considered as nature's way of balancing the limited cognitive capabilities of individual animals for the species common good.

Artificial Crowds Crowds can also consist of non living entities, such as a group of computers trying to solve a complex problem (e.g. unfolding the DNA sequence of the human species) or

Figure 1.2: (a) Ants are probably some of the most social insects on planet Earth; all ants in a colony socialize and collaborate on collecting food to transfer back to their base. (b) Starling birds.

swarms of robots collaborating to explore unknown areas (Figure 1.3a). Some of the robotic creatures that appeared have been heavily influenced by nature, such as Sony's AIBO dog like robot (Figure 1.3b) and Robot Quadrotors [41] (Figure 1.3c) that synchronize to play the theme of James Bond using real musical instruments.

1.2 Virtual Human Crowds

Due to their presence in our everyday lives, there has been a growing demand for simulating human crowds that populate virtual scenes. A virtual environment breaths and feels more alive when autonomous virtual entities, such as virtual humans and animals are added into it. Since it is currently not feasible to simulate even a single virtual human to absolute perfection (both in behaviour and appearance), simulation is different depending on the target application which may have completely different basic requirements; be it a movie or game, training simulation or evacuation scenario, interactive or not.

Movies Movies have benefited considerably from the addition of realistic virtual humans both to enhance the believability of a virtual scene such as city and to add virtual characters that interact

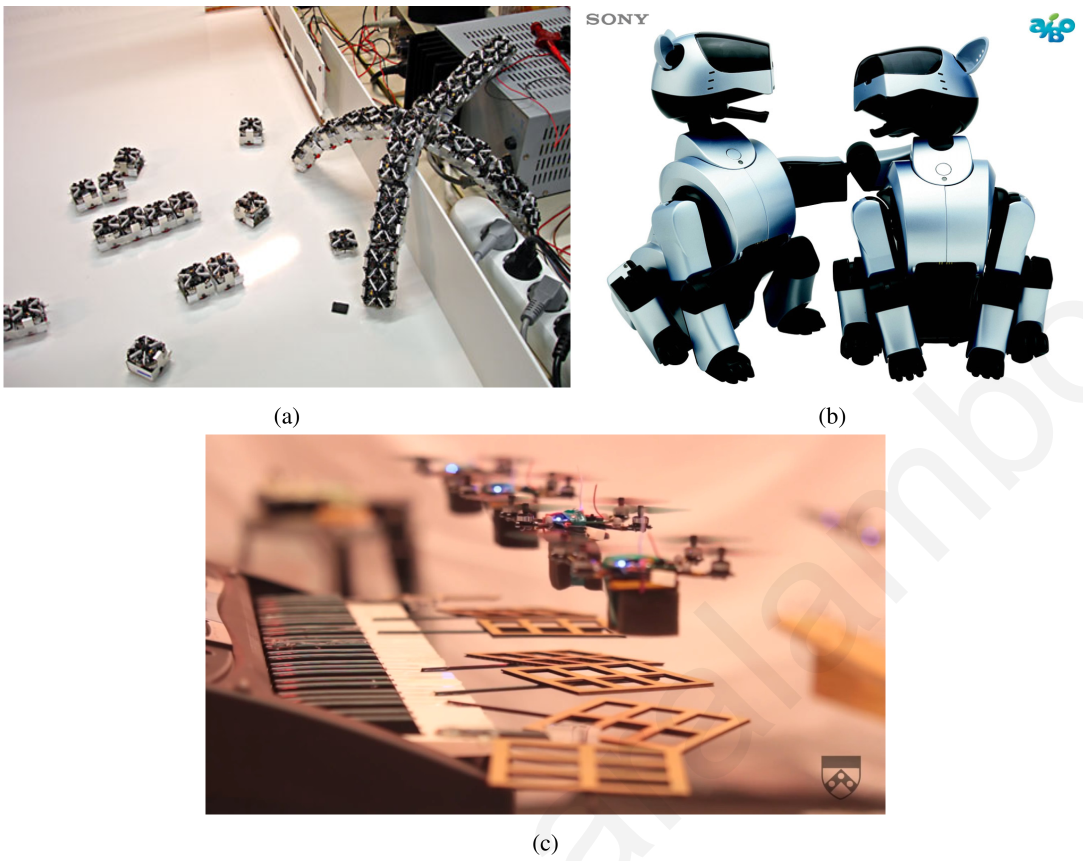


Figure 1.3: (a) A swarm of robots collaborating to escape from their prison (b) Sony's AIBO robots are inspired from nature. (c) A swarm of insect like robots collaborating to play the James Bond theme (video: `http://youtu.be/_sUeGC-8dyk`)

with the real actors. Classic examples of movies that benefited from the addition of virtual characters include the Lord of the Rings trilogy (Figure 1.4 top row), King Kong (Figure 1.4 bottom row) and World War Z (Figure 1.5). Crowds in these kind of movies are generated and added as a post-processing step (Figures 1.4–1.5) using specialized software such as Massive [57] which up to this moment is probably the de facto standard for realistic crowds. Most of the simulations developed for movies though are short term (i.e., they last a few seconds) and animators/directors are interested in having **control** on the generated simulation; spectators can notice this in the fast transitions between scenes or camera angles that are used as a way to concatenate the short

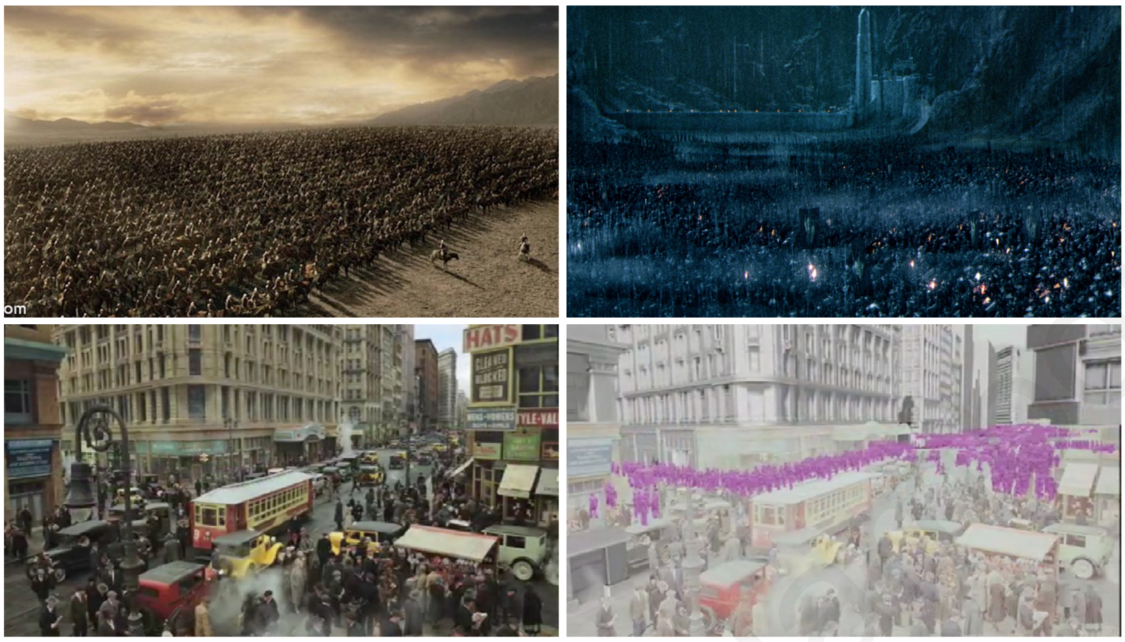


Figure 1.4: Crowd simulation is being used extensively in movie productions. Some crowds generated by Weta's Massive [57]. (top row) Massive crowds of competing armies of agents fight in the Lord of the Rings Trilogy (2001-2003). (bottom row) Background crowds from the movie King Kong (2005) are generated from a crowd simulator, whereas the front part of the crowd consists of real life actors.

clips together. Simulations from these kinds of systems are typically expensive to generate and practically difficult to use in a real time system such as games.

Games Nowadays, more and more games employ crowds to enhance the user experience and the virtual world (Figure 1.6). Depending on the type of game, crowd behaviour can differ. In a strategy based game for example, such as Battle for Middle Earth II virtual characters collaborate on defeating opponents whereas in games such as Assasin's Creed or Grand Theft Auto V they act as a means to increase the sense of believability and immersion of the player. Some games take the level of realism even further, by trying to emulate life of people such as the best selling The Sims games (Figure 1.6 bottom left). In The Sims series of games, virtual humans have different needs (like sleep, duty, socialization, hunger and amusement) and evaluate different *activity options* that will fulfill those needs using the concept of *Utility*; i.e., a representation of preferences over some goods and services.

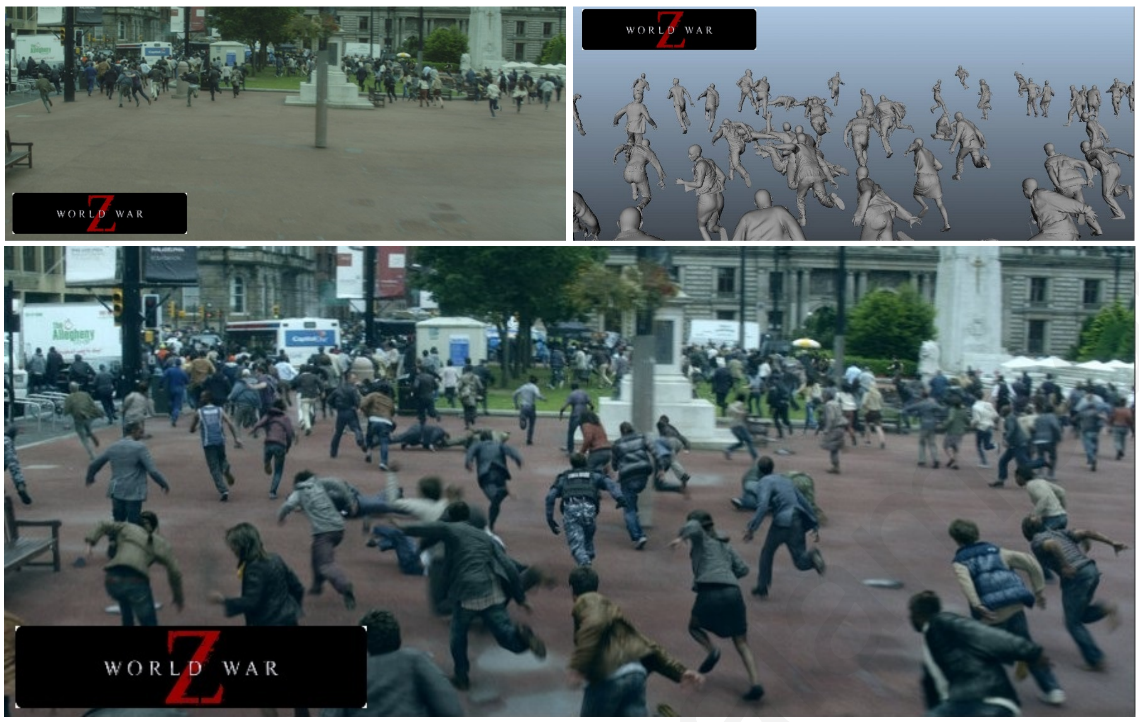


Figure 1.5: An example scene from the World War Z (2013) movie where virtual zombies were added in a post processing step to the scene with real actors.

Other But virtual humans can be simulated for other reasons also; and what really could be more interesting than understanding life itself? Biologists and cognitive scientists have long been interested in studying the collective bahaviour of people and animals and have developed a lot of models for doing so. In most of these models; scientists are mostly interested in the collective behaviours and not so much on individual behaviours and therefore these models employ either simplified individual rules or global ones.

One of the most important and widely used applications of crowd simulation is in safety modeling such as crowd evacuation scenarios to understand how buildings are to be designed. In these kinds of simulations, individual human behaviours are not so important as the global one, and therefore most of these simulations assume simple representations; both for behaviour and appearance.

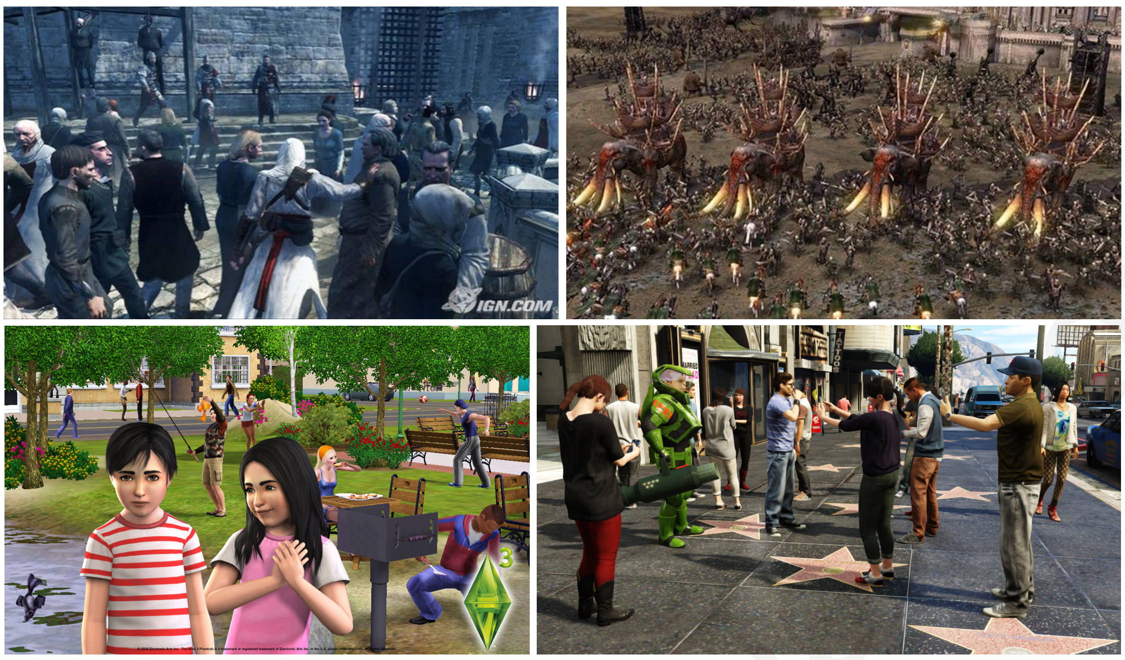


Figure 1.6: Recent advancements in computing power allow games to add multitude of computer controlled secondary characters that enhance the gameplay and the realism of the game. (top row) Screenshots from Electronic Arts Assasins Creed (2007), Battle for Middle Earth II (2006), (bottom row) The Sims 3 (2009) and Rockstars Grand Theft Auto V(2013) respectively. Each one has different crowd requirements, with the Sims games for example requiring smaller "intelligent" crowds whereas crowds in Assasin's Creed act as an enhancement to the scene.

Requirements Crowds have been studied for architectural visualizations, safety, urban modeling etc, with each field having different requirements. Requirements for performance, realism and control are in most cases conflicting and are highly dependant on the user of the system and the target applications. For example, movies should have realistic looking characters with believable behaviour and at the same time the designer needs to have full control on the output without necessarily the result be generated in real-time (except perhaps for previewing purposes). In safety modeling on the other hand, the basic requirement is the aggregate behaviour of all the virtual characters and not each individual characters' behaviour, giving less emphasis on the appearance of each individual character. An overview of the techniques for simulating crowds can be read in Chapter 2.

Complexity of a Crowd Simulator Crowd simulation can be thought of consisting of different layers with different researchers emphasizing on different aspects of it. Each layer can be thought of as a component that needs to be implemented in a simulator to capture the complexity of human beings (Figure 1.7):

- *Environment modeling* deals with modeling static geometry, obstacles, buildings, weather etc., anything that the virtual humans come across. Typically, the environment is annotated with behavioural annotations to indicate allowed behaviours and suggest actions to be performed.
- *Behaviour* is responsible for the modeling of crowd response to the environmental conditions (i.e., to the environment and other virtual characters) and it can range from individual to collective behaviour. This component is typically divided into more than one layers, depending on the required complexity and believability compared to real world crowds
  - *High Level Behaviour*: is responsible for the more goal oriented cognitive simulation of either individual characters or the whole crowd such as decision making and task scheduling
  - *Middle Level / Path Finding*: is commonly used so that virtual characters find paths to move from one point of interest to another and is some times merged in the High Level layer
  - *Low Level Steering / Navigation*: is responsible for moving the characters in the scene whilst they are trying to accomplish some goals (such as following a path) and (typically) at the same time trying to avoid any collisions with other characters or obstacles as realistically as possible.

- *Character Animation* that is responsible for animating the virtual characters. Typical animations for a crowd simulation system are walking, talking, waving, social interactions etc.
- *Rendering / Appearance* is responsible for displaying the virtual characters and their actions in their environment. Depending on the crowd size and the application requirements, rendering can range from something as simple as a dot per character or something as complicated as a dedicated rendering engine capable of displaying realistic human characters.

Depending on the application; be it crowds for a movie or an evacuation scenario, some or all of the above mentioned components might need to be implemented. Usually these components are not completely independent since there should be some kind of data exchange between them. For example, the behaviour component is responsible for decision making and in order to do so it needs data from both the environment and behaviour components. The animation and rendering components are responsible mostly for the appearance of the crowd; i.e., how each individual virtual characters is rendered and animated. In more advanced crowd simulation systems there is some data exchange between the animation and behaviour components so that depending on the available animations of different characters different actions should be taken and vice versa; depending on the actions the appropriate animations should be supported. If for example a virtual character is not equipped with a running animation its movement speed cannot be over the walking speed or if a character is equipped with a side stepping animation, different angular changes at steering can be triggered such as a sudden 90 degree change in movement.

Crowd Authoring In addition to simulating all these layers, tools that help authoring crowd simulation scenarios such as [12, 18, 20, 44, 50, 57] are typically developed. Most of these tools should be able to implement all of the above componentes; or at least integrate into other tools

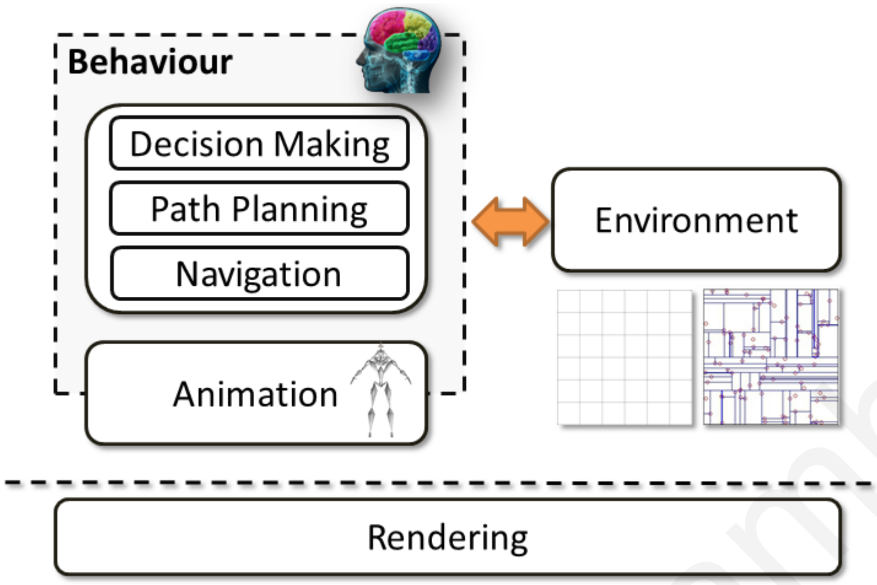


Figure 1.7: A typical crowd simulator for virtual humans has to deal with most of these components depending on target applications.

(such as rendering with Autodesk's Maya [58]) and most importantly be flexible, easy, fast and efficient all at the same time.

Crowd Analysis and Evaluation One often overlooked aspect of synthesizing virtual crowds is their evaluation. Human beings are accustomed to real life crowds in their everyday lives and can identify abnormal crowd behaviours easily when these occur. In a real-time simulation though; such as the ones in computer games it is quite difficult and expensive to capture all possible problematic observations by hand and therefore sophisticated approaches need to be applied.

1.3 Problem statement / Motivation/ Contributions

As part of this thesis, data-driven methods to simulate, evaluate and extract higher level understanding of crowds are proposed. Over the past years, a multitude of approaches have been proposed to address these issues with most of them relying on empirical rules and observations; typically a set of rules are used to empirically and explicitly define the behaviour of crowds for simulation purposes. Rules are also used to evaluate crowds such as for example the principle of least effort that defines that people prefer the fastest route to accomplish a goal.

Recently, data-driven techniques have been proposed as alternatives to extract these rules implicitly from data of either real life crowds or well behaved simulations. By using data, complex rules can be implicitly derived that can capture the subtle behaviour variation inherent in real world crowds. Since data describe correct (or desired) behaviours implicitly, evaluation can also be performed. This thesis addresses some of the limitations of data-driven approaches such low performance and quality of simulations and almost non existent methods for evaluation. More specifically, a new data-drive framework for simulation is proposed (the PAG) that displays large performance gains(typically more than an order of magnitude of performance gains can be observed) over traditional approaches with good collision quality [7]. Also, a data-driven framework based on multi-criteria anomaly detection is presented to evaluate both real and simulated crowds [9]. Finally, a method to annotate crowd behaviours with high level descriptors is presented that can give more intuitive descriptions of crowds [6]. A short description of the contributions over traditional approaches is presented over the next few paragraphs.

1.3.1 Data-Driven Crowd Simulation

Data-driven approaches for simulating virtual crowds have been recently proposed as alternatives to manually defining the rules that govern crowd behaviour. By using data of real crowds, the burden of manually defining very complex rules that describe desired crowd behaviours is removed. The main issue with these approaches is performance, since typically at each simulation frame expensive queries have to be applied to retrieve the best possible actions to be performed by the virtual humans.

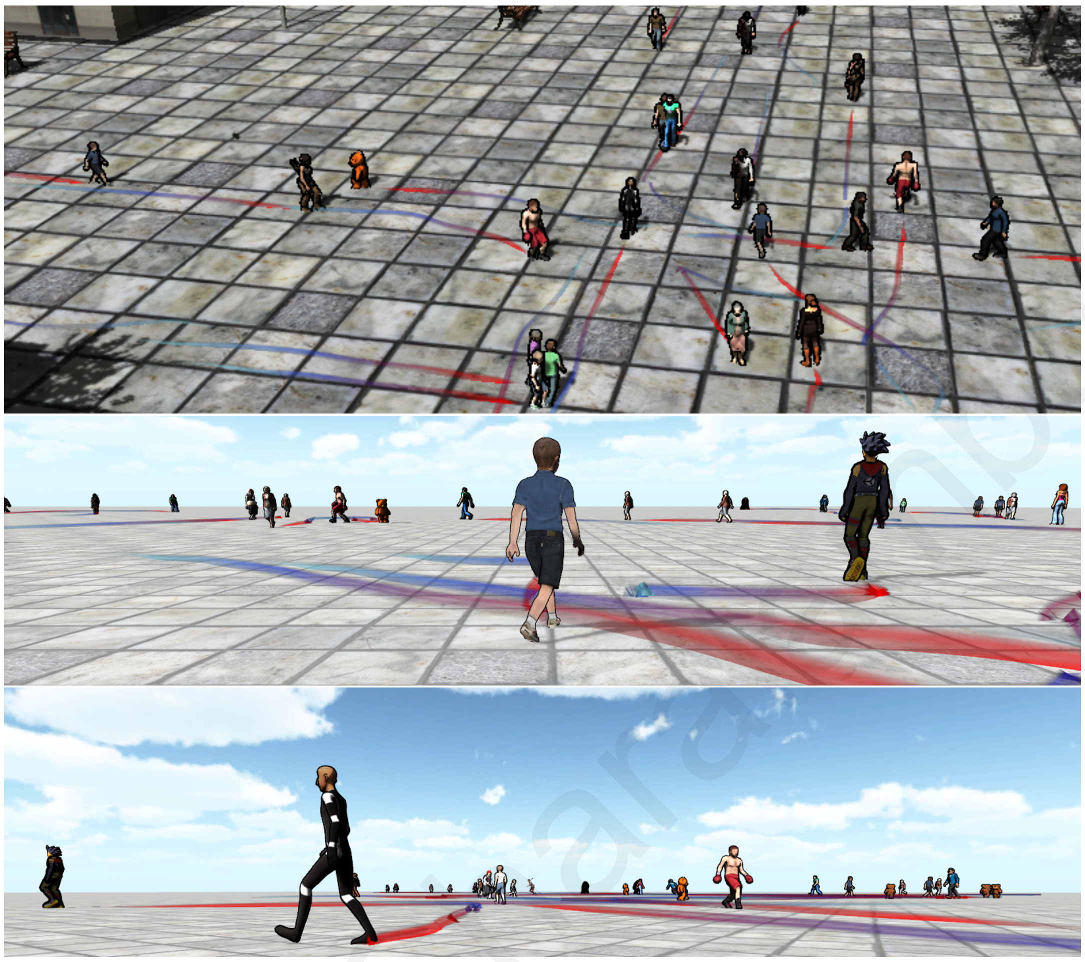


Figure 1.8: **Example Simulations employing the PAG** Example renderings of crowd simulations generated by the PAG data-driven crowd simulation framework that is proposed in this thesis.[2]

We propose a data-driven framework for crowd simulation, and more specifically crowd steering [7]. The proposed method interlinks examples of crowd behaviour into a structure called the Perception–Action Graph (PAG) which can be used at run-time to efficiently synthesize believable virtual crowds. The given examples encodes a character's temporal state from tracked data of real or simulated data at any given moment in time.

Nodes of the proposed graph structure represent groups of similar states (the agent's perception) whereas the graph's edges indicate transformations and actions (trajectory segments)) that

[2]Some of the 3D models of the characters were found for free or bought from various 3D libraries on the Internet such as TurboSquid and the Unity Asset Store whereas the buildings were bought from the Unity Asset Store.

lead from one state to another. The proposed method has been tested in various scenarios using a number of different input data and was compared against the nearest neighbors alternative approach which is used in most crowd data driven approaches. The results indicate performance improvements of an order of magnitude with similar and most of the times better quality – assuming that enough data is given as input to the simulator.

Publications A number of publications related to this work has been published [4,5,32] and/or are currently under print [7].

1.3.2 Crowd Evaluation

Evaluating crowd simulations if a difficult problem that has not been studied yet extensively enough in the literature. The main issue with evaluation is that deciding on what is a correct simulation is most of the times biased that depends on the point of view of the evaluator. Typically three approaches are followed to evaluate crowds: user based, statistical and data-driven approaches. User based evaluations are a common approach due to an important factor; human beings are accustomed to real life crowds in their everyday lives and in the end they are going to be the final users/viewers of the system. This kind of evaluations though pose some problems due to a user's bias to the real world. If the task for example is evaluating the crowd navigation of characters and the characters do not have human like appearance with human like animations, users might not rate the navigation quality of the simulation correctly due to their real life biases to real world appearance. Statistical approaches on the other hand, try to measure some statistics and data such as speed histograms of people, interpersonal distances or distance to goal and rate a simulation's quality in different scenarios such as the SteerBench suite [93]. This approach is still problematic though, since statistics are note linked to real world experience and the ranking could be objective to the selection of statistics. This is where data–driven techniques come in place, where user based experience (or real world data to be precise) are linked to the statistics.

A new data-driven crowd analysis framework is introduced [9], that formulates the problem of crowd evaluation as anomaly detection and applies multi-criteria anomaly detection methods and more specifically kLPE [110] and Pareto Depth Analysis (PDA) [29], a state-of-the-art anomaly detection technique from the field of machine learning. PDA is uniquely appropriate for the crowd analysis problem due to its ability to handle multiple criteria in a robust way. The main advantages of the proposed approach are:

- It enables unsupervised anomaly detection, i.e., there is no requirement on manually labeling the reference data.
- By comparing a simulation against a known dataset both atypical behaviours as well as missing features from the simulation (Novelty Detection) can be identified.
- Given a simulation consisting of mostly nominal behaviours, atypical behaviours can be identified by using the simulation itself as reference (i.e., in essence Outlier Detection).
- Through combining individual and social evaluation criteria, it allows the identification of issues related to the social behaviours of the agents, which are normally much harder to detect.
- Through the introduction of randomized PDA, a variation of PDA, the analysis of crowd simulations is enabled at interactive rates.

Publications This work is currently under preparation and is expected to be submitted at a Computer Graphics journal [9].

1.3.3 Crowd Behaviour Annotation

Being able to extract higher level understanding of pedestrians from their trajectories can have a wide array of applications such as crowd synthesis, crowd evaluation and outlier detection. Having a method for automatic labelling of trajectories, or parts of them, with behaviour tags such as walking, standing still, talking to another person or as being part of a group, real and virtual crowds can be better understood, evaluated and simulated.

As far as crowd synthesis is concerned, trajectories from large databases could be extracted so that they exhibit a particular behaviour pattern and then used to train a data driven crowd simulator resulting in more believable crowds. Typical methods to evaluate a virtual crowd consist of quantitative approaches such as gathering statistics from the simulated trajectories and getting scores for various tasks such as time to reaching goals, minimum time to collision, average speeds etc. or qualitative approaches that employ typically user evaluations. By comparing higher level characteristics of the simulation and to real world data, we believe that better evaluation algorithms can be developed.

In this thesis, a set of features that help in high level classification of trajectories is proposed based on both local and global characteristics of trajectories. This selection of the two type of features is capable of capturing behaviours that only one type of them would fail due to both local interactions between pedestrians and global state. By training a Random Forest Classifier, we display high accuracy in classification of new data; both simulated and real.

Publications This work has been presented at a conference [6] and is currently being extended.

1.4 Thesis Outline

In Chapter 2, relative work to this thesis on crowd simulation, evaluation and analysis are presented, while in Chapter 3 required knowledge to generate and analyse agent based crowds are presented. In Chapter 4, the Perception–Action Graph is presented as the proposed data–driven crowd simulation framework whereas in Chapter 5, our multi–criteria analysis framework for evaluating real and virtual crowds is presented Chapter 6 a simple approach for higher level annotation of crowd behaviours is presented and finally Chapter 7 presents possible future work and extensions to the work presented in this thesis.

Chapter 2

Related Work

“If I have seen further it is by standing on the shoulders of giants.”
Sir Isaac Newton

2.1 Virtual Crowds

As it was already mentioned in Chapter 1, a complete crowd simulation system should consider the following components (Figure 2.1):

- Human Behaviour, which can be divided into more layers depending on required complexity – from higher level goal oriented ones to lower level mechanical ones
- Virtual human animation
- Environment representation
- Rendering and appearance

In the remainder of this section we present literature work on the behaviour components of crowd simulation, which are the most relevant to our work. In Section 2.2 we we look at issues

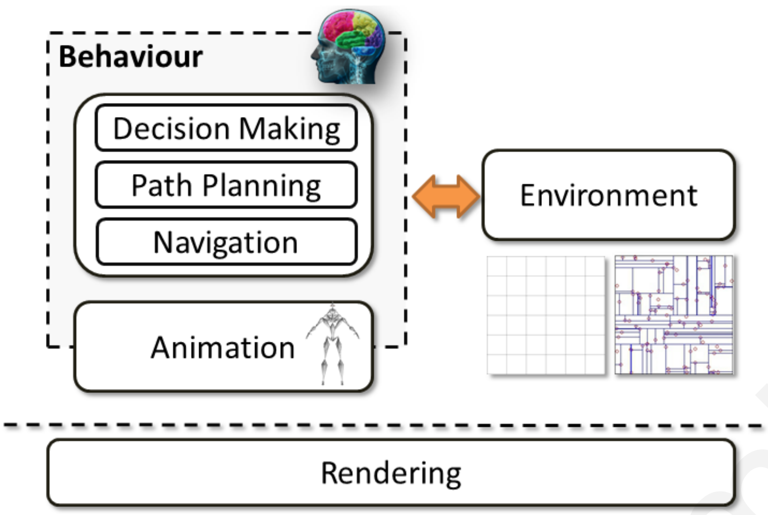


Figure 2.1: **Components of a Crowd Simulator** A typical crowd simulator has to deal with most of these components.

concerning the overall problem of crowd behavior whereas in Section 2.3 we give more emphasis on crowd steering. Since this thesis addresses the problem of simulating virtual humans, for the remainder of this chapter when discussing for crowd simulation we are implicitly referring to humans unless noted.

2.2 Crowd Behaviour

Typical examples of complete crowd behaviour systems are the Sims series of games (Figure 1.6), the Pennsylvania train station simulation by Shao et al. [89] and the HiDAC system by Pelechano et al. [72, 73].

A complete crowd simulation framework for relatively large crowds in a simulated version of the Pennsylvania Train Station, having an architecture similar to the approach just discussed was introduced by Shao et al. [89] (Figure 2.2). Similarly, the HiDAC system by Pelechano et al. [72] implements a two layered architecture for each individual agent having a high and low level component (Figure 2.3). Each individual character in the Sims series of games is modeled as an agent

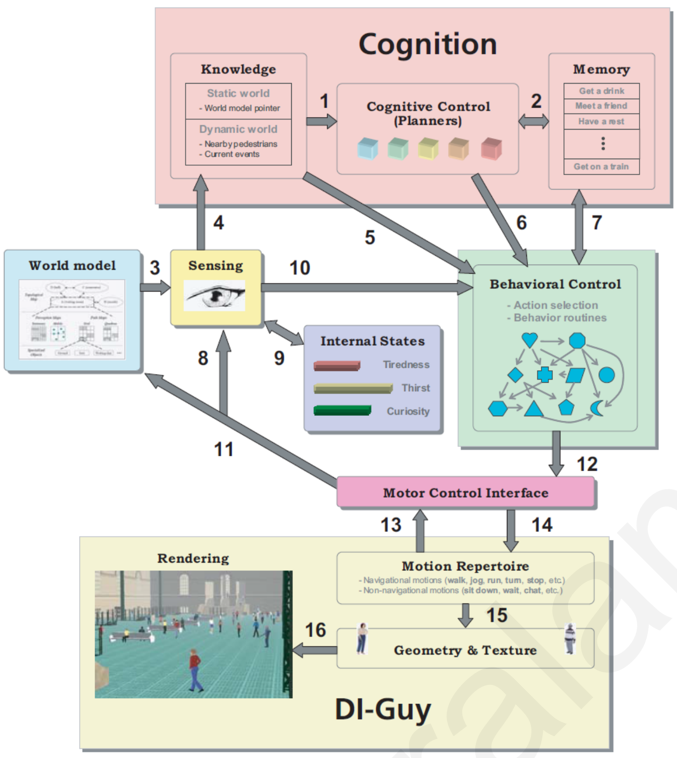


Figure 2.2: **Autonomous Pedestrians** Shao et al. [89] introduced a complete simulation framework for autonomous pedestrians in a relatively large crowd simulation of the Pennsylvania Train Station.

(the *Sim*) with certain physical characteristics (age, height, sex), current internal/psychological state (happy, sad, hungry, etc) and a set of goals. The main focus on these games is the higher level behaviour of the Sims and therefore they try to optimize a utility function based on these states [95] whereas local collision avoidance with other Sims is not implemented.

2.2.1 High Level Behaviour

High level behaviour is responsible for giving purpose and goals to individual characters. This component should be able to present enough variation to the characters to look and feel different but not to such an extent that this variation is unrealistic.

Loyall et al. [56] give individual characters goals and priorities to enrich virtual characters with distinct, recognizable personalities, whereas Perlin et al. [76] uses scripting to do so. Funge

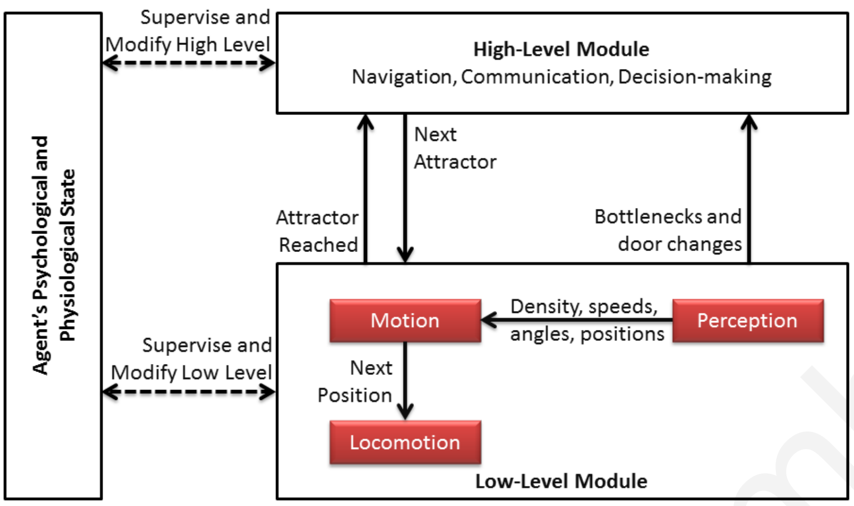

Agent's Psychological and Physiological State
Supervise and Modify High Level
High-Level Module
Navigation, Communication, Decision-making
Next Attractor
Attractor Reached
Bottlenecks and door changes
Supervise and Modify Low Level
Motion
Density, speeds, angles, positions
Perception
Next Position
Locomotion
Low-Level Module

Figure 2.3: **HiDAC** Behaviour in the *HiDAC* system consists of two layers; high and low. Figure adapted from [72].

et al. [17] proposed a cognitive model over virtual humans it is defined what a character knows, how this knowledge is acquired and how it is used to plan actions for the character. This model was later used to give purpose to individual characters in a train station [89]. Yu et al. [108] use hierarchical decision networks to simulate the social interactions between pedestrians in urban settings. More recently, Shoulson et al. [92] introduced ADAPT that helps in defining character high level behaviours using parametrized behaviour trees.

2.2.2 Path Planning

Path planning is the process by which autonomous characters find collision free paths from one point of the scene to the other. To do so, the characters should be equipped with either local of global knowledge of the scene. This differentiation between local and global knowledge can affect both the simulation and in the case of interactive applications such as games or VR installations, the experience felt by the user.

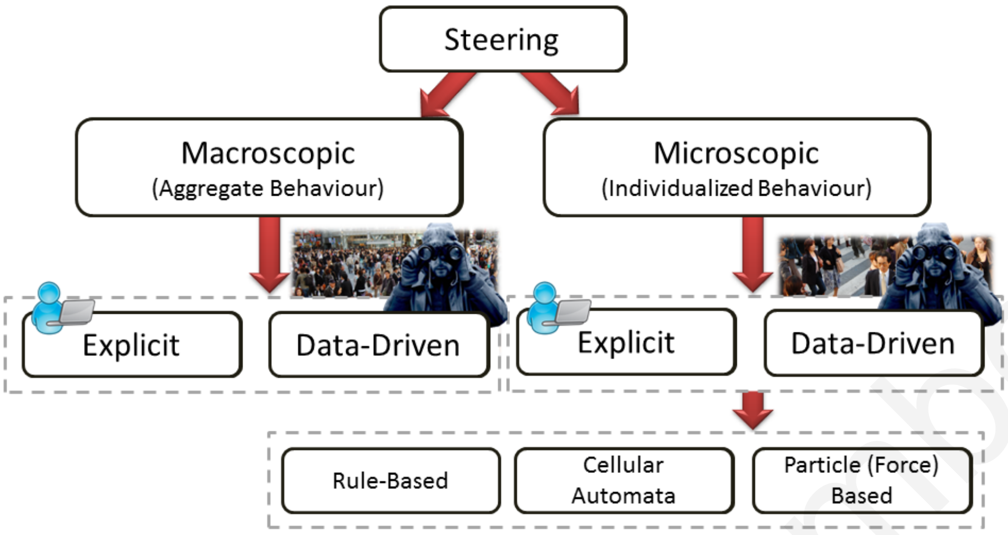


Figure 2.4: **Crowd Steering approaches** can be divided into two major categories: macroscopic that deal with the crowd as a whole and microscopic which give more emphasis on individualized behaviours. Both can be implemented either though giving explicitly rules or implicitly from examples of the required behaviour.

Popular approaches for path planning include A^* [23] which is an extension to Dijkstra's shortest path algorithm [13] that uses heuristics on the paths, potential fields [91, 106] and probabilistic roadmaps [37]. Pettré et al. [77] employ navigation meshes calculated over the geometry of the scene for real-time rendering and simulation of crowds whereas Geraerts and Overmars [19] introduced the concept of path planning using corridor maps.

2.3 Crowd Steering

Crowd steering is the process by which virtual characters move in their environment between intermediate goals (i.e., in the most simple form it could be "move on path $\mathbf{P}$ towards point $\mathbf{A}$") and is responsible for handling interactions between characters and the environment. Over the years, a large number of different methods have been proposed in the literature taking widely different approaches, which can be divided into two major categories (Figure 2.4): macroscopic and microscopic presented in Sections 2.3.1 and 2.3.2 respectively. In both of these categories, we

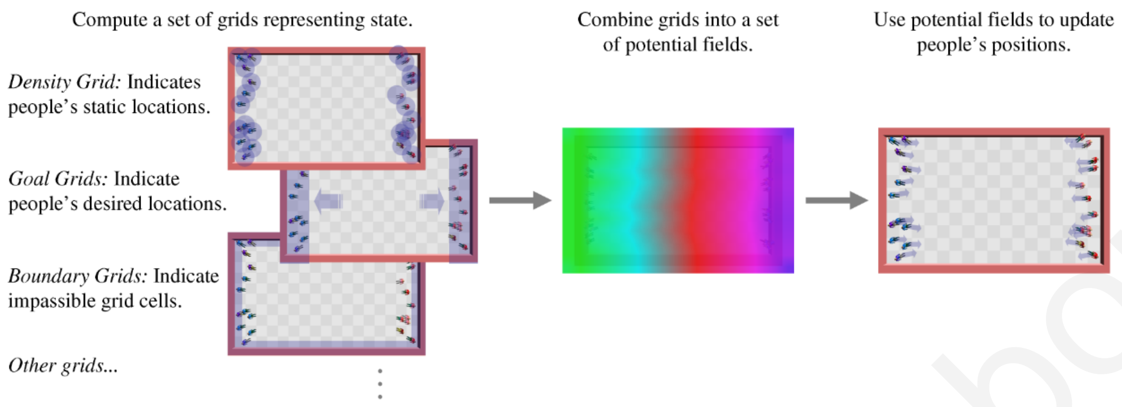

Compute a set of grids representing state.
Combine grids into a set of potential fields.
Use potential fields to update people's positions.
Density Grid: Indicates people's static locations.
Goal Grids: Indicate people's desired locations.
Boundary Grids: Indicate impassible grid cells.
Other grids...

Figure 2.5: **Continuum Crowds** The Continuum Crowds algorithm from [99]; a dynamic potential field is calculated based on various goals, dynamic obstacles, etc. which is then used to update virtual people's positions.

can find models that are either explicitly defined by an expert or implicitly defined by some form of data such as videos of real world people.

2.3.1 Macroscopic Simulations

2.3.1.1 Explicit Models for Macroscopic Steering

Macroscopic crowd simulation methods aim to simulate the crowd as a whole with a greater emphasis on the global appearance; individual character behaviour is usually not required and is of little importance. One common approach in these methods is the use of velocity or force fields to guide the characters in a collision free manner. Approaches with this formulation are the tiled velocity field approach of Chenney [10] and the dynamic potential fields of Treuille et al. [99] and Hughes [31]. The fields in [10] are typically predefined by a designer whereas the ones in [99] change over time depending on goals and the dynamic positions of other characters (Figure 2.5).

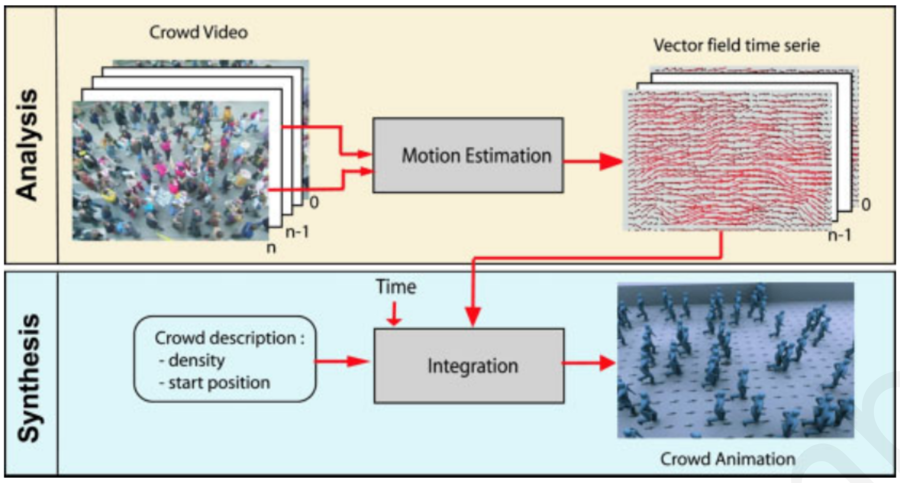


Figure 2.6: Vector field learning from Courty et al. [11].

2.3.1.2 Data-Driven Models for Macroscopic Steering

Courty et al. [11] proposed a data driven approach to learn such a vector field from videos of crowds; frames from the source video of crowds are extracted and a vector field is estimated finding inter-frame changes (see Figure 2.6). Crowds are then synthesized by finding and creating transitions between vector fields, similarly to the work done by Schodl et al. [86] and Kwatra et al. in [42] for videos and by Kovar et al. [39] for single character animations.

2.3.1.3 Some comments on Macroscopic Models

The main advantage of macroscopic models is the capability of simulating large numbers of virtual characters at minimum cost and therefore are ideal for cases where large environments are to be considered. Due to their nature, i.e., they do not take into account high level goals for the individual characters (with the exception of the Continuum Crowds [99] approach where they are taken into account during the update of the dynamic potential fields), there is not much diversity in the virtual humans behaviour which makes them look less realistic when compared to real world humans. Narain et al. [64] introduced a hybrid approach where global behaviour is defined similarly to Treuille et al. [99] and local collisions are handled using a microscopic (geometric) approach to enhance the believability of local behaviour.

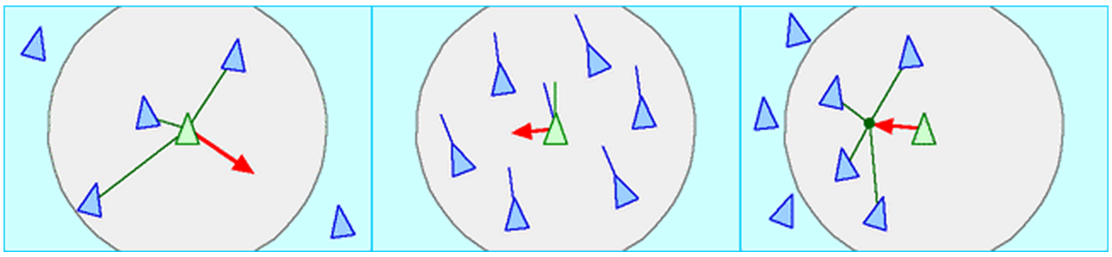
If there is a need for smaller scale, more believable and divergent crowds, microscopic models are more appropriate.

2.3.2 Microscopic Simulation

Microscopic crowd simulation approaches model each individual character in the scene as an entity; usually following agent based or particle based approaches. The main idea behind microscopic models is that complex believable global behaviour patterns can emerge from simple local rules. Some of the early works, such as the seminal work by Reynolds [81] and many that followed thereafter [46, 82, 89] employ manually defined rule-based approaches. Others make use of particles and social forces [25, 26]. Lately, there has been an interest in data driven approaches [34, 48, 51, 66, 68, 78] where data from real crowds such as people walking in a mall or people in controlled environments are used to implicitly extract rules automatically [48, 51] or are used to fine tune other simulators (such as the work by Ondřej et al. [66] and Paris et al. [68]). Data driven algorithms have also been used to learn flows of crowds [11] (see Section 2.3.1.2).

2.3.2.1 Explicit Models of Rule Based Crowd Steering

In the work of Reynolds on simulating flocks of birds [81], it was proposed that global flocking phenomena can emerge by using simple local reactive rules per individual member of the flock (the *boids*). He proposed using three basic rules: separation, cohesion and alignment (Figure 2.7). Birds that belong in a flock sense nearby flock mates in a small circular area around them and try to separate from them so that they do not collide, align to their moving direction and also try to stay near their center of mass so that the flock doesn't separate. Additionally, force fields around



(a) Separation (b) Alignment (c) Cohesion

Figure 2.7: **Boids** A small number of local reactive rules to simulate the behaviour of flocks of birds from [81]. Birds sense nearby flock mates in a circular area around them and try to (a) separate from them so that they do not collide, (b) align to the moving direction of the local flock and (c) try to stay near the center of mass of the neighbors so that the flock doesn't separate. Images downloaded from `http://www.red3d.com/cwr/boids/`.

(a) Bird Flock (b) Stanley and Stella (c) Schools of fish

Figure 2.8: **Flocking Simulations** Typical simulations generated using the boids model (Figure 2.7) In (a) we can see a typical bird flock, in (b) the *Stanley and Stella in Breaking the Ice*, the first 3D animation using Reynolds boid model from SIGGRAPH 1987 and in (c) simulated schools of fish from Reynolds [83].

static obstacles were added to add avoidance of static obstacles. This approach demonstrated that indeed complex flocking behaviour can emerge from a small set of simple rules and it was used to successfully simulate flocks and herds of animals for various movies such as *Batman Returns* and *The Lion King*. Reynolds later extended his model with additional steering rules [82, 83] to simulate more complex characters such as pedestrians, cars and also mechanisms to help simulate higher level behaviour such as leader following, running away from predators, etc. In Figure 2.8 some images from flocking and herding behaviour using Reynolds models can be seen.

Van den Berg et al. [105] introduced a geometrical approach to agent based simulation; the Reciprocal Velocity Obstacles (RVO). The main idea behind this work is that agents calculate collision free velocities using as input the current positions, velocities and shapes of other agents and obstacles. Guy et al. [21] proposed to use the *principal of least effort (PLE)* (which states that humans prefer to use as little effort as possible to achieve any goals) to calculate energy-efficient trajectories for individual heterogeneous agents using the RVO framework.

Helbing and Molnár [26] introduced the *social force models* approach, where each individual in the crowd is considered a particle with a given mass, and repulsive/attracting forces are applied to guide them in the environment. These forces are a measure of the internal motivations of each individual in the crowd to perform certain movements. To eliminate some of the limitations of the social forces model, Pelechano et al. [71, 72] introduced the HiDAC (High-Density Autonomous Crowds) system for navigating autonomous heterogeneous agents with different personalities using physiological, psychological and geometrical rules – i.e., particles have "soul".

This rule based approach proved to be very popular and it was employed by many researchers and commercial products such as the Massive Prime crowd simulation tool [57]. Although some of these methods are quite scalable [89, 96, 97] and can simulate, contrary to macroscopic approaches, individualized behaviours they cannot fully capture the range or the subtleties of individual behaviours in real life. Real life people react differently in similar situations; some times they make unexplained decisions such as running into each other or they might stop and start talking instead of just avoiding each other. These behaviours, even though they look and feel natural to human observers, are difficult to replicate and capture. The only way to achieve such behaviours is to implement such a complex system that allows the designer to define a large number of finely tuned rules [12, 57] and character parameters, which is quite a laborious and especially difficult task to do by a non expert.

One way to remove this burden from a designer is to introduce data-driven techniques that try to extract individual agent behaviours from examples of real people.

2.3.2.2 Data-Driven Crowd Steering

Data-driven techniques have been extensively used in many areas of computer graphics. For example, many recent texture synthesis techniques are able to synthesize large textures from small examples [42] and fill in holes in images [14]. The image analogies approach uses examples to learn about and reproduce relationships between image pairs [27]. Additional applications include surface completion [90], image colorization [33], image segmentation [85]. Zhang et al. [109], use an idea similar to the motion graph for simulating fire by transitioning between already simulated fire particles.

Recently, data-driven crowd simulation methods have emerged as an attractive alternative to defining crowd rules directly. These techniques derive the rules governing a crowd indirectly from source data such as videos of crowds [34, 48, 51], sketching of trajectories by users [98] and also synthetic data [45, 48] and so in essence these techniques try to reverse engineer and imitate the complex behaviour of people and animals. The promise of using these kind of systems is that agents will "learn" how to behave from real-world examples, keeping the natural crowd ambiance with a wide range of complex individual behaviours without the effort of defining an explicit behavioural model.

One of the earliest data-driven crowds [45] used a motion graph approach for synthesizing group behaviour. In this approach, data from an existing rule based system that simulates flocks of birds [81] were used to create a tractable motion graph that is used at runtime to generate a new flock with similar behaviour to the input. This method makes the assumption that the input examples follow a well defined behaviour model, such as a flocking system with a restricted configuration space (i.e., a constant number of birds) and therefore this approach used in the context of a complex crowd simulator such as an inhabited city is not possible. Graph based

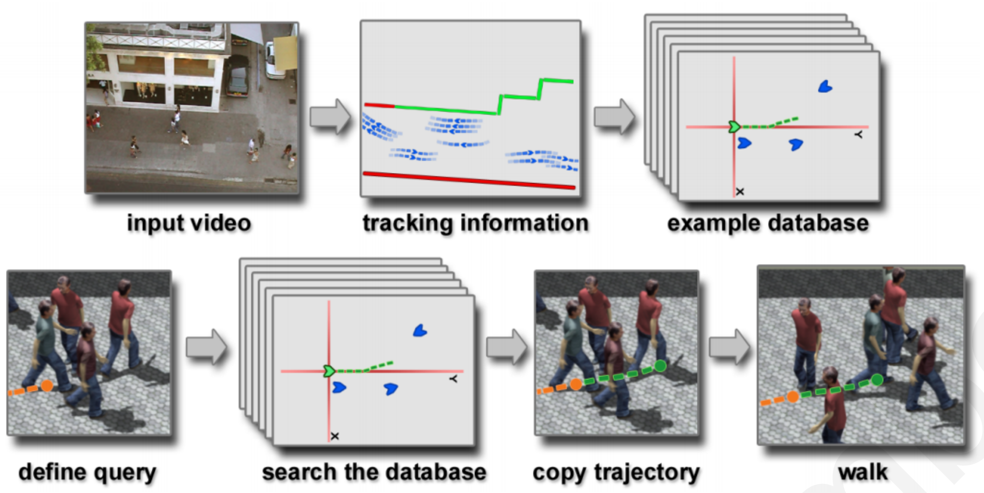
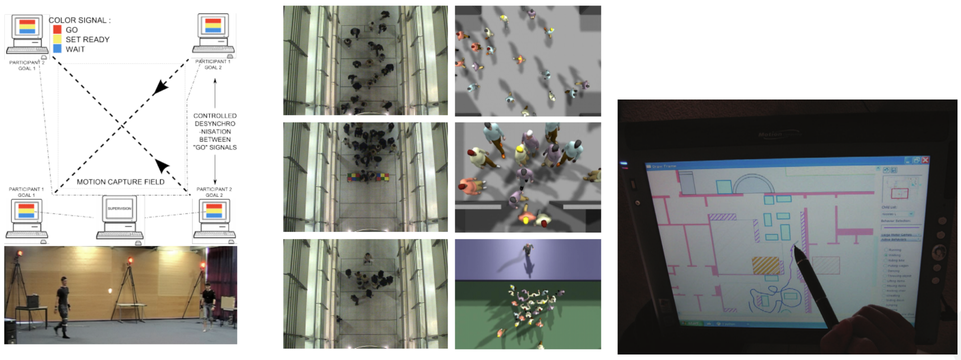


Figure 2.9: The data driven framework from Lerner et al. [51]

simulation was also used in [43] for guiding a single group of agents navigating together, but again the limitation that these agents should move in groups and the great variation in the movements of a general human pedestrian crowd would render these methods impractical.

Databases of examples In fairly recent works by Lee et al. [48] and Lerner et al. [51], trajectories learned from videos of real life crowds are stored in databases alongside some representation of the stimuli that affected their steering behaviours (Figures 2.9 and 2.10b). During simulation time agents match their current stimuli to the ones stored in the database, and navigate accordingly. Following from these Lerner et al. [53] used the database in order to add secondary actions to the characters such as talking to each other, waving etc. Ju et al. [34] take input data that represent different styles of crowds and blend inbetweenthem to generate new collision free crowd animations (Figure 2.11). Metoyer at al. [60] allow the user to define specific examples of behaviours, while Musse et al. [63] extract paths from a video for a specific environment.

Parameter Estimation Several works [68, 78, 102] (Figure 2.10a) use motion capture data in a controlled environment to estimate collision avoidance and anticipation parameters and proposed prediction based approaches to crowd steering. Moussaïd et al. [61, 62] used data from videos of



(a) (b) (c)

Figure 2.10: **Data-driven crowd simulation frameworks.** (a) Controlled tracking of people from [68]. (b) Captured and simulated data from [48]. (c) Sketching interface to define input trajectories from [98].

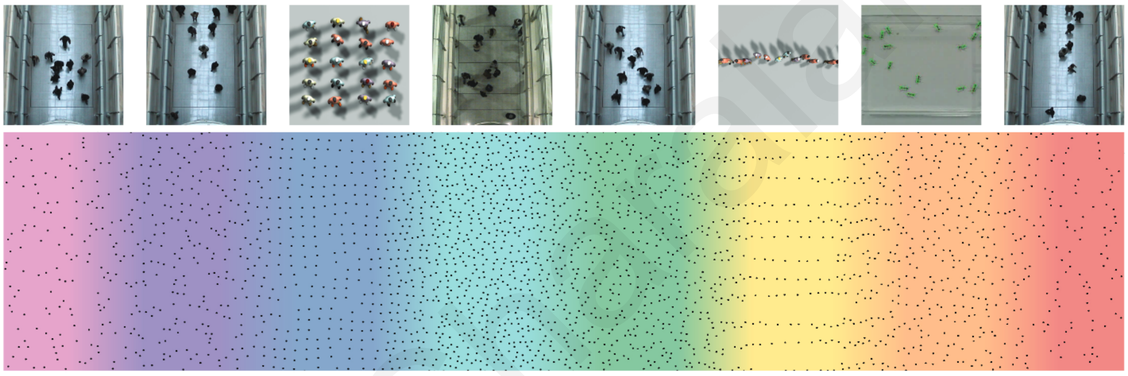


Figure 2.11: **Morphable Crowds** Ju et al. [34] introduced the idea of morphing between existing crowd motions to generate new larger collision free crowds.

real crowds to modify Helbing's social forces model [26], looking in particular on how to treat group formations in a more realistic way. Karamouzas et al. [36], used empirical data of crowds from [62] and their distributions to simulate and evaluate social groups in crowds.

Looking a bit further away, Biology researchers [28] proposed using input from stereoscopic videos of Starling birds to estimate a statistical model of their massive and complex flocking behaviour. In all the above cases the examples are used to refine an underlying behaviour model therefore they are still bound by the limitations of the model.

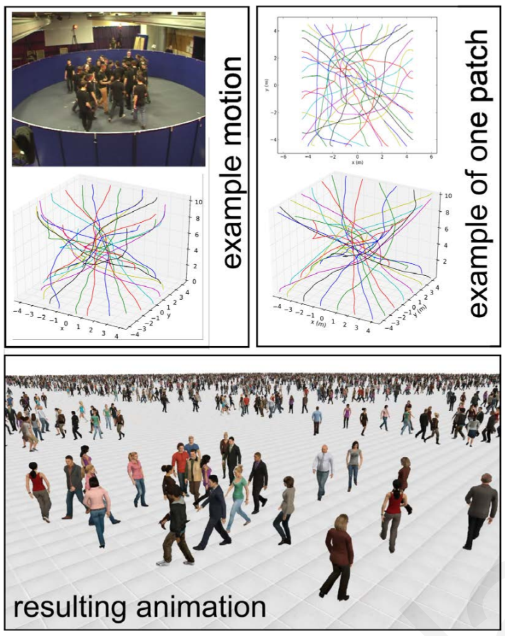


Figure 2.12: Li et al. [54] proposed recording small crowds in a controlled environments and then generating larger groups of people using crowd patches.

Tiling Motions Other techniques, such as the one recently proposed by Li et al. [54] record small crowds in a controlled environment and then generate larger crowds by concatenating smaller patches of crowds finding proper transitions between them (Figure 2.12). Similar works by Lee et al. [49], Yersin et al. [107] and Kim et al. [38] concatenate crowd motions stored in patches.

Very relevant to the proposed methods in this thesis are the data driven methods used in animation and motion synthesis for single character motion synthesis [30, 40, 55, 70, 80] and especially Motion Graphs [40].

2.4 Evaluation and Analysis

There are three general approaches to crowd evaluation: user/perceptual studies, statistical measures and data-driven approaches.

2.4.1 User Based Evaluation

Almost every work on crowd simulation has some qualitative evaluation of the simulation based on videos or images of the simulations. After all, the final judges of crowd realism will be human beings who live amongst crowds all of their lives. There has been a lot of work on evaluating appearance of crowds such as the work by McDonnell et al. [59]. Pelechano et al. [74] use presence as a metric to validate simulated crowd behaviours.

The problem with users though, is that if there is no careful selection of what to show the user, he/she might be biased; if for example steering behaviours are to be evaluated and the designer adds virtual humans with bad animation quality the user might underestimate the performance of the algorithm and vice versa. In steering approaches for example, some simple primitives to represent the character might be enough. But this could also lead to misconceptions, since steering behaviours that look good as circles might not look as good when rendered as virtual humans since the movements might require them to make unusual poses. Therefore more quantitative approaches should be used.

2.4.2 Statistics Based Evaluation

The most typical way to assess the correctness of a simulation is by devising a number of meaningful evaluation metrics, such as the time required for the characters to reach their destinations or the average number of collisions. In the animation community, Reitsma and Pollard [79] defined task-based metrics for evaluating the quality of individual trajectories. Singh et al. [93] proposed a number of predefined test-case scenarios along with different quantitative metrics to objectively assess the steering behaviours of virtual characters (SteerBench framework). More recently, Guy et al. [22] proposed the entropy metric to estimate how closely a given simulation state matches real-world data. Kapadia et al. [35] introduced an approach to characterize the space of all possible scenarios that a simulated agent can encounter and proposed a set of metrics to capture the quality of a simulator in this space.

Assessing the quality of crowd steering using statistics could still be problematic; for example crowds in real life do not always try to find the most efficient route or take the least time to reach to a goal. This is due to the large variability of humans with different age, sex, size, preferences, culture or psychological state. Data-drive approaches try to solve some of these problems by comparing simulations with real world crowds or data from correct (or desired) simulations.

2.4.3 Data-Driven Evaluation

Lerner et al. [52] proposed a data-driven approach that determines how similar the behaviours/ trajectories of simulated entities are to the ones obtained from video footages of real crowds. This approach was extended in [36] to evaluate the behaviour of small pedestrian groups. Guy et al. [22] proposed an information theoretic approach that employs entropy for evaluating the behaviour of a crowd as a whole using data from real world crowds as reference.

Conceptually, the proposed framework from Chapter 5 seeks to capture the advantages of both the user-driven analysis presented in SteerBug and the data-driven driven evaluation of the data driven approaches presented in this paragraph in a robust fashion.

2.4.4 Automatic Behaviour Annotation

A lot of work has been done to for mining similar trajectories in spatial databases, but these methods are mostly concerned with the spatial distribution of trajectory points [84] and not local agent based approaches. Work on classifying trajectories has been extensively studied in surveillance systems to identify abnormal pedestrian trajectories [67] or classifying vehicle trajectories [16]. In [47], Lee et al. use a series of features for characterizing satellite captured vessel trajectories. This last approach bears some similarity to the approach presented in Chapter 6 in the sense that both of the approaches take into account local and global characteristics of the trajectories for classification, however both the classifiers and the application domain are very different.

2.5 Conclusion

Even though some agent based data-driven techniques for simulating virtual humans have shown realistic results, this comes with a large performance penalty due to continuous database searches at every simulation step. Additionally, decisions for each simulated agent are based on only their current state and therefore quality might suffer. In this thesis, methods to improve both performance and quality are studied and implemented (Chapter 4). Using data to evaluate and analyze crowds and individual agents are a very promising method to help develop better methods for simulation. Most of the proposed data-driven methods examine the crowd as a whole or each agent's entire trajectory. Additionally, these methods work in a particular environment. Methods to analyze agent centric temporal segments are proposed (Chapters 5 and 6). These methods manage to analyze, tag, rank and identify abnormal local agent behaviours.

Chapter 3

Framework

A simple agent based crowd simulation framework is presented here as a basis for the proposed data-driven crowd simulation system that is presented in Chapter 4. Firstly, an agent based model of a human being is presented, followed by a simple agent based crowd simulator. Finally, different state representations are presented that encode the information as this is sensed by the virtual agents.

3.1 Agent Based Crowd Simulation

As discussed in the previous chapters, one of the most popular approaches for simulating virtual characters is with modeling individual virtual characters as *Intelligent Agents* that collaborate to accomplish individually (or globally) assigned tasks. In order to understand what an intelligent agent is, we first describe a model for the human being that can be simplified so that autonomous virtual characters can be implemented on computational systems.

3.1.1 A Model for Humans

A *Human Being* can be considered as an autonomous biological entity (Figure 3.1) that lives in different environments (a house, streets or even the whole world), it perceives it using mainly five main senses (sight, hearing, taste, smell and touch) through sensors found on its body (eyes, ears, tongue, nose and hands/body), it processes information though its brain (mainly) and acts to achieve goals such as working, educating itself, exercising etc. Humans have other senses also such as the vestibular system that helps in balancing and having a sense of orientation or thermoception and chronoception that help on temperature and time perception respectively. Even more so, humans have an innate ability to learn (well…most of them at least) ; this means that their actions not only affect the environment and themselves at the given time, but they also adapt, they gain new knowledge to solve similar problems in different ways and they can also can affect the culture and history of their species. Different people react to the same stimuli in different ways due to different experiences, education, upbringing and physical characteristics.

Humans as information processors The amount of information processed by humans is enormous and cannot be possibly simulated (at least with current technology) to the full extent. Figure 3.2 shows an information theoretic approach of modeling the sensing/action model of humans by Tor Nørretranders [65]; humans sense large amounts of data ($\sim 11 GBytes/sec$) process them using a very efficient processor that is capable of identifying important data and acts by using its muscles. Out of these data, only a portion is processed ($< 1\%$) with each sense contributing different amounts of information. Most of the information comes from the optical system, with big contributions coming also from the acoustic and tactile sensors.

Notice that the optical information is at least 10 times more than the acoustic and 25 times more than the tactile ones, with the other senses providing even less ($\leq 1\%$). That doesn't mean

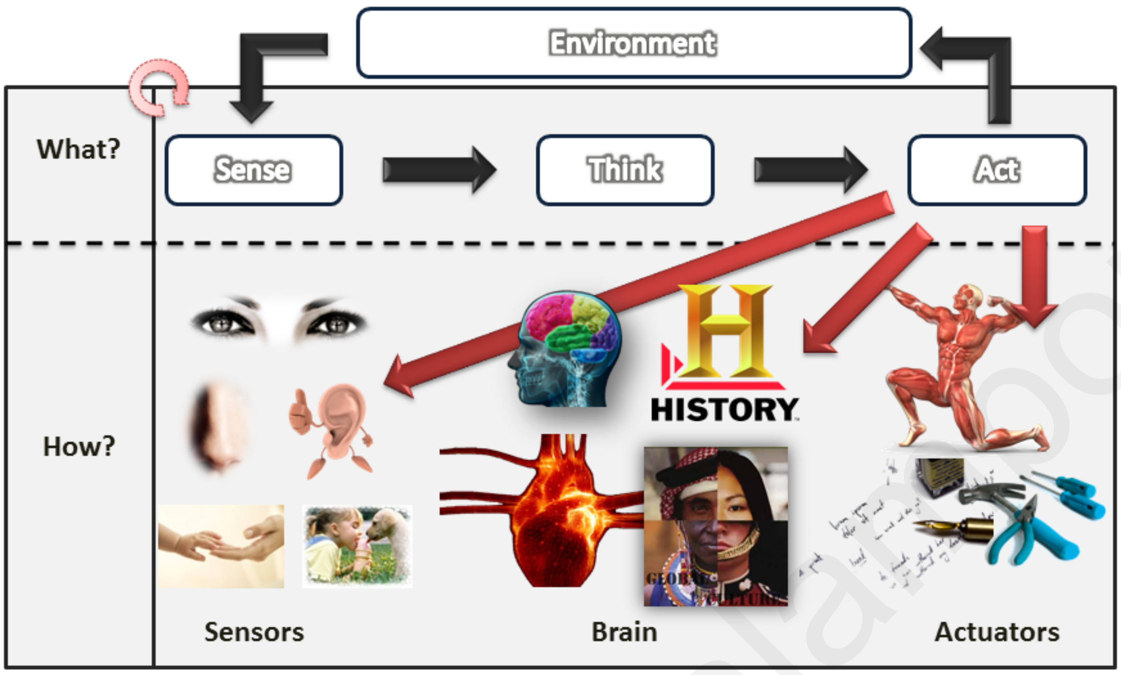


Figure 3.1: **Humans as agents in the real world** Humans are highly complex life forms with multitude of senses, and even more complex decision making mechanisms that are affected by brain, passion, culture and experiences. Their actions affect not just the environment they act in, but themselves also; they learn, adapt (both mentally and physically), they affect history and cultures and they do so throughout their entire lives.

of course that the data is of equal importance due to various reasons such as noisy data, the environment, coherence between time (sound for example with static imagery could be more important or some senses in people with inabilities such as blindness) and attention exhibited by the human. Humans process all this information many times per second using knowledge and experience they gained through their lifetime and act to express themselves using their muscles.

Muscle movement comes from 4 main components of the human body which more or less are of the same importance as far as muscle activation is concerned ($\sim 20 - 30\%$):

- Skeletal, which is mostly responsible for locomotion and expressive movements
- Facial, to display emotions and subtle messages
- Hands, for object manipulation and expressions

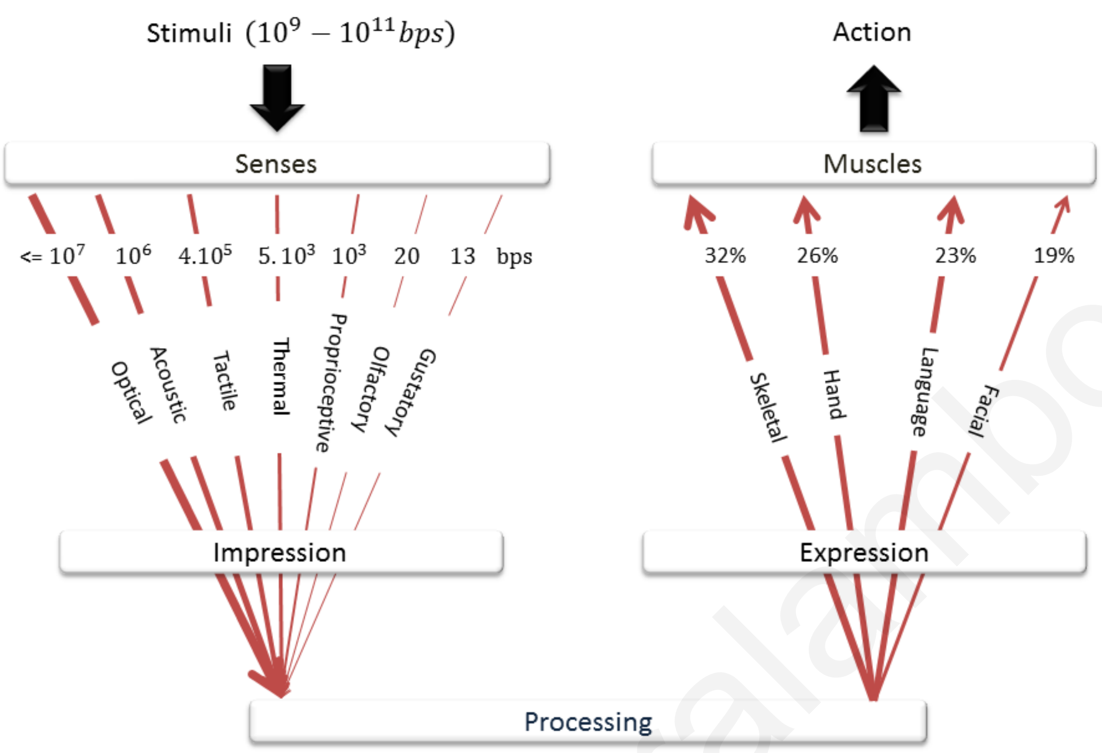


Figure 3.2: **Humans as information processors** An information theoretic model of the human senses; huge amounts of data flow through a human being every second ($\sim 11GBytes/sec$), with most of the data coming from the optical, acoustic and tactile sensory systems. A human being is capable of processing all these data very efficiently and take decisions that are mostly expressed through its muscular system; notice that even though the face is a small part of the human body it affects 19% of the muscular system. Image adapted from Tor Nørretranders [65].

- Language, for auditory expressions

This tells us, that even though some of them constitute a small part of the human body (such as the facial muscles), all of them are important and therefore if one wants to have proper simulation of humans; he/she should take into account all of these ways of expressing actions. Obviously, we are far away from the moment where we will be able to model (**IF ever** that is) such complexity of not just a single virtual character but multitudes that interact with each other. There are a lot of unknowns such as how do really people store, access and process previous knowledge to handle new situations. This leads us to a more simplified approach; similar to the one shown in Figure 3.1.

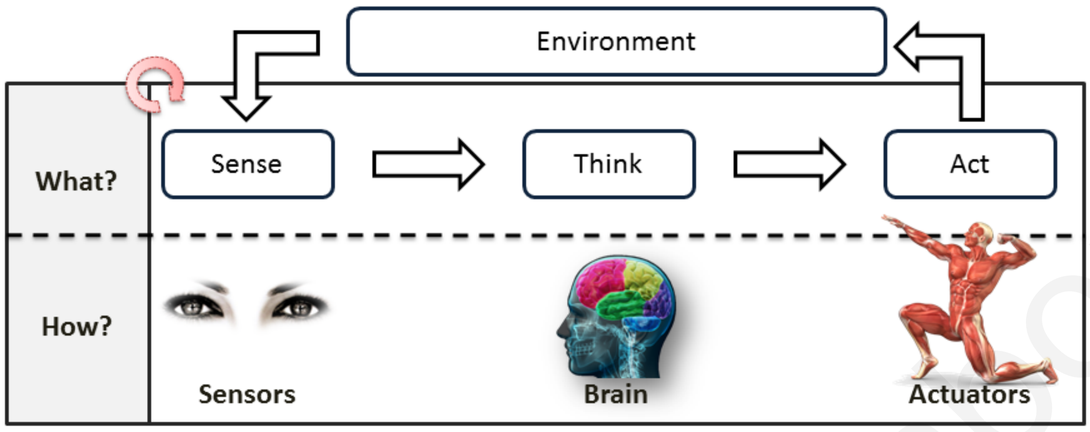


Figure 3.3: **Typical Agents** Agents sense the environment they “live” in through sensors, such as their eyes or ears (when talking about humans). Then they process the sensed data and decide on actions to perform, through their brains and actuators (such as their body) respectively. Their actions affect the environment, which results in a bi-derectional relationship between agents and their environment.

3.1.2 Virtual Humans as Intelligent Agents

The model of the human being (and other animals) that was described in the previous section influenced the design of *Autonomous Intelligent Agents*, which are autonomous entities that live in an environment, retrieve information from sensors (i.e., an agent’s eyes and ears), processes them and performs actions to achieve some goals (Figure 3.3). A lot of the times, these agents can collaborate to achieve some goals, they can have some kind of learning and memory modules to be able to adapt to their environments (similar to real humans but at a much smaller degree). As such, humans can be considered as the most complex agents; the holy grail of artificial intelligent researchers.

The three main components of a virtual human are *sensing*, *thinking* and *acting*:

Sensing In most of the work presented in this thesis, sensors return simplified representations of the local state near the agent. Agents typically store their own internal state and inform a global spatial database with their positions so that agents can retrieve positional information for all the other agents (Figure 3.5). Our main state representation assumes a very simplified vision based approach (the Temporal Perception Pattern – TPP, see Chapter 4). Alternative approaches encode basically the relative positions of neighboring agents to the agent's local coordinate system.

Thinking A typical model for "brains" of virtual humans can be seen in Figure 1.7; a virtual human should be able to make high level decisions, path routes from one goal to another and steer between (sub)-goals in a layered manner. Typically, in agent based simulations, a ruled based system is responsible to make decisions based on knowledge given by the designer. Alternatively, and this is the approach we employed in this thesis; data from real world crowds act as the *experience* gained by a real human being and the decision mechanism for navigation is extracted implicitly based on these data and used throughout the lives of the agents. Goals are set using a higher level user defined system, since it is quite difficult to capture higher level goals of real human beings just from videos.

Acting Agents in the proposed framework are mainly equipped with one type of action; *steering* between higher level goals. This means that virtual humans move from one point to another whilst trying to interact as realistically as possible. These actions are performed by mimicking the steering behaviour of people of of (mainly) real world examples; which in a lot of the cases in not simply avoiding potential collisions with other humans but also higher level interactions such as talking, stopping etc.

Having a model that describes each individual virtual human as an agent, a simple simulator can be easily defined.

3.1.3 Basic Simulator

Each agent in the environment is equipped with a set of characteristics (such as size, age, genre etc) and a set of goals (typically a priority queue), which are decomposed into simpler tasks (such as move from point A to point B).

On the most basic level, a simple crowd simulator (Algorithm 3.1) loops over all agents and lets each agent query the environment to get its current state, select appropriate actions and perform them. Querying the environment is usually performed through some spatial data structures such as grids or kd-trees that hold the positions of other agents and static objects. Action selection can be performed using either ruled based, data-driven (i.e., rules are implicitly defined) or hybrid systems (i.e., a combination of the two). Typical actions in a crowd simulation system include velocity changes, forces or trajectories to follow. Performing an action involves moving the agent to its new position, and in the case of a 3D environment, the proper animations are played. Most of the times, performing these actions influences both the agent and the environment and therefore the previously mentioned spatial data structures are informed for changes (or they are rebuild). This procedure typically continues until all agents accomplish their goals; proceeding to new goals typically means decomposing it into simpler goals that are easily manageable.

Updating each individual at every simulation frame is usually not recommended since (a) typically human beings take actions that last more than a percentage of a second and (b) it's computational overkill Typically agents are split into N_b buckets and agents in each individual bucket are updated either sequentially every N_b frames with longer lasting actions or concurrently (in a multi-threaded environment). Most of the times, and this is similar to the approach we follow on the Perception-Action Graph (Chapter 4), agents should not perform actions immediately, rather

Algorithm 3.1 Basic Crowd Simulation Algorithm

```
**INPUT**
**A**: Set of agents
**E**: The Environment

{Main Loop}
**for all** $A_i \in \mathbf{A}$ **do**
  **if** $A_i.hasGoals()$ **then**
    $state \leftarrow A_i.queryEnvironment(\mathbf{E})$
    $action \leftarrow A_i.brain.calculateAction(state)$
    $A_i.performAction(action, \mathbf{E})$
    **if** $A_i.currentGoal().achieved() == True$ **then**
      $A_i.proceedToNextGoal()$
    **end if**
  **end if**
**end for**
```

they select possible actions and at the end of the simulation step they collaborate on selecting best actions (Algorithm 3.2).

Frameworks by Shao et al. [89] (Figure 2.2) and Pelechano et al. [73] are similar to the one presented here; state of the pedestrians is composed of external state from perception of the environment and internal state that characterizes individuals. This information is used by the higher level layers (which are rule based) to decide on actions the agent should take.

Data-Driven Simulation In a data-driven simulator, the action selection mechanism employs data of real world crowds to decide on actions (Figure 3.4). This typically means, that whenever an agent has to decide on an action, a database of state–action pairs is queried to select an action (or a set of actions) to be applied. These techniques are slow since typically state representations are high dimensional and searches are continuous, but have the added benefit of being more realistic to a typical rule based system. Additionally, actions are in most cases dependant of the currently observed state only and do not take into account previously observed states and actions.

Algorithm 3.2 Crowd Simulation with agent collaboration

```
INPUT
$\mathbf{A}$: Set of agents
$\mathbf{E}$: The Environment

{Main Loop}
$possibleActions \leftarrow \{\}$
for all $A_i \in \mathbf{A}$ do
  if $A_i.hasGoals()$ then
    $state \leftarrow A_i.queryEnvironment(\mathbf{E})$
    $actions \leftarrow A_i.brain.calculatePossibleActions(state)$
    $possibleActions.add(\{A_i, actions\})$
  end if
end for
{Find the best actions to be performed for all agents}
$[\hat{\mathbf{A}}, \mathbf{actions}] \leftarrow selectBestActions(possibleActions)$
{Perform selected actions}
for all $A_i, action_i \in [\hat{\mathbf{A}}, \mathbf{actions}]$ do
  $A_i.performAction(action_i, \mathbf{E})$
  if $A_i.currentGoal().achieved() == True$ then
    $A_i.proceedToNextGoal()$
  end if
end for
```

3.1.4 State Representation

State representation is important, since it should be able to represent as compactly as possible the most important stimuli that affect the behaviour of agents. Since in typical crowd simulation systems we are mostly considered for steering; information such as positional information of agents and statistical information are the typical approaches employed in the literature. State should be able to represent not only the current instantaneous state of an agent and its surroundings, but also encode temporal information; that is information over time. Typically, **temporal** information is either encoded with temporal parameters such as velocity or a number of instantaneous states; i.e., a number of states over a timewindow. Additionally, state should be encoded using a constant number of parameters and each parameter should have the same semantics over time; e.g. the first parameter should always be the distance to the nearest neighbourr. This is

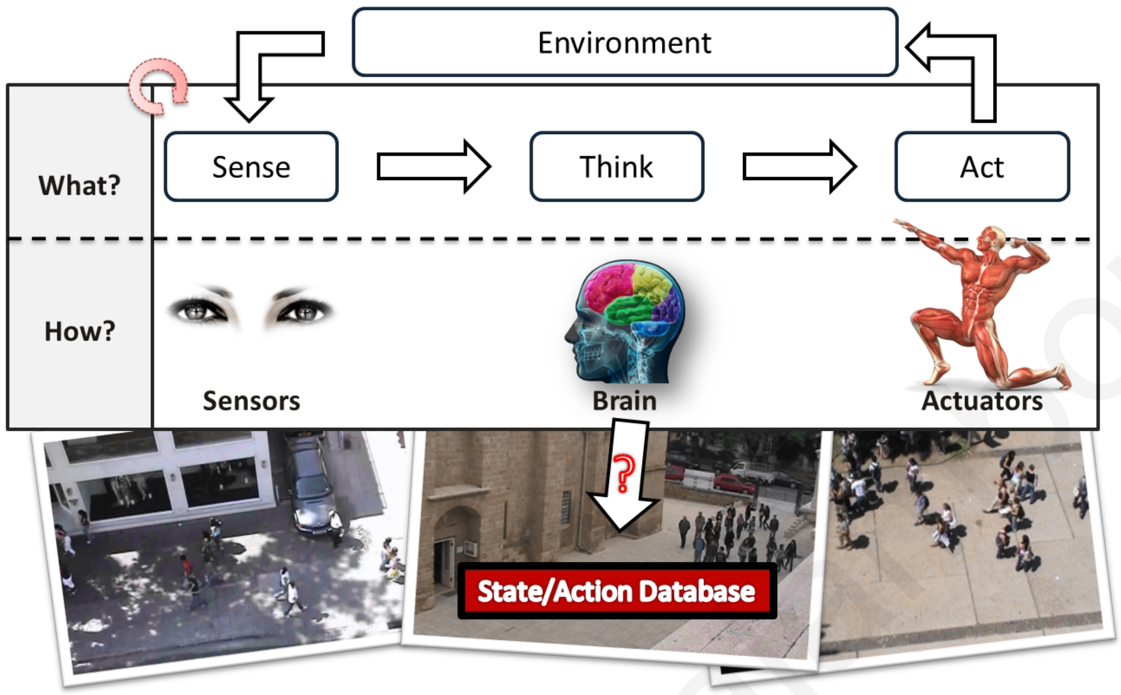
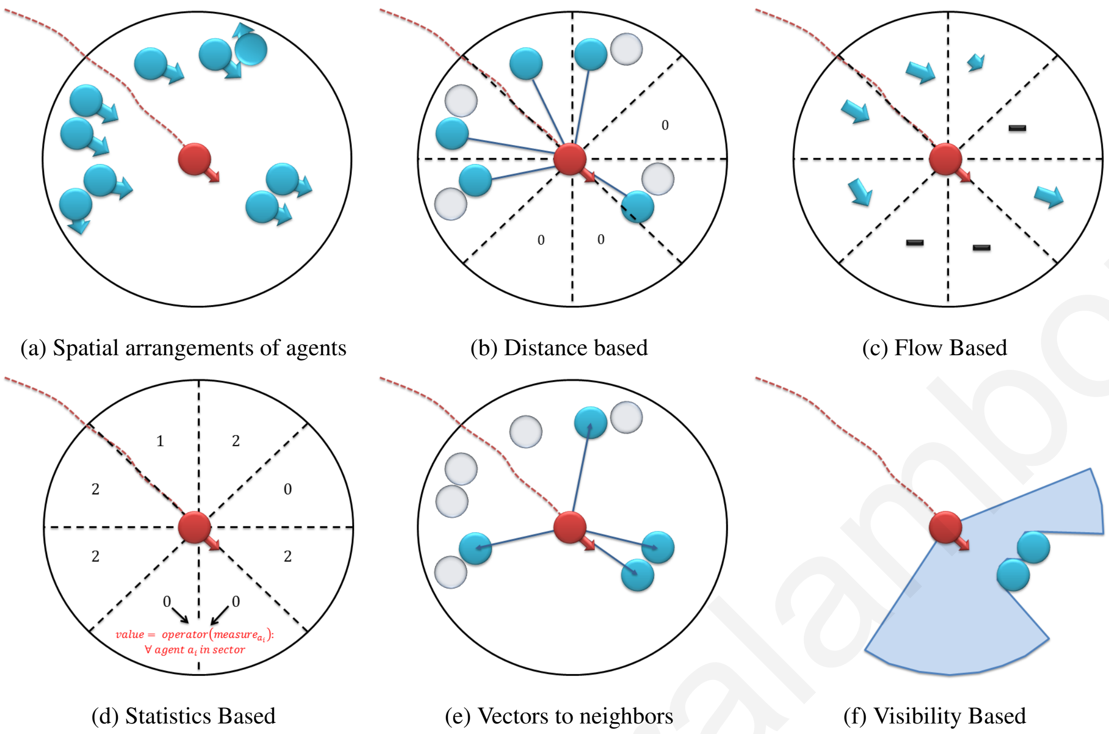


Figure 3.4: **Data-Driven Agents** A data-driven agent typically employs examples from real world crowds to decide on actions.

especially useful when we are considering data-driven approaches where databases of example situations are typically used and therefore searching should be as simple as possible. Lerner et al. [51] for example employed variable sized state representations, which proved quite difficult to manage.

In Figure 3.5, we can see some possible instantaneous state representations; that is state of the currently observed moment only. To obtain these representations the nearest neighbours of a subject agent are typically retrieved. The arrows indicate the current moving direction of agents and each state is *relative* to the moving direction of the subject agent. To have a constant number of parameters, some of the the approaches in Figure 3.5 divide the area around the agent into 8 regions and for each either the distance to the nearest agent (Figure 3.5b – similar to the work by [48]), the average flow (Figure 3.5c) or some statistics such as average velocity or density (Figure 3.5d) are found. The approach in Figure 3.5e on the other hand represents the state using relative positions



(a) Spatial arrangements of agents (b) Distance based (c) Flow Based

(d) Statistics Based (e) Vectors to neighbors (f) Visibility Based

Figure 3.5: **Some state representations** Given as input neighbouring agents (a) an agent's instantaneous state *relative to its moving direction* can be encoded using a constant number of parameters. Typical examples include sector based approaches where distances, (b) flow (c) and statistics (d) are recorded for each sector. Other approaches include relative positions of a constant number of agents (e) and visibility based ones (f).

(in polar coordinates – θ and r) to the 4 nearest neighbors, whereas the approach in Figure 3.5f represents state using a denser visibility approach where all agents in the field of view of the agent are considered (as cylindrical objects) – this is the approach we selected for our implementation in (Chapter 4). State in this thesis is based on visibility; based on the observation that most of the data observed by a human at any given time come from visual stimuli (see Section 3.1.1 and for more details Nørretranders [65]).

Additionally to the relative placement of nearby agents and obstacles, state could also hold other parameters such as the desired speed of agents, their goals, their stamina, etc – most of these data are difficult to observe in real world data for data-driven simulators and are therefore not considered in this thesis.

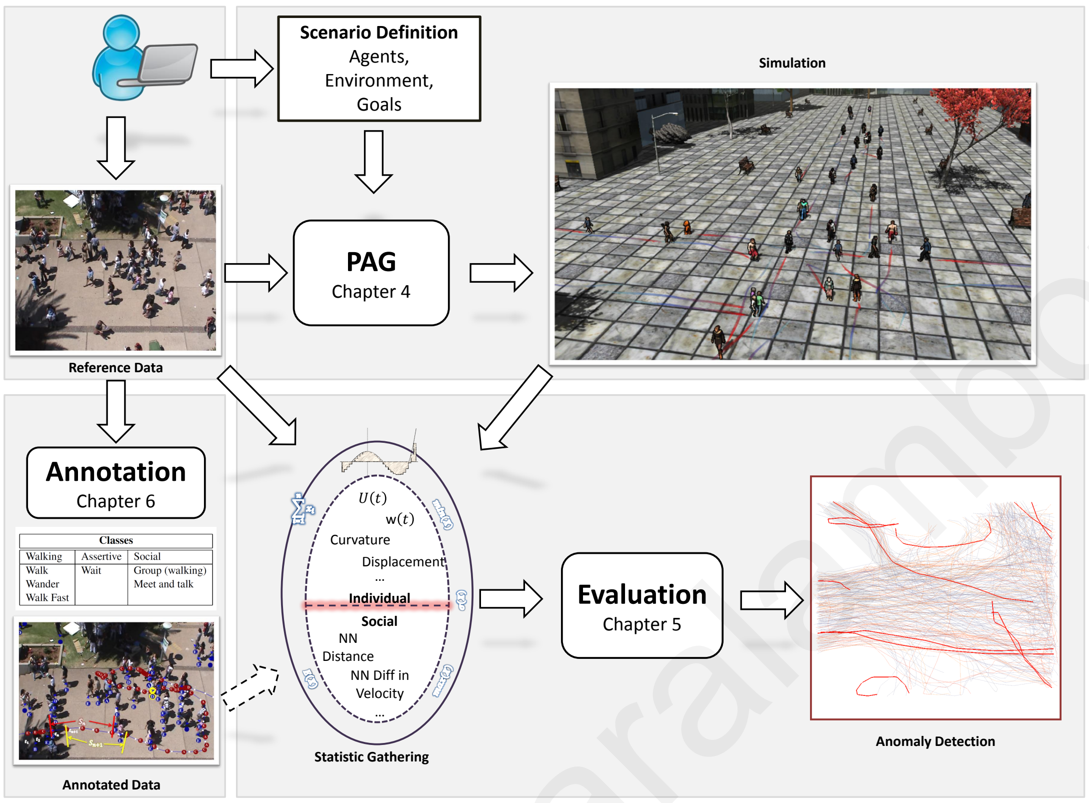


Figure 3.6: **The Overall Data-Driven Framework** An overall picture of the systems developed this system. These include a crowd simulator, a system to evaluate crowds based on data and a high level behaviour annotation system. The evaluation component of the framework can potentially use data from both the simulation and the annotation components; even though for the latest it is not required. On top of all these components is a user/designer that defines the desired behaviour through some data and the required scenario and evaluation metrics.

3.2 Discussion

In this chapter we presented a model for representing virtual humans as agents, possible state representations and a framework for simulating agent based crowds; both data-driven and rule based. The complete data-driven framework as it was developed in this thesis is presented in Figure 3.6. The developed components include an efficient data-driven crowd simulation framework based on graphs Chapter 4, a data-driven evaluation system Chapter 5 and a higher level behaviour annotation system based on random forests Chapter 6). An overall picture on how simulation is performed in the proposed simulation framework can be seen in Figure 3.7.

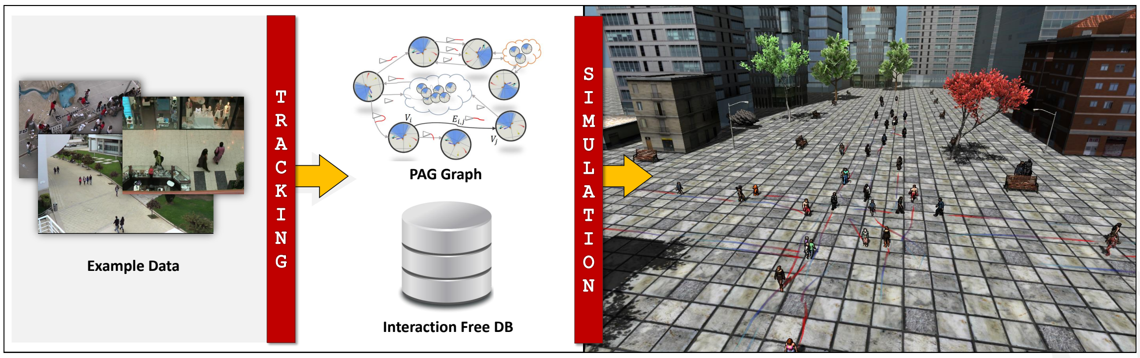


Figure 3.7: **The PAG Simulation Framework** Two auxiliary data structures; the interaction free database and the PAG graph are created out of data of real world or simulated crowds and then used to efficiently simulate realistic crowds in new environments.

Chapter 4

The PAG Graph

Efficient Data Driven Crowd Simulation

4.1 Introduction

Data driven approaches to crowd simulation such as the ones proposed by Lerner et al. [51] and Lee et al. [48] are typically slow, since agents in the simulated environment query potentially high dimensional databases of (state, action) examples at regular intervals, usually multiple times per second (Algorithms 3.1 and 4.1) to retrieve a number of actions that could be performed by the agents. Usually these queries are executed using information about the current state of the agents and not any prior information such as previous states and actions. By using this prior information, the computation time could potentially be reduced with an added increase in simulation quality; actions of real life people depend on previous actions they took and do not change rapidly.

Based on this observation, a data driven method for real-time synthesis of believable steering behaviours for virtual crowds is proposed in this thesis to resolve these issues. The proposed method interlinks the input examples into a structure called the Perception–Action Graph (PAG)

Algorithm 4.1 k-Nearest Based Action Selection for data based crowd simulations

INPUT

A_i: An agent that can be queried for current state.

k: the number of nearest neighbors to search for in the example database.

OUTPUT

$best\_action$: The best action to be applied on the agent. The algorithm selects the k nearest neighbors and combines them into a single action that will steer the agent away from the potential collision.

```
$state \leftarrow A_i.query\_environment()$
$actions \leftarrow find\_closest\_matches(state, k)$
$best\_action \leftarrow combine\_actions(actions)$

return $best\_action$
```

which can be used at run-time to synthesize believable virtual crowds efficiently. The input examples are encoded using Temporal Perception Patterns (TPP) which in essence encapsulate the temporal state of people in tracked data at any given moment (Figure 4.6). TPPs are basically a set of temporally consecutive instantaneous agent centric states (Section 3.1.4) such as visibility, distances to nearest neighbors, statistics, etc. Nodes of the proposed graph structure represent groups of similar TPPs (which define the *perception*) whereas the graph's edges indicate transformations that were observed between each other. The edges store trajectories that are to be followed by the agents (the *actions*). The proposed method has been tested in various scenarios using a number of different input data and was compared against the nearest neighbors alternative approach which is used in most crowd data driven approaches.

We demonstrate results with up to an order of magnitude speed-up with similar or better simulation quality.

4.2 Overview

In typical data-driven crowd steering techniques such as the ones proposed by Lerner et al. [51] and Lee et al. [48], the (state, action) database is queried multiple times per simulation step, potentially once for each agent, in order to retrieve the best matching examples and apply the appropriate action . These queries are usually the bottleneck due to both the high dimensionality of the state representation and the time to select the appropriate action from the returned ones. Dimensionality is a problem, unless approximate approaches, such as Approximate Nearest Neighbors (ANN) search or Principal Component Analysis (PCA) for dimensionality reduction are employed. Even more so, these queries are executed at each simulation step since these approaches assume that crowd steering is more or less a Markov process; agent decisions are based on their current state or at most their previous step's decision.

In the preprocessing phase of the proposed methodology, the input data are arranged into a data structure called the *Perception-Action Graph (PAG)* (Figure 4.1), in order to improve both run-time speed and quality for data-driven crowd simulation. Run-time speed is improved due to the dramatic reduction in nearest neighbor searches whereas the emergent behaviour patterns are of high quality due to the selection of actions that are dependent on previously selected ones and are therefore more consistent. The PAG framework can be used to control the low level steering behaviour of a large number of virtual characters without taking full control over an independent higher level controller/path planner that assigns, modifies or cancels higher level tasks and targets for agents at simulation time.

It should be noted here that, as in previous data-driven methods, agent behaviour is assumed to be affected mainly by external and not internal stimuli such as other agents and nearby geometry. It is acknowledged that internal stimuli emanating from the person's state of mind, mood, beliefs,

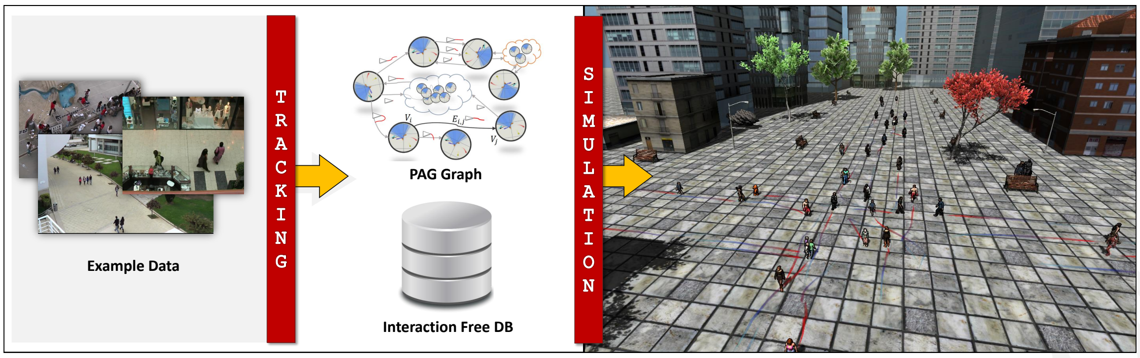


Figure 4.1: **The PAG Simulation Framework** The proposed data-driven simulation framework is divided into two parts; data preprocessing and run-time simulation. During preprocessing, two auxiliary data structures; the interaction free database and the PAG graph are created out of data of real world or simulated crowds and then used to efficiently simulate realistic crowds in new environments.

higher level goals and so on, are also important but these are hard and most of the times impossible to observe in the input video frames and are therefore neglected. In this work external stimuli is represented to account for the flow of perceptual information as this is observed by the pedestrians. Therefore, the steering behaviours of pedestrians, such as turning, accelerating or stopping is assumed to be correlated to this representation and in addition to their higher level goals. This means in practice that the agents move towards their higher level defined goals while at the same time reacting to changes in their visual stimuli.

Input data can be obtained either from videos of real life crowds or extracted from other crowd simulations. At a preprocessing stage, similar situations are identified, clustered and interconnected in a graph structure (the PAG) exploiting possible transitions from one perceptual state to another. Each cluster stores also the different actions taken by the pedestrians due to that situation. Additionally a database of actions that were taken in interaction free (IFDB) situations is generated. At run-time, simulated agents employ both the PAG and the IFDB to simulate efficiently behaviour similar to the input data.

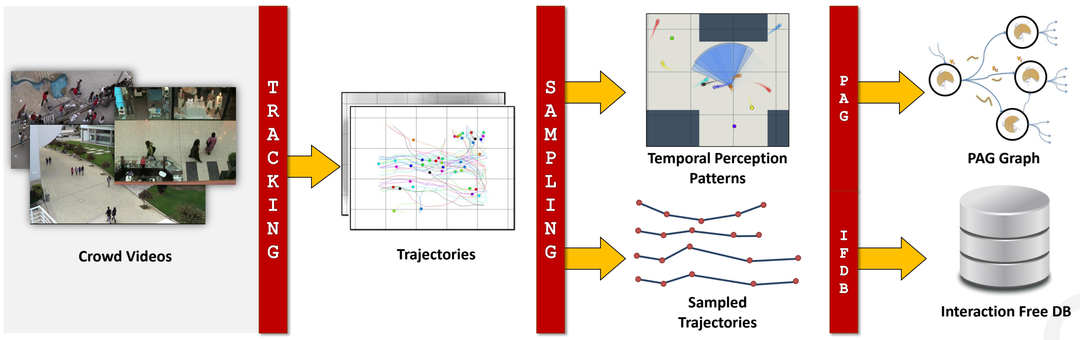


Figure 4.2: **The PAG framework's *preprocessing* pipeline** People from videos of real life crowds or expensive simulations are tracked, their trajectories are extracted and sampled resulting in the PAG graph generation for the samples having interactions alongside a database of interaction free trajectories (IFDB).

4.2.1 Data Preprocessing

Training data is obtained from either videos of crowds or simulations. These data consist of trajectories that were followed by agents over the time they were tracked and descriptions for obstacles such as buildings, benches etc.

These trajectories are then sampled at regular intervals (Figure 4.2). Each sample encodes the state of an agent in the pedestrian's local coordinate system, i.e., with the user at the origin facing along the positive Z-axis. A set of N consecutive samples on the same trajectory (typically 0.5–2 seconds) represent an example situation encountered by the pedestrian (the TPP) which in essence encode the aforementioned flow of information observed by the pedestrian (Figure 4.6).

These examples are then processed and linked together into the PAG data structure depending on their similarity and order of appearance. The PAG is a directed graph $G = (V, E)$ in which a node V_i represents a TPP and an outgoing edge $E_{i,j}$ represents an action $A_{i,j}$ that was *partially* responsible in transforming the TPP stored in node V_i to the one in node V_j (Figure 4.5). An action is defined as a trajectory segment followed by a pedestrian in the input.So, in essence the

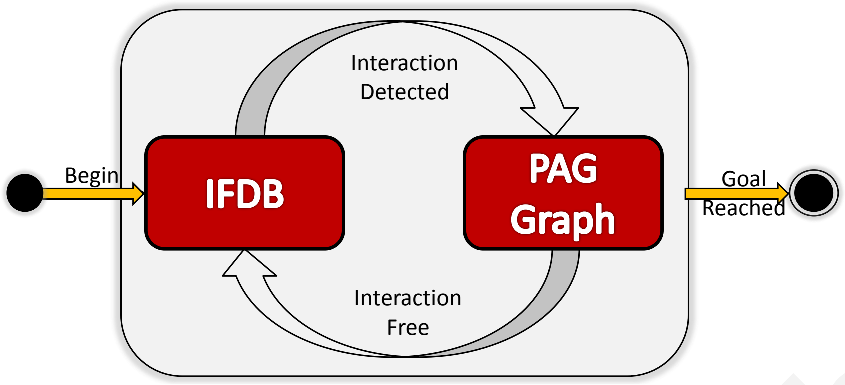


Figure 4.3: **Agents' state diagram** An agent moves towards its goal by using the interaction free trajectory DB. Whenever an interaction is detected; it utilizes the PAG to resolve it.

PAG structure encode states that pedestrians met in the input data interconnected with actions that partly were responsible for the transformation from one state to another.

4.2.2 Simulation

A simulation scenario is composed of the static environment alongside the initial configuration of the agents (positions, velocities, possible desired destinations). As the simulation runs, agents sense at regular intervals the environment to generate TPPs. If the agents do not have any stimuli in their field of view, they simply choose to move towards their goals using trajectories from the *Interaction Free DB* (IFDB), which is a database of all actions that occurred during states that did not contain any interactions. Whenever interactions are sensed (i.e., an external stimuli enters the sensing area of the agent), agents search the PAG to find the node that mathces their state the most as a starting point to handle the interactions (Figure 4.3).

Traversing the graph leads to behaviours such as collision avoidance, agents stopping to talk to each other, leader following etc. Neighboring agents collaborate to select the best actions at each simulation step by performing constrained walks of the PAG. Best actions are defined as the action that minimize the error between actual agent states and those stored on the currently traversed nodes of the PAG (Figure 4.10).

A more detailed description of the preprocessing and simulation phases follows in Sections 4.3 and 4.4 respectively.

4.3 Graph Construction

4.3.1 Data Acquisition

Given a video of crowds, a software to manually track crowds is used to track individual peoples' positions at each frame of the video. A set of trajectories are generated and stored as either a series of points or as a more compact spline representation. These trajectories are then used as the input to the graph construction algorithm. For experimentation reasons, trajectories from other rule based crowd simulation systems such as the ones by [82] were also tracked.

Trajectories More formally, after processing the videos, the input consists of N trajectories $\mathbf{T}_i$, $i \in [1, N]$ and each trajectory $\mathbf{T}_i$, is defined by a series of M_i 3-dimensional points $\mathbf{T}_i = \{\mathbf{p}_k(x_k, y_k, t_k)$: $k \in [1, M_i]\}$, where x_k and y_k represent the position of the tracked person ortho-projected on flat ground in real world coordinates at time t_k (with 0 indicating the beginning of the video) .

Obstacles Additionally, static obstacles are described by basic geometric entities such as circles and rectangles. A circular obstacle with center $\mathbf{C}$ and radius R is defined as $|\mathbf{C} - \mathbf{p}| = R$. A rectangular obstacle is defined by its width w, height h, it's center $\mathbf{C}$ and rotation angle θ around the normal vector of the ground plane.

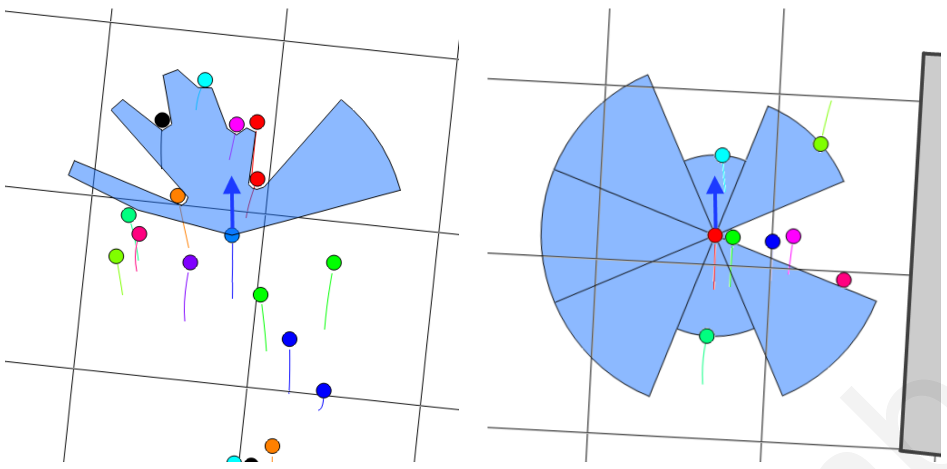

(a) Visibility Based (b) Radial Based

Figure 4.4: **Perception Pattern** In (a) state is represented as the agent's visibility whereas in (b) the agent's surrounding area is divided into radial regions and the distance to the closest neighbor is found for each (similar to [48]). For more state representations, please consult Section 3.1.4.

4.3.2 State Representation

In this work most of the agent states (i.e., the TPPs) are encoded using a visibility based approach, which means that behaviour is directly correlated to what people *observe* in their optical field, similar to the work by Ondřej et al. [66]. In contrast to the work done in [66], we do not use whole images to represent the visibility of agents and each agent is not represented as a cone, rather we use a more simplified version in 2 dimensions, assuming that each person is a circle (which translates into a cylinder in 3D). It is important to emphasize here that the underlying method can be used with other pattern representations also (such as the state representations presented in Section 3.1.4) as long as these state representations are encoded using a constant number of values (Figure 4.4).

Visibility

Visibility encodes the free space in the field of view (FOV) of a pedestrian, i.e., the space that is not occluded by other people or static objects at a given instance. The FOV is defined by it's

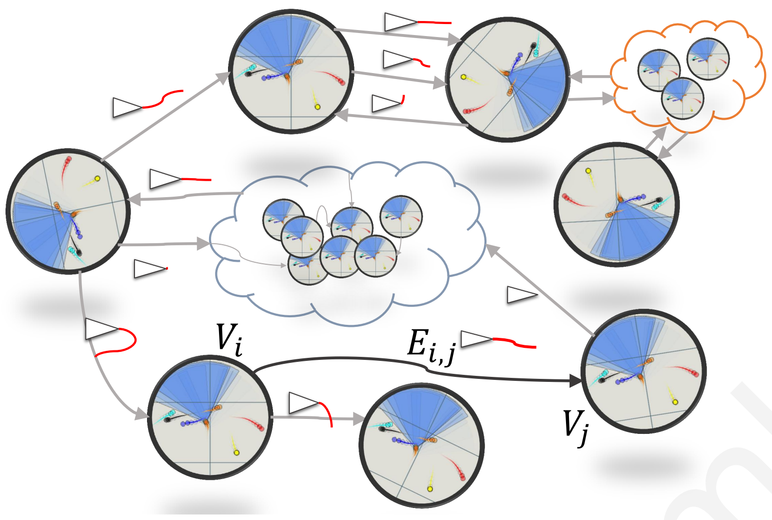


Figure 4.5: **PAG** A PAG represents transitions from one TPP to another. An edge $E_{i,j}$ between two nodes V_i and V_j represents an observed transition between them. This transition was partially the result of a trajectory (red lines) followed by a person with a TPP similar to the one stored in V_i.

spread angle ϕ and the maximum search radius R. It is aligned to the moving direction of the pedestrian (Figure 4.4a) and is sampled at regular angular intervals. For each angular sample the distance to the closest object (static and dynamic) is found. The sampling interval is dependent of the FOV angular width ϕ, the search radius R and the radius of the agents; denser sampling gives better approximation of the visibility at the expense of calculation time and memory storage. For most of the conducted experiments, the FOV width was between $90°$ to $240°$ and was sampled using 20 to 40 rays.We found that good values are field of view of $120°$ sampled with 20 rays.

Temporal Perception Pattern

A predefined number of consecutive trajectory samples, with each sample consisting of a visibility pattern are grouped together and form a *Temporal Perception Pattern* (Figure 4.6). This representation encodes the perception of people over some time period; usually in the range of $0.5 - -2$ seconds. TPPs that have no stimuli in any of their PPs (i.e., they can be considered interaction free) are removed and added in a separate database of trajectories for interaction free cases

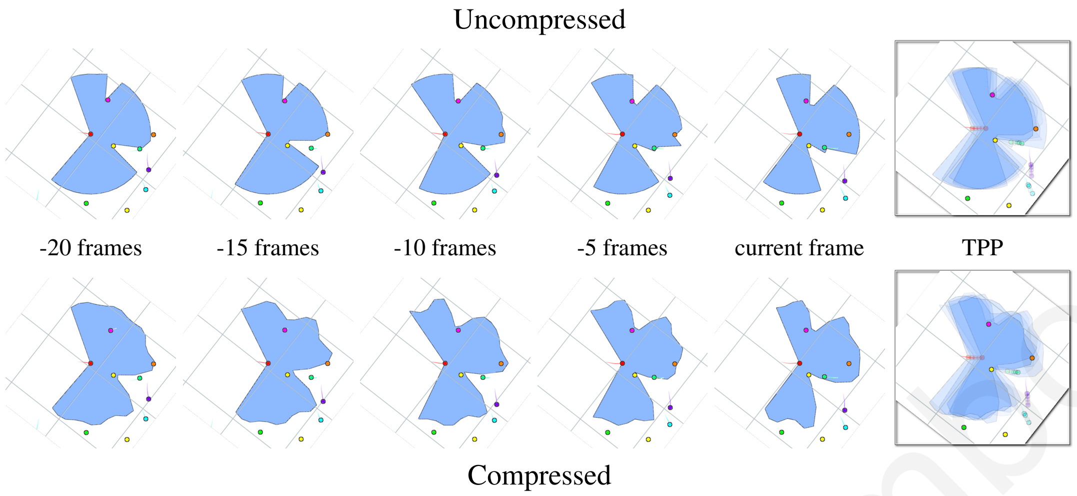


Figure 4.6: **Temporal Perception Pattern** A *Temporal Perception Pattern (TPP)* consists of a set of consecutive in time visibility patterns. (top row) Five equally spaced in time visibility patterns define a TPP. (bottom row) A compressed version of the TPP.

(IFDB) (Figure 4.2). The remaining patterns are inserted into the Perception Action Database (PADB) of $(state, action)$ examples where states are represented by TPPs and actions are trajectories followed by the pedestrians during transitions from one observed TPP to another one. Each of these states has a relatively large dimensionality: if for example a person's trajectory is sampled 5 times per second and for each sample the visibility is represented with 20 values, the state for a TPP of 1s length is encoded with a total of 100 values.

In a traditional data driven crowd simulator, the PADB could be queried by each agent at simulation time to extract the best matching examples to it's current situation using a nearest neighbor algorithm and then apply the recovered actions, either directly or after further processing such as combining the results (as in [48]). This approach is the most commonly used in the data-driven literature for crowds [48, 51, 98]. Instead of following this practice which is expensive, a different approach is employed here: similar TPPs are identified, grouped together and interconnections between them indicating transitions from one to another are found and used to form the PAG, a directed graph which is used to speed up the simulation in a reliable manner.

In Sections 4.3.3 and 4.3.4 a description of the similarity metric and the graph generation algorithms are presented.

4.3.3 TPP Similarity

TPPs are analyzed and grouped together to form the PAG using an approach conceptually similar to the one employed in single character animation by Kovar et al. [40]. To find Motion Graphs, good transition points between poses of bipeds are found using a metric that takes into account joint positions and orientations over a time window. Since a biped is defined by a constant number of parameters and no other characters are taken into account, distance between poses can be found by finding the euclidean distance of joints (assuming the characters have been aligned properly).

TPP Distance The distance metric in this work should be able measure the difference between TPPs rather than poses: for each pair of agents/pedestrians (A, B) and each pair of TPPs $(P_{A,k}, P_{B,m})$ at times k and m respectively their distance should be found using a proper correlation metric and a normalized distance matrix for all frames (with a value of 1 indicating maximum and 0 minimum similarity respectively) should be found. For an input dataset of N pedestrians, $N^2/2$ distance matrices are generated due to symmetric properties of the data. Example distance matrices and a similarity map that interconnects similar TPPs from a real dataset can be see in Figures 4.7 and 4.8 respectively. Dark points on Figure 4.7 indicate low similarity, whereas white indicates high similarity.

Similarity metric Unlike pose comparisons in [40] where the number of joints is constant, the number of stimuli in an agent's TPP might vary from 0 to any number. To measure similarity between TPPs, a metric is required that has a constant number of features and in addition compresses well without being too sensitive to small changes,especially in the distant stimuli. The proposed

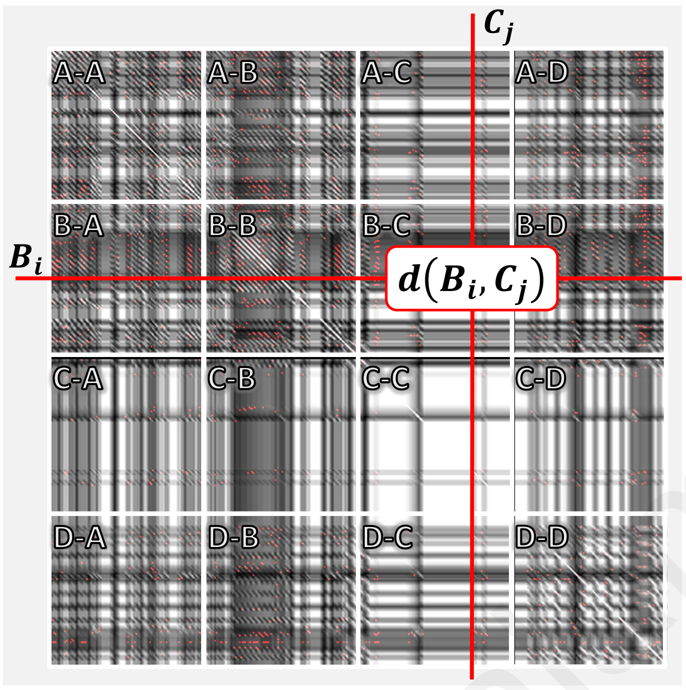


Figure 4.7: **Distance Matrices** Distance matrices between 4 different pedestrians $(A-D)$. The x and y axes represent time and the gray scale values represent similarity values with white indicating the best matches. $d(\overline{B}_i, \overline{C}_j)$ is the distance between the i-th and j-th TPP of pedestrians B and C respectively. The colored dots represent the local maxima that are grouped together to form the PAG graph. Pedestrian C had a lot of interaction free patterns and therefore it was mostly ignored.

metric is based on smoothed out versions of the TPPs using the lower frequency coefficients of its type-II DCT (Discrete Cosine Transform) [1]: each visibility pattern of the TPP (Figure 4.6) is considered as a signal $\mathbf{x}_j(\theta)$, where $\mathbf{x}_j : [-\theta/2, \theta/2] \longmapsto [0, r]$, θ is the field of view and r is the maximum search radius of the agent.

By concatenating all m visibility signals $\mathbf{x}_j$ of the TPP, we get a signal representation of the whole TPP: $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_m)$. This signal is sampled at regular intervals to generate the sampled signal $\mathbf{x}' = \{\mathbf{x}_i : i \in [0, N-1]\}$. The DCT of this signal is $\mathbf{X} = \{X_k : k \in [0, N-1]\}$ where:

$$X_k = \sum_{i=0}^{N-1} x_i cos\left[\frac{\pi}{N}(i + \frac{1}{2}k)\right] \quad (4.1)$$

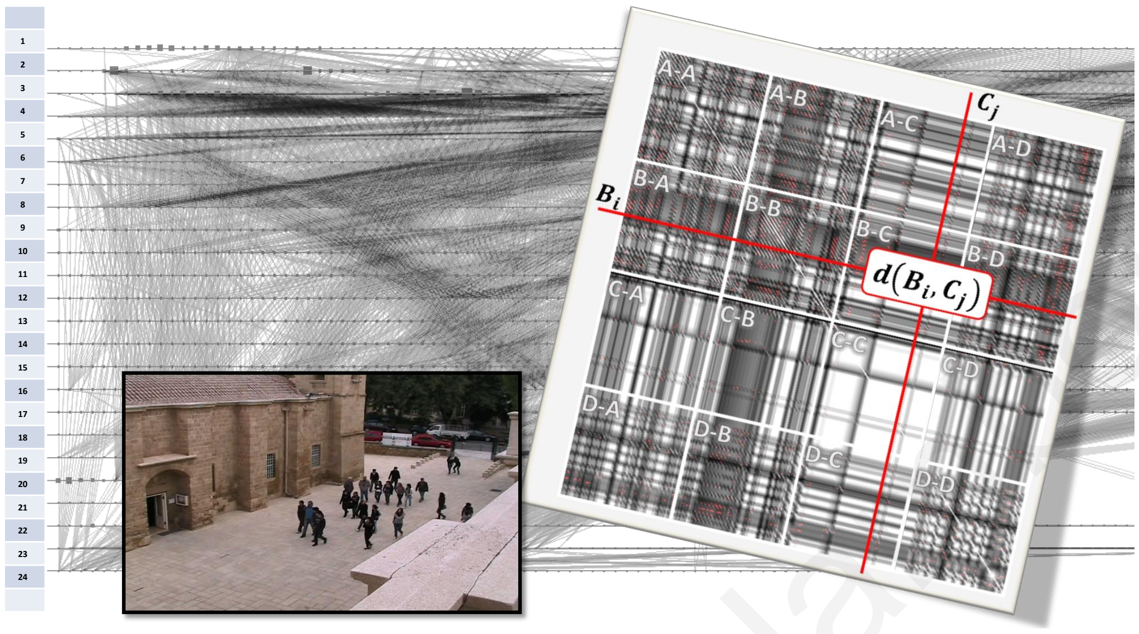


Figure 4.8: **Similarity map** for a video of a "flock" of 24 people in a tourist site. Horizontal lines indicate the trajectory of a tracked pedestrian over time and the dots points where TPPs were calculated. Lines interconnect similar TPPs found by distance matrices (top-right corner). Notice the high similarity between the people due to their flocking behaviour.

Out of these N coefficients (X_0 to X_{N-1}), a small number $M(M < N)$ of the lower frequencies are enough to approximate the original TPP quite well (Figure 4.9). Additionally, the TPP comparison is more stable since data is de-noised at the boundaries of the visibility. In effect, this approach smooths out the distance matrices. To compare two signals $\mathbf{x}$ and $\mathbf{y}$, the Root Mean Squared Error (RMSE) between their lowest $M \in [8, 12]$ DCT frequency coefficients is found and normalized to $[0, 1]$. The RMSE is calculated at the frequency domain to reduce calculation cost of calculating the IDCT (Inverse DCT) of the signals, since the energy of a signal over it's entire domain is equal to the energy of its Fourier based transform over all the frequencies [69]. The correlation/similarity of the two signals is calculated using Equation 4.2:

$$CORR(x, y) = 1 - \sqrt{\frac{1}{N}\sum_{i=0}^{M-1}(X_i - Y_i)^2} \tag{4.2}$$

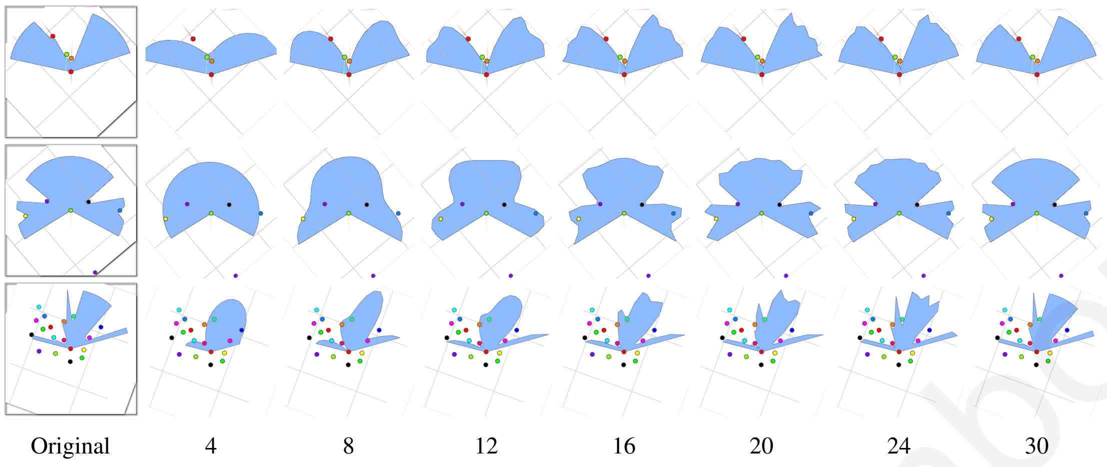


Figure 4.9: The effect of DCT coefficients (bottom row) on visibility based TPP compression for various cases. Here only one of the visibility patterns of the TPP is shown. By using 8-12 coefficients, the original pattern is estimated without much loss in accuracy.

$\mathbf{X}$ and $\mathbf{Y}$ are the DCT transformed signals $\mathbf{x}$ and $\mathbf{y}$ respectively. Notice that the sum is up to frequency M, since frequencies over that are 0.

Best Similarity One scan line of the distance matrix indicates the similarity of a specific TPP to all other TPPs in the input data. Therefore, the local maxima for each scanline indicate the most similar ones. This approach of finding the local maxima at *every* scan line differs slightly from the approach in [40], where the best transition points are found in a local neighborhood. We have found that our approach allows for greater flexibility and increased simulation quality due to increased connectivity of the generated PAG graph with a small performance penalty. Out of all the scanline maxima, the best over a threshold are selected; increasing this threshold value results in increased quality but with fewer interconnections and therefore less choices during simulation (Section 4.5). Usually this threshold is selected to be over 90% similarity (typically even $> 95\%$).

4.3.4 TPP Interconnection

Once the best matches are found, a directed graph (the PAG) connecting transitions between TPPs is generated (Figure 4.5). In Figure 4.8, a semantically similar graph to the PAG displaying the connections between similar TPPs can be seen. Each node V_i of the graph groups together similar TPPs and an edge $E_{i,j}$ between nodes V_i and V_j is created when a TPP in node V_j was a followup of a TPP belonging in V_i. We note a important feature of the graph here: TPPs in nodes are *temporally overlapping*; i.e., the starting section of the TPP in V_j is similar to the ending section of V_i . The edges of the PAG store trajectories that the pedestrian in the source data followed during a transition between TPPs. A node has as many outgoing edges as the number of TPPs that are clustered there; even though the trajectories in the edges might me similar. Potentially, a node V_i can have multiple edges pointing to node V_j but each edge can store different actions (trajectories) that led to that state. After the nodes are interconnected, the PAG is post-processed to remove any dead-ends so that the graph can be used at simulation time without any special handling.

4.4 Simulation

The PAG graph can be viewed as a temporal state-action database; where state is temporal in the sense that is encodes state over some time period and actions are short trajectories. Obviously, if agents follow blindly random edges of the graph, trajectories will be statistically similar (e.g. average speed or acceleration) to the input, but the emerged steering behaviour will not be realistic since agents will not be taking into account other agents and their actions. Additionally, a pedestrians' perception (i.e., the state) changes over time not only because of his/her own actions but also due to other peoples' decisions; a fact that should be taken into account during action selection in the simulation. Typically, TPPs observed during simulation will differ from the patterns the agents observe whilst traversing the graph and therefore agents should collaborate so that dynamically generated TPPs match as much as possible the ones stored on the graph nodes.

More details on the aforementioned issues are to be discussed in the remainder of this section.

Algorithm 4.2 Crowd Simulation - Update Stage.

```
Input:
$A$: Set of Agents $A = \{A_i : i \in [1, N]\}$
Output
$A$: Set of Agents $A = \{A_i : i \in [1, N]\}$ with updated positions
Update($A$):
  for all Agents $A_i \in A$ do
    if $A_i.timeout()$ is True then
      $A_i.action \leftarrow select\_next\_action(A_i$, k)
    end if
    $A_i$.update_pos()
  end for
```

4.4.1 Basic Idea

A constrained walk on the PAG graph is proposed to simulate realistic virtual crowds: whenever any agent in the simulation senses a possible interaction with other agents or obstacles (i.e., something enters its FOV in the case of visibility based TPPs) it selects the node on the graph that mostly resembles its current state as a starting position for interaction handling (Figure 4.3). For as long as there are interactions in the agents sensing area, the agent traverses the graph by collaborating with neighboring agents and applying at the same time the actions stored on the traversed edges. Behaviour such as speeding up, turning, stopping or even stopping so that the agents act as they talk emerges depending on the input data. Identifying possible interactions is dependant of the TPP; in the case of a visibility based representation of the state, an interaction can occur when something (e.g. another agent or building) enters the agents sensing area (Figure 4.4a).

Simulation Example An example of the proposed approach can be seen in Figure 4.10: three agents (A–C) interact in a small simulation environment. **X** represents times at which the agents were employing the PAG, whereas at all other moments they were employing the IFDB. The agents are layered so that agents are on the back when they visited a place before another one. A and B sense each other at timesteps 2 and 3 respectively and begin using the PAG from potentially different starting nodes. B stays on the PAG for only 3 timesteps; i.e., up to the moment it resolves

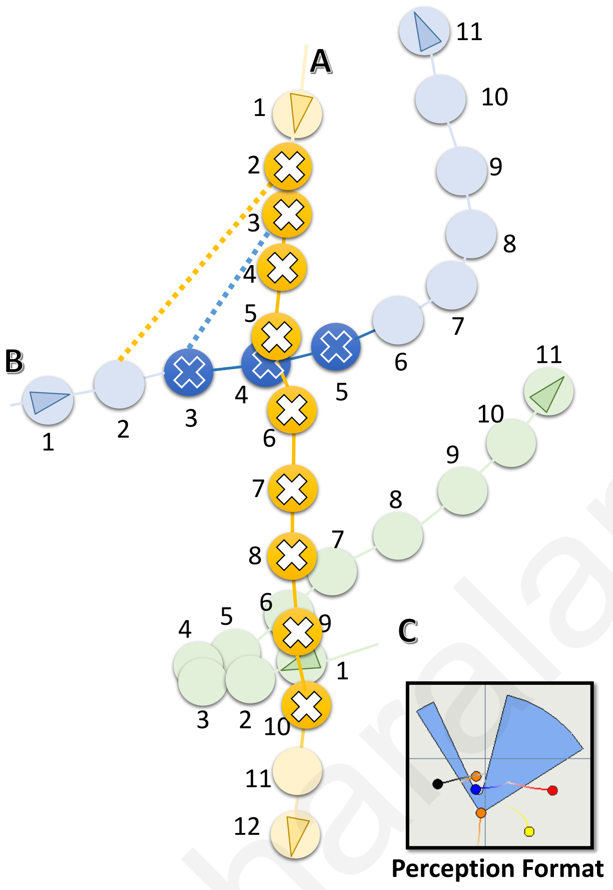


Figure 4.10: State transitions for three agents (A-C). The image in the bottom shows the perception format of the agents. The numbers next to the agents represent the simulation step and the **X** symbol indicates that the agent is currently using the PAG.

the interaction it sensed, whereas A keeps using the PAG to steer from both B and C up to timestep 10. C never had to use the PAG since it never sensed the other two in it's sensing area. The whole time agents A and B use the PAG, they collaborate with all the other agents in their neighborhood (even C which is not using the PAG) to select the best actions (edges).

4.4.2 Initialization

In the proposed implementation, each agent is assigned a path that is either pre-calculated or dynamically generated using traditional path finding algorithms [24, 103]. Although this is not strictly necessary, this shows that agents can be controlled by a higher level controller, something

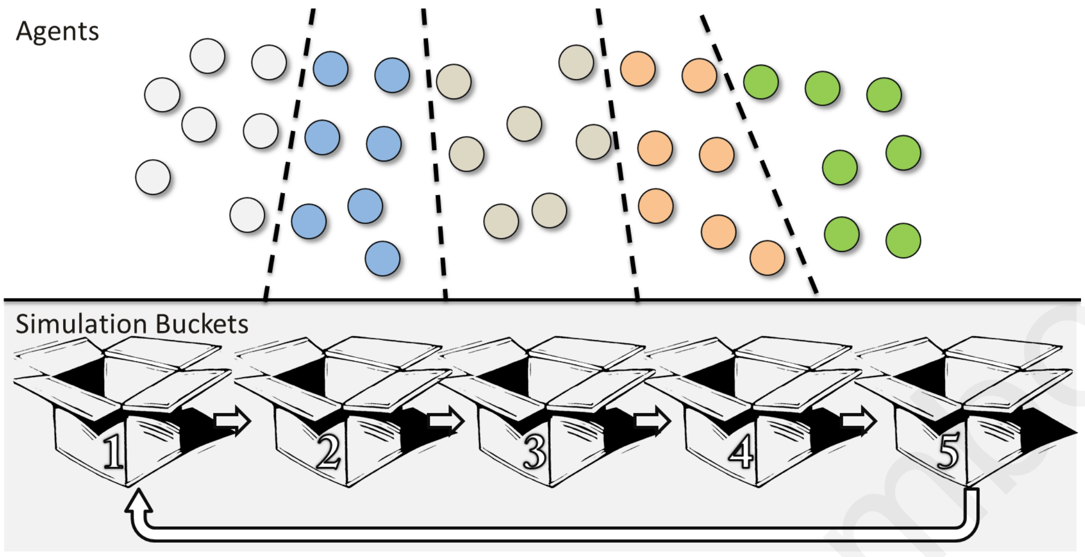


Figure 4.11: **Binning agents for performance** To improve run-time performance, agents are split into bins of equal size, and at every simulation step only one of these bins selects actions; the rest follow previously selected actions. In this example, 35 agents are split into 5 bins of 7 agents each and action selection is performed using a round robin approach.

that is lacking in most data-driven crowd simulation approaches. During the first few steps of the simulation (Algorithm 4.2), agents simply move toward their first target recording at the same time their TPP at regular intervals. This could be thought of as the steering memory of the agents, the memory that holds the state needed for selecting steering behaviours. In the case of visibility based TPPs, parameters of the visibility such as the field of view angle θ, the FOV sampling rate, the maximum search radius r etc. are selected to be the same as those used during the preprocessing of the source data and the construction of the PAG.

Binning for Load Balancing To improve simulation speed and to balance the work more or less evenly between simulation steps, agents are randomly assigned into X bins of equal number of agents and agents in the same bin are updated synchronously (i.e., on the same time step) (Figure 4.11). Agents in the same bin are updated periodically after every X frames of simulation time making sure that in every simulation frame a fraction of the agents are selecting actions to apply (which is the expensive section of the simulation) whereas the remaining ones simply follow previously selected actions. X is selected to be equal to the length in frames of the action stored on the edges of the PAG; e.g. if an action is 5 frames, each agent is updated every 5 frames.

4.4.3 Interaction Free States

As long as the agent has an interaction free TPP, it can just move towards the target by selecting a trajectory from the interaction free trajectory database (Figures 4.12 and 4.13). A simplified version consisting of 4–5 2D vectors of the last 1–2s of the agent's trajectory (e.g. points $\mathbf{p}_{t-4} \rightarrow \mathbf{p}_t$) is used to query the IFDB and retrieve trajectories from the input that had similar history to the agent's. This trajectory is then aligned to the moving direction of the agent and assigned to it.

Possible Deadlocks We note here, that the single best match could be selected as the actions of choice but this could lead to problems such as deadlocks. If for the example the agent was stationary, and the best action was to stay at its position, the agent would stay in it's position indefinitely (or until some other agent enters its FOV and therefore the agent switches its selection mechanism to the PAG). To avoid issues like the one mentioned above, a k actions are retrieved (typically $k \leq 10$) and one that moves the agent towards the next target is selected and applied. A more clever approach here, could be to find in the data average times that pedestrians stayed in a similar state and let the simulated agents follow on their footsteps; i.e., the agents will have a well defined probability based action selection mechanism.

4.4.4 Agent Interaction

As soon as an agent detects a potential interaction (Algorithm 4.3), it uses its current TPP to search for the node with the best matching TPP in the PAG using a traditional k-nearest algorithm. To improve on the speed and quality of the node search in the PAG, we store simplified (low dimensional) versions of the TPPs in a spatial data structure (*kd-tree*). A good strategy is keeping

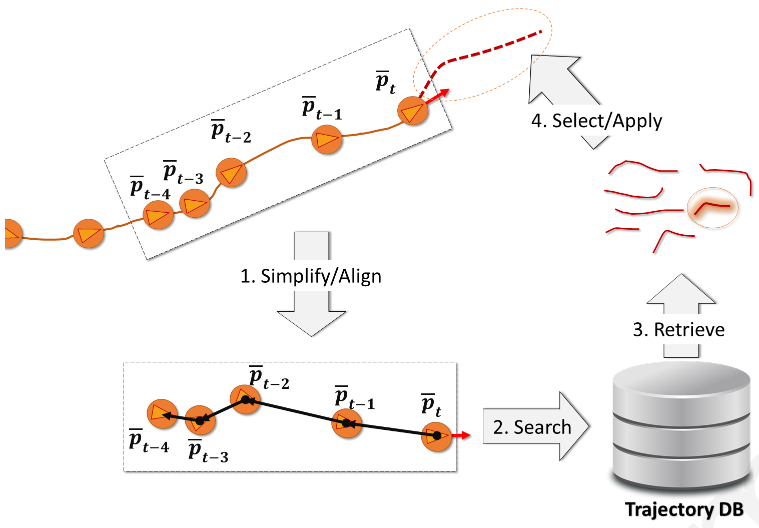


Figure 4.12: Searching for trajectories in interaction free cases is accomplished using a trajectory database (the IAFD). A simplified version of the trajectory is used to index into the database and a trajectory is selected amongst the results.

the lower frequencies of the first and last visibility patterns of the TPP, which together encode the relative movement of an agent against its neighborhood. A total of 8 to 12 values (4 to 6 coefficients per visibility pattern) were found to be enough to compress the TPP. By indexing into the kd-tree using this representation, the k nearest matches are found and the best one is selected by comparing between the uncompressed TPPs. This node indicates the agent's starting node in the PAG to handle any interactions up to the moment the agent is in an interaction free state.

Action Selection

When an agent finishes traversing an edge and arrives at a PAG node V_i, a series of actions are to be considered depending on the outgoing edges $E_{i,j}$ of the node. These edges lead to a set of different nodes V_j that correspond to different possible future states that the agent should have; we note again that these nodes store states found in the input data which correspond to situations found in the input data and could be slighlty different than the observed states in a simulated environment. A fitness function O has been defined (Equation 4.3) that takes into account the neighboring agents predicted future positions and the TPPs stored on nodes V_j. The edge E_k with the minimal fitness value is selected as the action to be applied by the agent.

Agents Update States This action selection is performed using a *greedy approach*. Knowing that the trajectories stored on the edges are of a constant length (time wise), an agent traversing the PAG predicts the future positions of all the other agents in its neighborhood as this is enforced by their state. During simulation, agents can be in one of two states: they are either using the IFDB or they are traversing the PAG (Figure 4.13). Additionally, some of the agents are already assigned a trajectory from a previous simulation step since they were assigned to a different update bin (Section 4.4.1) whereas the agents in the same update bin are not assigned any trajectory yet and are updating the same time as the agent. Some of the agents that have already been assigned a trajectory are traversing an edge of the PAG while the rest are following a trajectory from the IFDB. For agents already assigned a trajectory, their future positions are well defined by this trajectory. For the currently updating agents, i.e., the ones with uncertain possible future positions, their current velocity is used to extrapolate future positions.

Having all the future positions of the agents in a given neighborhood, the visibility patterns for both the current timestep and a future timestep (i.e., the next time the agent should update and select a new action) can be estimated by observing the trajectories on the edges. If the currently visited node V_i stores a TPP_i, which in turn is composed of n visibility patterns $VP_{i,1}, \ldots, VP_{i,n}$, then visibility patterns $VP_{i,2}, \ldots, VP_{i,n}$ alongside the predicted pattern VP_f represented a possible future TPP (TPP_f), the state an agent will *probably* observe after traversing an edge. Having a set of possible future perception patterns, the agent selects the edge $E_{i,j}$ that minimizes the difference between the predicted pattern and the target node's V_j stored one (TPP_j). In essence, during simulation each agent tries to be consistent with the information that is stored on the graph;

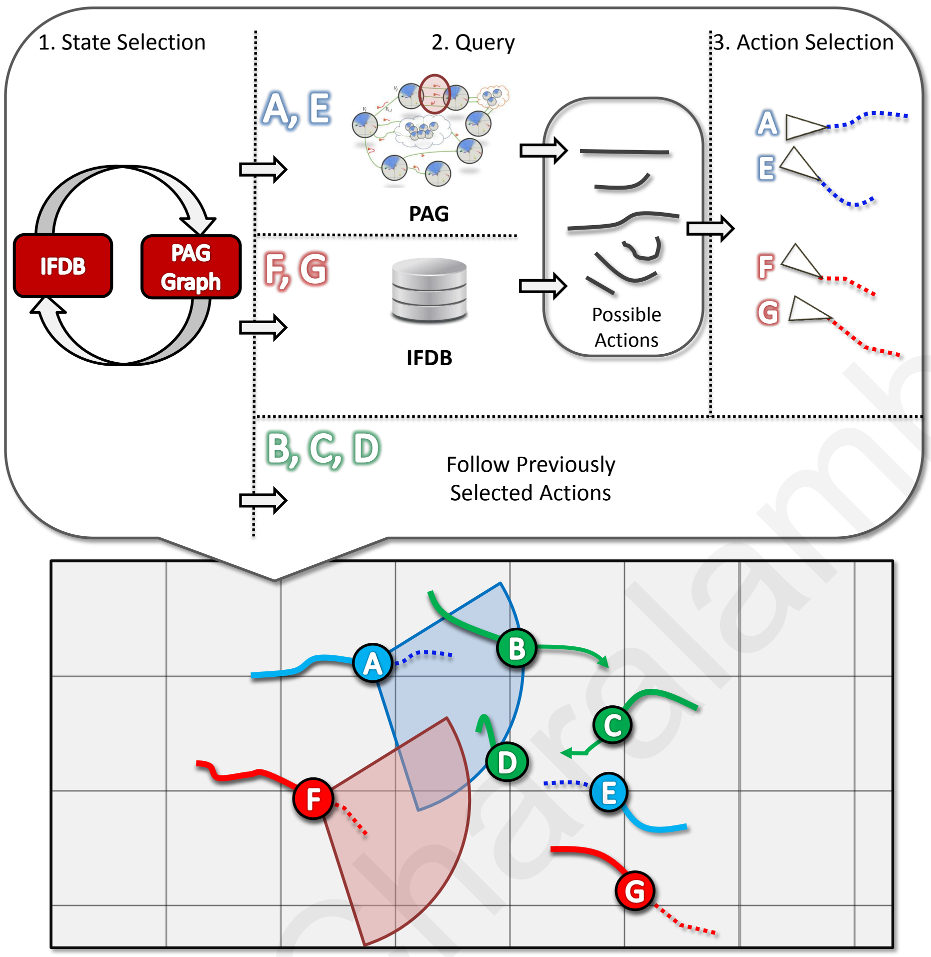


Figure 4.13: **Action Selection Mechanism.** Agents B, C and D selected an action in a previous simulation step, whereas the rest of the agents have to select an action. A and E use the PAG graph for action selection whereas F and G use the interaction free database since they currently do not sense any interactions. A and E collaborate to select the best possible actions using known trajectories of B, C, D, F and G and extrapolated information.

whatever the agent perceives from the environment and represents its state should match what the agent is observing on the currently visiting nodes of the PAG graph.

Instead of only trying to minimize the error between the perception patterns, an objective function that takes into account the target distance is defined:

$$O(TPP_n, TPP_f, T_n, T_f) = (1 - corr(TPP_n, TPP_f))^{\alpha} \left(\frac{T_f}{T_n}\right)^{\beta} \tag{4.3}$$

Algorithm 4.3 PAG: Select Next Action

```
INPUT
$A_i$: An agent that can be queried for current state.
$k$: Number of nearest neighbors
select_next_action($A_i$, k):
    $error \leftarrow 0$
    $action \leftarrow new\_action()$
    $state \leftarrow A_i.query\_environment()$
    if $state.free()$ is True then
        $action.trajectory \leftarrow get\_trajectory\_to\_target(A_i)$
    end if
    if $state.moving\_on\_graph()$ is True then
        $A_i.node \leftarrow A_i.edge.target\_node$
        $error \leftarrow A_i.node.visibility - state.visibility$
    end if
    if error is large or $state.on\_graph()$ is False then
        $A_i.node \leftarrow select\_best\_node(A_i, k)$
    end if
    $A_i.edge \leftarrow select\_best\_edge(A_i.node)$
    $action.traj \leftarrow tranform\_locally(A_i.edge.trajectory)$

    return $action$
```

TPP_f and TPP_n represent the predicted TPP and the TPP stored at the end of the edge respectively, whereas T_f and T_n represent future and current distances from the agent's target (or distance to path). $corr(TPP_f, TPP_n)$ calculates the correlation between the two parameters, with a value of 1 indicating maximum correlation and 0 no correlation whatsoever, as seen in Equation (4.2). Parameters α and β represent weights for the two factors. For most of the presented experiments, weights α and β are assigned to 1; i.e., both are of equal importance.

Jumping nodes on errors After traversing an edge, there is a probability for error between the observed TPP and the one on the current node. This can lead to inconsistencies on the graph walks: an agent might drift and move on nodes that differ significantly from what the agent observes. Two policies were considered here to handle this issue:

- *jump*: when a large error is observed, agents could jump to better matching nodes instead of continuing traversal, employing the same approach as in Section 4.4. This approach introduces penalty in performance at the cost of better quality.
- *no jump*: the agent simply ignores the errors and continues traversing the graph keeping the cost low at the expense of quality

For most of the conducted experiments, assuming that the graph is well connected and the update frequency is high enough (i.e., 5 times per second), the *no jump* strategy works well, since agents in inconsistent nodes try to adapt and move towards more consistent nodes in just a few simulation steps. Problems arise in complex simulation environments that differ a lot from the input data. In these cases, nodes tend to have different states than the states observed by the agents, and therefore the *jump* policy is helpful. The *jump* policy can be though of as providing a weight factor on selecting between two data driven approaches; the simple k-nearest one and the PAG. A threshold value of 0, means that the agent simply jumps to a new node on every simulation step, whereas a threshold value of 1 indicates a case where the agents never jump and follow the edges of the graph. Any inbetween value, balances between the selection of traversing edges or selecting nodes. An *always jump* policy that forces jumps at every simulation step (threshold value is 0) is basically the simple k-nearest neighbors approach; at each time step a search is committed to find the best matching state and is in essense similar to the other proposed data driven methods for crowd simulation.

Controlling agents By inserting into Equation (4.3) the distance to the path, a sense of control is added on the simulated agents; they try to stay near their higher level set path. If more control is required, one could simply add other factors in Equation (4.3) such as for example the distance to nearest neighbors, or setting as target another agents to simulate leader-follower with locomotion

Name	Trajectories	Input Size (frames)	Construction Threshold
lerner-flock	23	766	0.95
lerner-zara	148	9013	0.95
lerner-students	330	4432	0.98
eth-hotel	280	18060	0.98
eth-eth	360	12380	0.98
lee-chat	10	602	0.95
reynolds-pedestrians	50	500	0.99

Table 4.1: Input Datasets used for the experiments in this work.

behaviour similar to the input. This means that control can be achieved by just modifying this equations and no other rules are to be set anywhere else in the simulator.

4.5 Results

To evaluate the proposed simulation framework, a number of tests that use as input both real and synthetic data (Table 4.1) were performed. These data were collected from various sources and consisted at their most primitive form of a set of spatio-temporal positions (x, y, t) for each trajectory. Pedestrians from some of the videos were tracked using our own dedicated semi-automatic tracker whereas some of the others were already tracked.

Implementation The simulator was implemented employing a server-client based approach (Figure 4.14) for two main reasons:

- to allow for wide system integration and
- to decouple the simulation from rendering (assuming a relatively fast and reliable network).

The server was implemented using both the Python and C/C++ programming languages and results were rendered in real time using both an OpenGL and a Unity3D rendering clients, programmed in Python and C# respectively. All simulations were single threaded and run on a single core of an

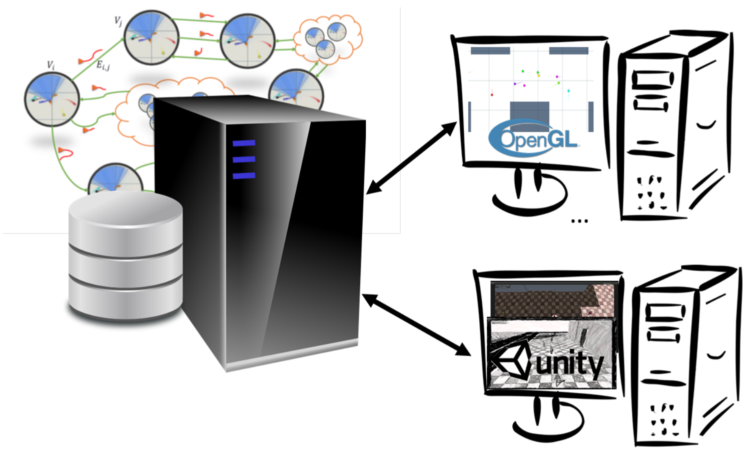


Figure 4.14: **System Implementation** Simulation logic runs in a server that provides agent position and state to the renderers.

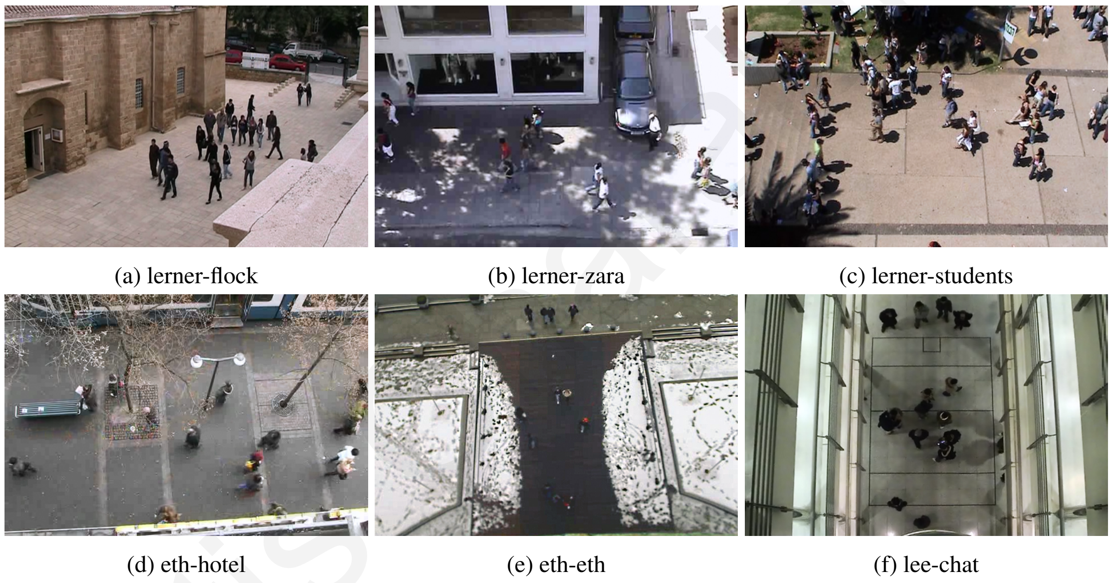


Figure 4.15: Frame grabs from the input datasets of real crowds used in the experiments in this thesis.

Intel Core 2 Quad Core Q8300 clocked at 2.50GHz PC with 4GB of RAM and no GPU acceleration; even though some parts of the algorithm would definetely benefit from a multi-threaded/GPU implementation. Evaluation was performed for both simulation quality and performance.

Quality To evaluate the simulation quality, two approaches were followed. On one hand different scenario cases were run using different input data (such as videos of people walking in a busy street, people chatting, students in university campus etc.) from different video sources and existing rule based crowd simulation systems and the simulated result was visually compared to the input data (Table 4.1). On the other hand, different metrics such as speed, angular change histograms and various other metrics were calculated for both the input data and the simulations and then compared (similar to the work of Singh et al. [94]). The first ”looks good” approach is employed by most of the previous authors on crowd evaluation and can act as a first indication of the quality of the simulations, whereas the second approach tries to correlate the input and output statistics in a quantitative manner. This two-fold strategy was followed since we believe that crowd simulations should be evaluated both quantitatively and qualitatively; people tend to expect more when evaluating virtual people and in a lot of the cases they might overestimate or underestimate the quality of the underlying algorithms, whereas statistics alone might be misleading since simulations with similar statistics might be completely different in their behavioural appearance.

Additionally, a method is proposed that tries to find outlier behaviours using a set of statistics. This method is described in more detail in Chapter 5.

Performance A method similar to the one proposed by Lerner et al. [51] was implemented to study the timing benefit of using the PAG approach instead of just a database of examples. This approach is very similar to the one employed by Lee et al. [48]; the main difference is that in the work by Lee et al. the retrieved examples from the database of example are combined into a single response, whereas in Lerner et al. one of the examples is selected to retrieve the action to follow. This method will be known as $k - NN$ from now on.

In this method, during simulation and at every simulation step, all of the agents query a database of examples using the agent’s current state using a nearest neighbors approach ($k-NN$). Each query returns the *k* nearest (state, action) pairs (Algorithm 4.1) out of which the best example is selected using a more aggressive and accurate comparison method. In order to make the comparison fairer and eliminate any variations that might arise due to different state representations, state in the $k - NN$ approach is represented using TPPs instead of the state representation in [51]. It should be noted here that Lerner's et al. state representation could be used to generate the PAG graph; TPPs can be any time varying state representation of constant dimensionality (Figure 4.4). The same coding optimizations and parameters are used throughout the implementation of both methods, thus in the results that follow we emphasize more the relative speed-up and pay less attention to absolute frame rates - a pure C/C+, more optimized, implementation of the same methods would surely be significantly faster but relative differences are the same.

For all the experiments described in this section, each scenario was run multiple times (10 times, unless mentioned explicitly) and results were averaged out to reduce artifacts due to noise such as OS task scheduling, disk operations, etc. It should be noted here that in most cases each separate simulation is actually well behaved with low variance (i.e., it's of $\Theta(g(n))$ complexity).

4.5.1 Preprocessing

Typical graph construction time ranges from 1 to 60 minutes and is highly dependent on the input data size. This time includes the data preprocessing and analysis time, similarity comparisons and graph generation which is simultaneously stored on disk for later use alongside with all the necessary parameters. A large PAG graph takes up to 30MB when loaded in memory. During our experiments it was found that good values for graph construction are of $95 - 99\%$ similarity which provide good connectivity and also keep the number of edges relatively small. Typically the similarity threshold is selected to have on average ≈ 10 edges per node and small variance. This ensures for a fairly good number of alternate choices during graph walks and relatively constant runtime per simulation step. Also, 8–12 DCT coefficients were used for most of the experiments since they keep the most important features of the TPPs (Figure 4.9).

4.5.2 Scenarios

A series of different simulation scenarios were used as a testbed to demonstrate the adaptability of the algorithm on different data and environments. Simulated trajectories and timing results for agents with simple targets can be seen in the corresponding figures. Please consult the accompanying videos for the full simulated results.

Parameters For most of the scenarios described in the following paragraphs, the same parameters were used: the agents field of view was 90–120 degrees, with a history of 0.6–1 seconds and graph construction quality $90 - 99\%$.

4.5.2.1 Two Opposing agents

In the simplest of scenarios, two agents start moving in opposing directions to switch places (Figure 4.16). This simple scenario demonstrates the adaptability of the algorithm on the data: all the experiments were run using different input data with the exact same parameters (as described in the previous paragraph). Using different datasets as input, has a profound effect on the simulated trajcectories.

lee-chat Using as input data from a group of people standing still and talking (some enter the video walking and then stop to talk), the agents walk towards their targets and stop when they reach at a close distance to the other to talk for a few seconds. After some time, they continue to their destinations. Both agents reach their destinations after 53 seconds (at 25fps).

eth-hotel Using as input pedestrians walking in a busy street, the agents react similarly. Instead of stopping like the *lee-chat* dataset, the agents avoid each other gracefully and reach their destinations much faster (20 seconds). Different source data have a profound effect on the simulations; by using as input a source video with people chatting (*lee-chat*), in both cases the simulated agents stop when they sense each other and start chatting.

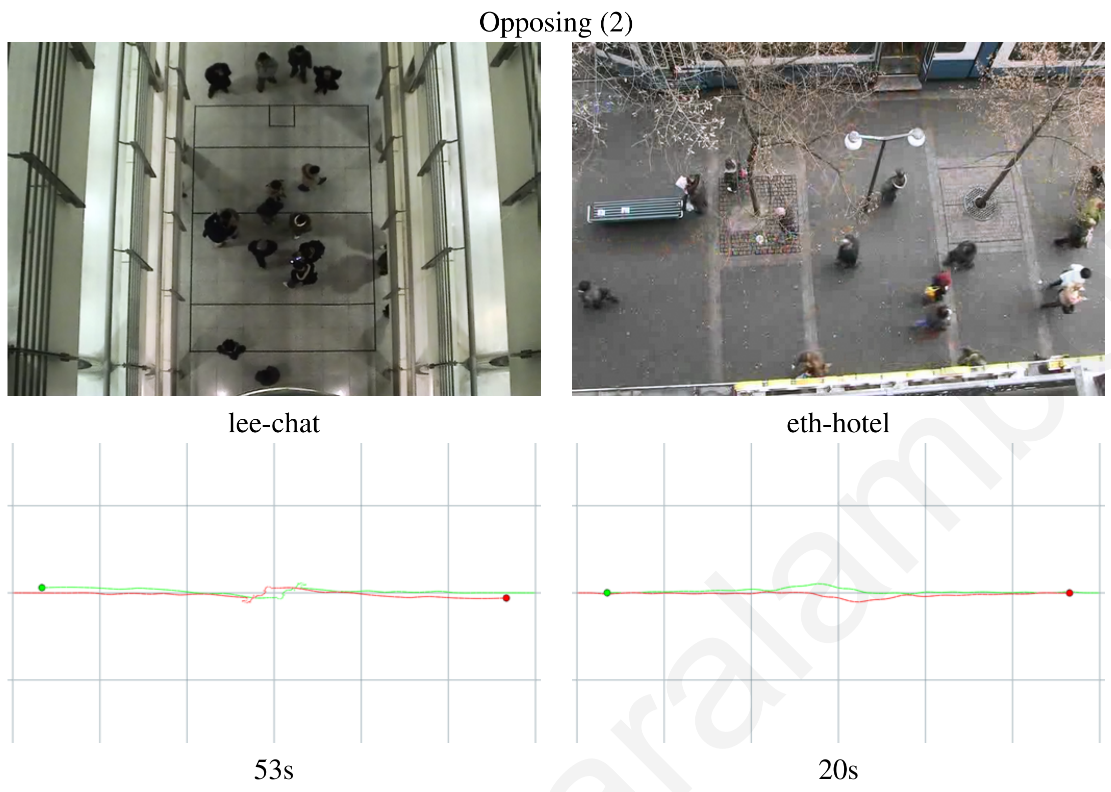


Figure 4.16: Trajectories for two agents moving against each other using as input videos of people chatting and in a street. Please consult the accompanying video for the simulations.

4.5.2.2 Four Opposing Agents

This scenario is slightly more complex than the scenario with the two agents, in the sense that when reaching the center of the simulated area, agents should be able to react to a more complex situation (Figure 4.17). All the parameters are the same as in Section 4.5.2.1:

lee-chat As in the previous experiment, agents start chatting when they reach the center of the simulation environment and then they disengage to move towards their targets which they eventually reach at the 68 second mark. Time is longer here, since there are more interacting agents and whilst one agent leaves earlier, the others stay in a chatting mood and continue until everybody leaves.

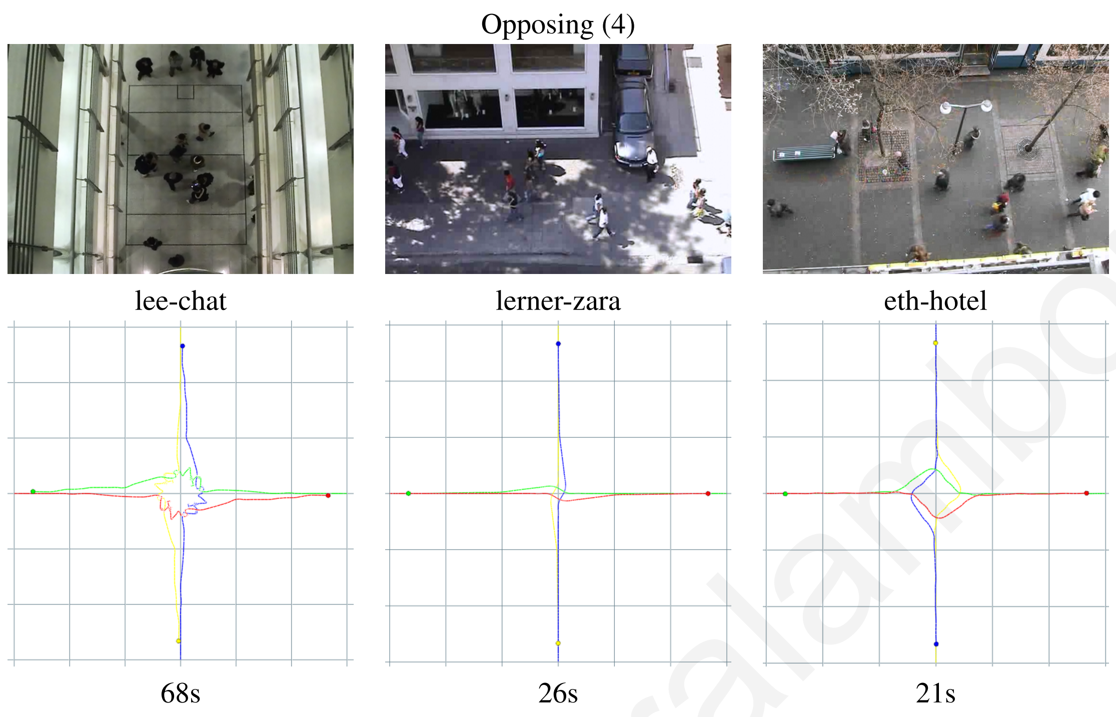


Figure 4.17: Trajectories for four agents moving against each other using as input videos of people chatting and in two different pedestrian areas. Agents in the chat simulation took longer to reach their destinations due to the chatting interaction whereas in the other two simulations agents reach their destinations in shorter time periods adapting to the moving patterns in the input videos.

eth-hotel, lerner-zara These two datasets consist of pedestrians in different streets from different countries/regions. Macroscopically, the behaviour of the people in the video is very similar but people in the *eth-hotel* dataset are somewhat faster (on average) than the people in *lerner-zara*, whereas people in *lerner-zara* have a smaller distance to the nearest neighbors. This means that in essence, the agents using the *eth-hotel* dataset reach their destinations in a shorter time (21 instead of 26 seconds) and with larger interpersonal distances than the *lerner-zara* dataset.

4.5.2.3 Circular Scenario with 14 agents

This scenario shows the potential of the PAG framework to use data from other rule based simulators, such as the one by [82]. A simulation of 50 agents walking in a hexagonal path against

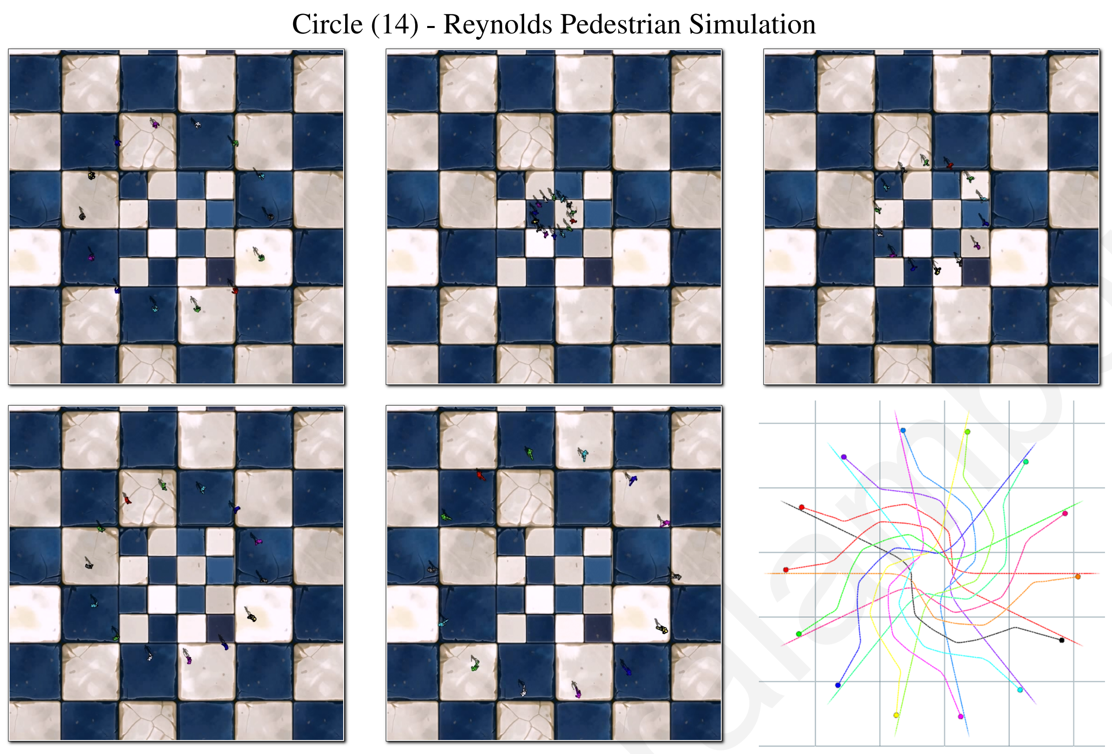


Figure 4.18: 14 agents in a circle switch positions with their opposite agents using as input data from the Opensteer simulator ([82]).

each other was used as the input (*reynolds-pedestrians*). Only 20 seconds of simulation data (500 frames at 25fps simulation rate) were enough to capture the steering behaviours enforced in the Reynolds simulation. 14 agents were placed in a circle with each agent trying to reach the diametrically opposite direction (Figure 4.18). All the agents managed to do so without colliding with each other; notice though the low variability of the agent trajectories indicating the simple local rules of the source data.

Different Input Data This scenario was also run with different input datasets to demonstrate adaptability (Figure 4.19). In the case of the *lee-chat* dataset, agents end up in endless chatter (well, actually more or less 25 minutes of chatting with agents leaving conversations at different

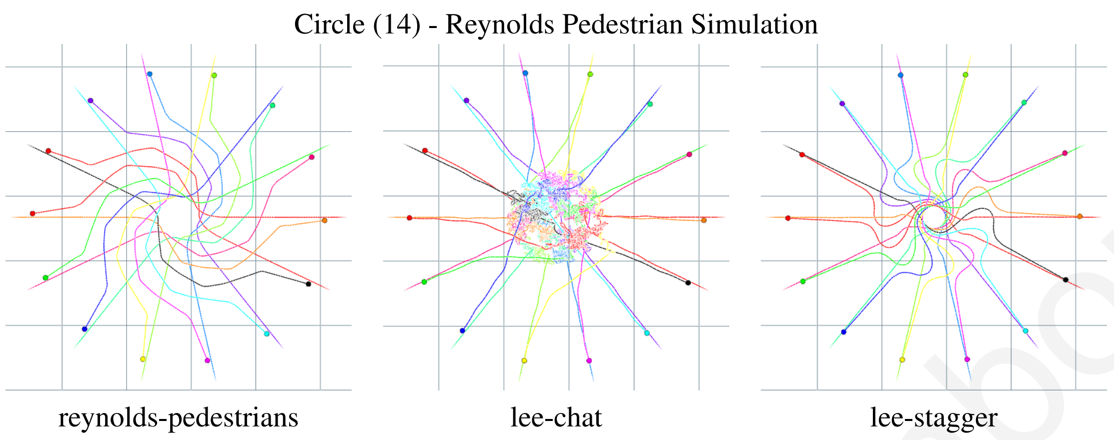


Figure 4.19: 14 agents in a circle switch positions with their opposite agents using as input data from various data sources

parts of the simulation) whereas in the case of the *lee-stagger* dataset, trajectories end up deviating abruptly from the desired paths.

4.5.2.4 Butterfly

To demonstrate once again the subtle differences between the *lerner-zara* and *eth-hotel* datasets of pedestrians a butterfly like environment was simulated (Figure 4.20). This scenario displays also the higher level control that can be achieved using our method: a higher level controller can assign targets (runtime or not) to agents and agents try to move towards these goals using the PAG to handle any potential interactions. A number of agents (80) are assigned paths that form a butterfly like path. The *eth-hotel* dataset results in a simulation with larger variations in the trajectories; i.e., interactions end up having larger to the desired path whereas the behaviour derived from the *lerner-zara* dataset is more uniform and interactions do not deviate much from the paths.

We point out here the fact that the input data had pedestrians walking in a single street; roads with intersection were not present and most certainly the input data never consisted of 80 pedestrians walking at the same time (in the limited view we are tracking). The PAG framework; i.e the

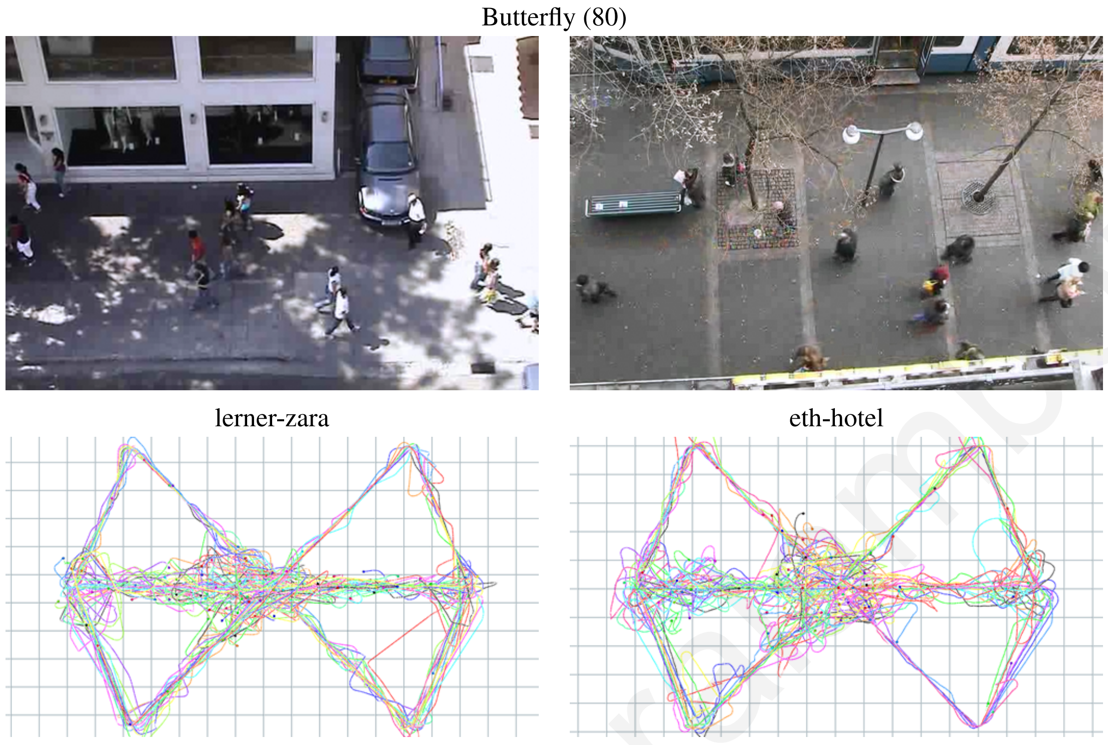


Figure 4.20: Trajectories for agents moving in a ”butterfly” like formation.

PAG, the IFDB and the Optimization Function managed can capture the local dynamics of crowds and apply them in novel environments.

4.5.2.5 Manhattan Inspired Environment

A relatively large crowd of 300 agents is simulated (8000 frames of simulation time) in a Manhattan like environment with city blocks and streets occupied by pedestrians (Figure 4.21). By using the *lee-chat* dataset, the resulting trajectories are closer to each other and ”chatting hotspots” between simulated agents emerge resulting in a city where nobody works and everybody is gossipping. The *eth-hotel* dataset on the other hand, which is a dataset of pedestrians in a busy street, results in wider groups of trajectories that cover most of the street areas with some minimal

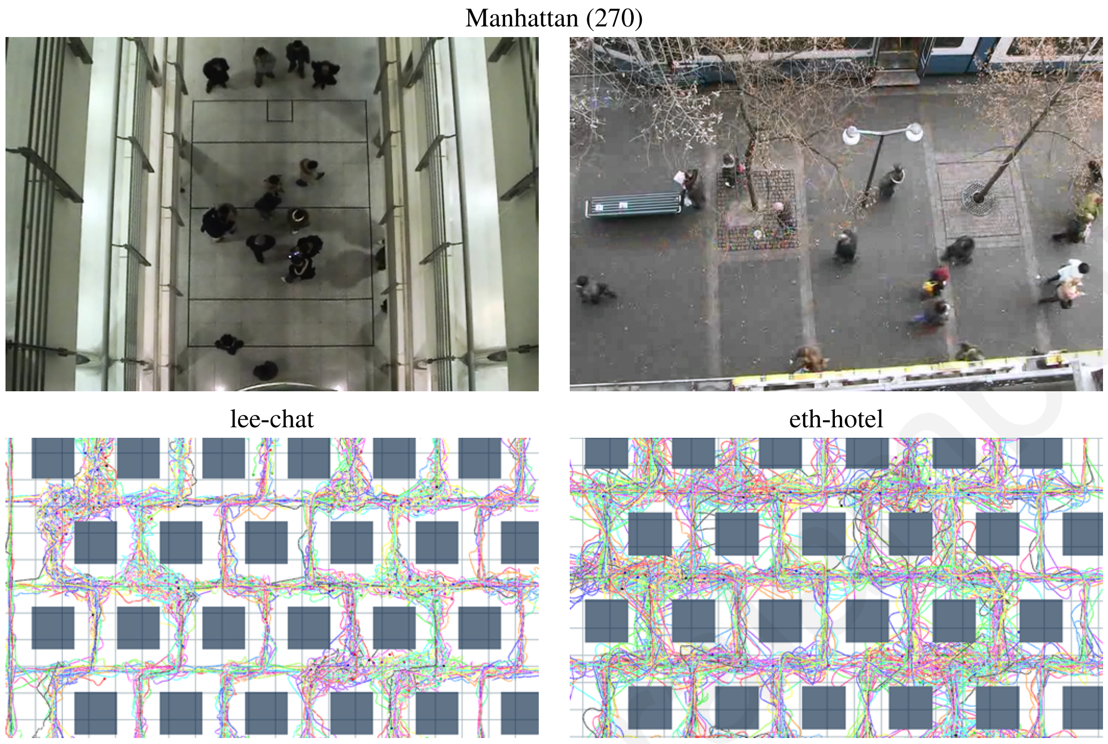


Figure 4.21: Agents moving and interacting in a city blocks like environment. Please consult the accompanying video for the simulations.

chatting; i.e., the simulation adapts to the input data has both people walking and people standing still and talking.

4.5.2.6 Complex Paths

To demonstrate once again that the proposed framework can be controlled from a higher level controller, agents were assigned to write some text. Paths were assigned on the strokes of the text and agents moved on them, interacting with each other whichever way the data dictated.

Computer Graphics Forum – CGF A number of agents (90) in these 5000 frame simulations try to stay on paths that form the *CGF* logo whilst at the same time interacting with each other with varying results depending on the input data (Figures 4.22 and 4.23). The *lee-chat* dataset generated a thinner logo than the *eth-hotel* dataset with chatting areas visible.

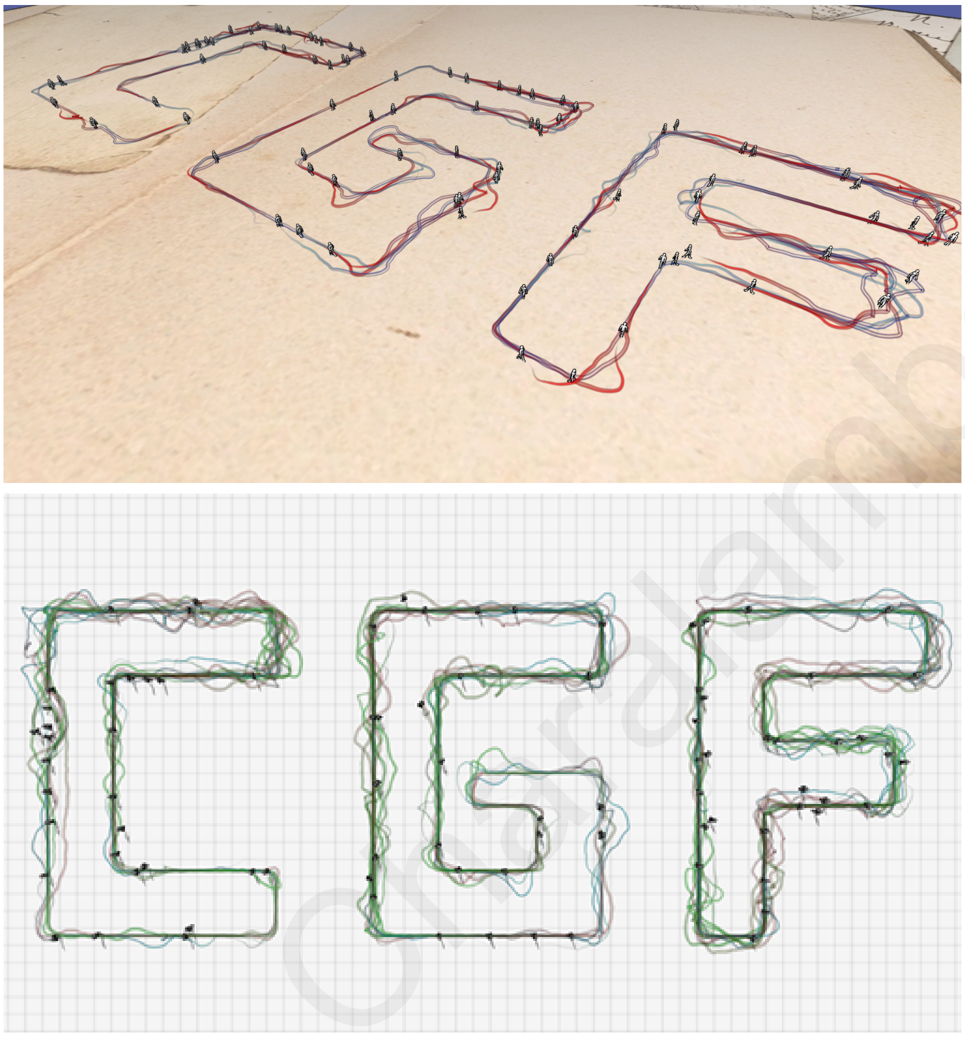


Figure 4.22: Trajectories for 90 agents forming three letters (CGF) using as input the *eth-hotel* dataset. Please consult the accompanying video for the simulations.

Hanzi 100 Agents form the word ”Love” in Chinese Hanzi, using as input the *eth-hotel* input dataset (Figures 4.24 and 4.25). Notice that even though there is a large number of interactions amongst the agents, the derived trajectories approximate the word, indicating that the agents aim to follow their higher level goals.

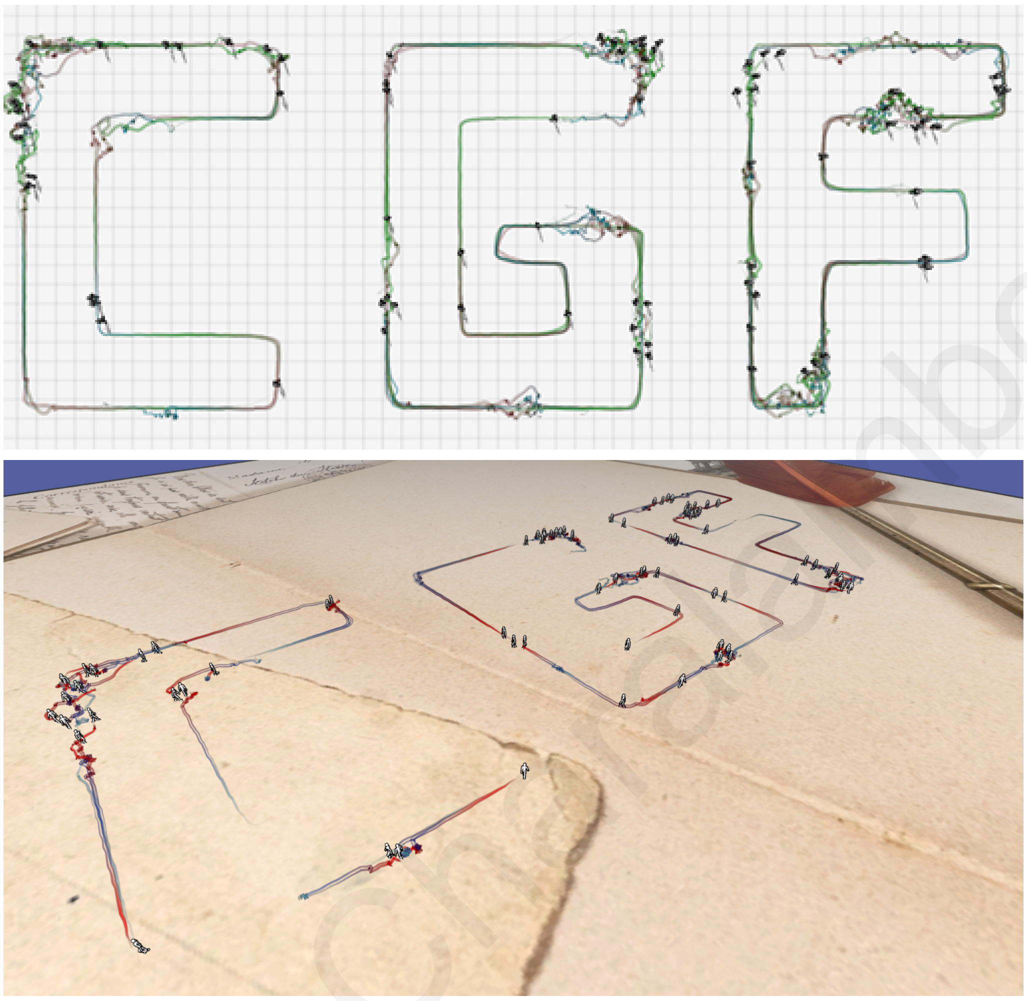


Figure 4.23: Trajectories for 90 agents forming three letters (CGF) using as input the *lee-chat* dataset. Please consult the accompanying video for the simulations.

4.5.3 Simulation Update Step

In order to have a smooth result the update phase of the simulation needs to be small. As the agents compute a new trajectory segment more regularly, they become more responsive to changes that happen around them and they make smaller deviations (Figures 4.26 and 4.27). However, in a traditional data-driven simulation there is a fixed cost per query and as the frequency of the simulation increases, the overall cost increases linearly. In the PAG method the cost of graph traversal is usually much smaller than the initial query to enter the tree; the performance cost

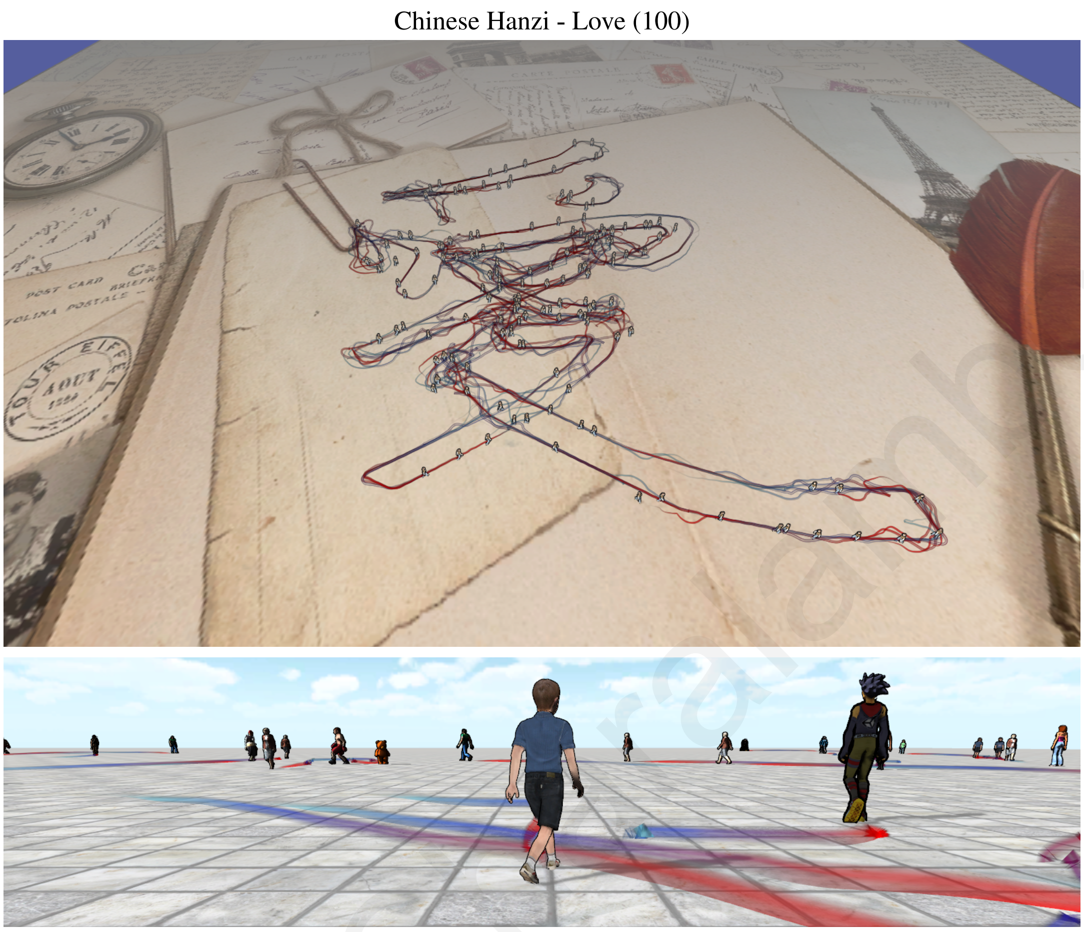


Figure 4.24: Trajectories for agents forming the word ”Love” in Chinese Hanzi. The image on the bottom is a closeup of the same simulation (with a different floor). This demonstrates that higher level control can be added over the data driven framework proposed in this thesis.

comes from the edge selection mechanism and not nearest neighbor searches which are quite slow. Increasing the size of the input data might increase the average edge count, but this happens in a much slower rate and cost increases by a much smaller rate. The input for the presented example was the *lerner-zara* dataset and the simulation environment was a similar sized rectangular area with 30 agents entering from locations similar in position as the input data.

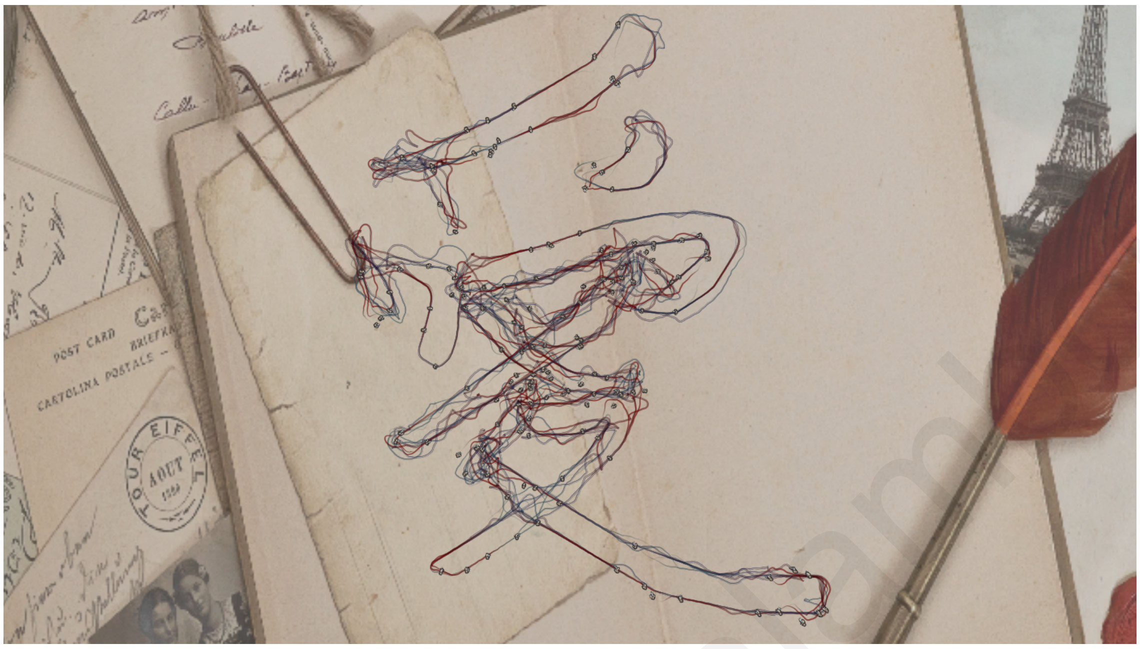


Figure 4.25: Trajectories for agents forming the word ”Love” in Chinese Hanzi. Please consult the accompanying video for the simulations.

4.5.4 Speed vs Quality

In what follows, a new trajectory segment for each agent is fetched after every 5 simulation frames in a 25 frames/sec simulation environment. In traditional data-driven techniques at each simulation step every agent generates a query describing its current state and searches an example database for the best matching example (Algorithm 4.1). Usually databases are stored in a spatial acceleration structure such as a kd-tree which has $O(log_2 n)$ performance, where n is the number of examples in the database. If the data is high dimensional though, the kd-tree approach tends to $O(n)$; i.e., it tends to exhaustive search of the data, unless the number of examples is very high (if N is the number of examples in the database, then $N >> 2^k$).

In the proposed method, a kd-tree is also used in a similar way but only at the beginning of an agent's interaction (see Section 4.4), i.e., whenever an agent senses some stimuli in its sensing area to find the best starting node in the PAG to handle the interaction. Approximate methods

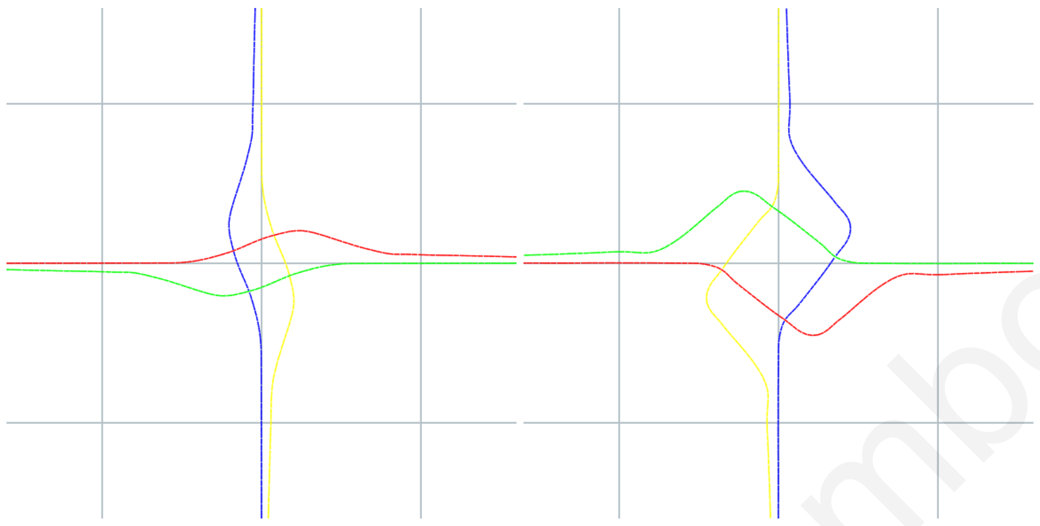

(a) Period = 5 frames (b) Period = 10 frames

Figure 4.26: Increasing the period from 5 to 10 frames generates a more unresponsive simulation with slightly different behaviour (Opposing Scenario trajectories) - the agents sense interactions more or less at the same time but because they use different future predictions (5 against 10 frames) the resulting simulation is different.

such as Approximate-Nearest Neighbor (ANN) and FLANN are usually followed that typically sacrifice quality over speed. This differs from our approach in the sense that by using the PAG, the example data are interlinked to have semantic relationships between them; a data sample is usually retrieved after an edge traversal which implies that some other data samples were retrieved before it in an order implied by the source data). For all subsequent simulation frames, the agent just traverses the PAG until interactions are resolved. The number k of (state, action) pairs that are returned by the kd-tree query has a big effect on the speed and the quality of the methods since all these k examples are used for action selection either by selecting one of them or by combining them.

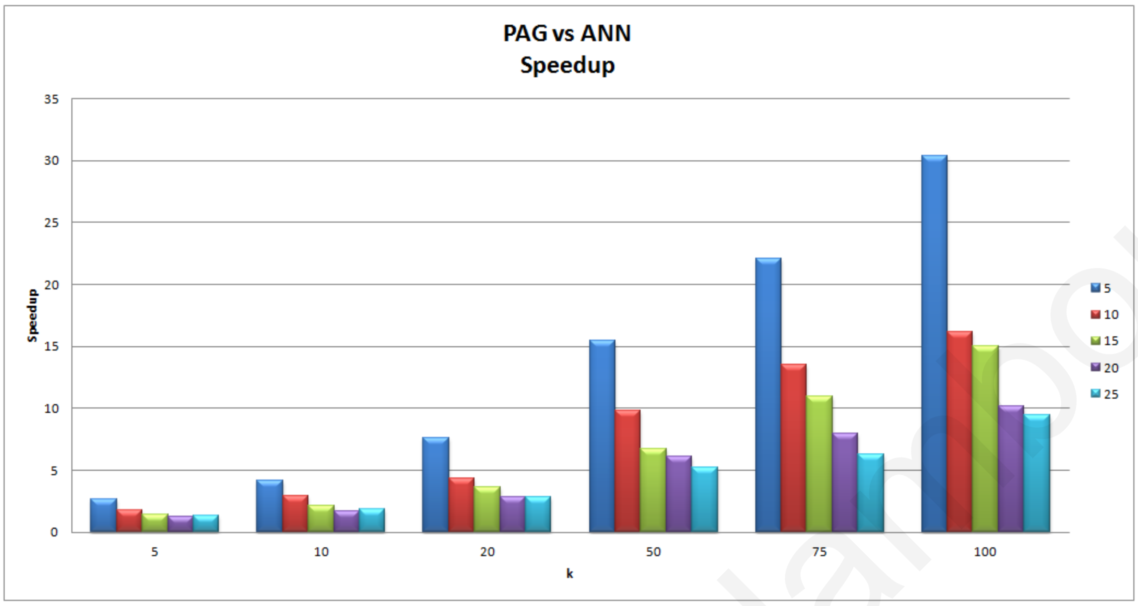


Figure 4.27: Comparison of the speedup of the PAG method with different update rates (ranging from 5-25 frames on a 25fps simulation) over the corresponding kNN approach with the same update rates. Increasing the period over which the simulation is updated, the PAG method outperforms the $k-NN$ one by a large margin.

Sensitivity to k

During selection of the nearest neighbor, a number k is typically selected that corresponds to the number of nearest examples to be retrieved from the database. This number influences the run time of the corresponding algorithm, especially in the k-nearest approach where all the retrieved examples are processed to further decide on actions.

In the proposed scheme, we aim in reducing the number of examples that are retrieved at every simulation step; i.e., this is achieved indirectly by the edges since these data are both preselected to be as close as possible and to have ordering between them. To measure the performance improve of the PAG over the kNN approach, we ran several scenarios, using different input data with different parameters and varying number of nearest neighbors (k) examples retrieved. Additionally, each experiment was ran multiple times and the timing results were averaged out to remove any noise

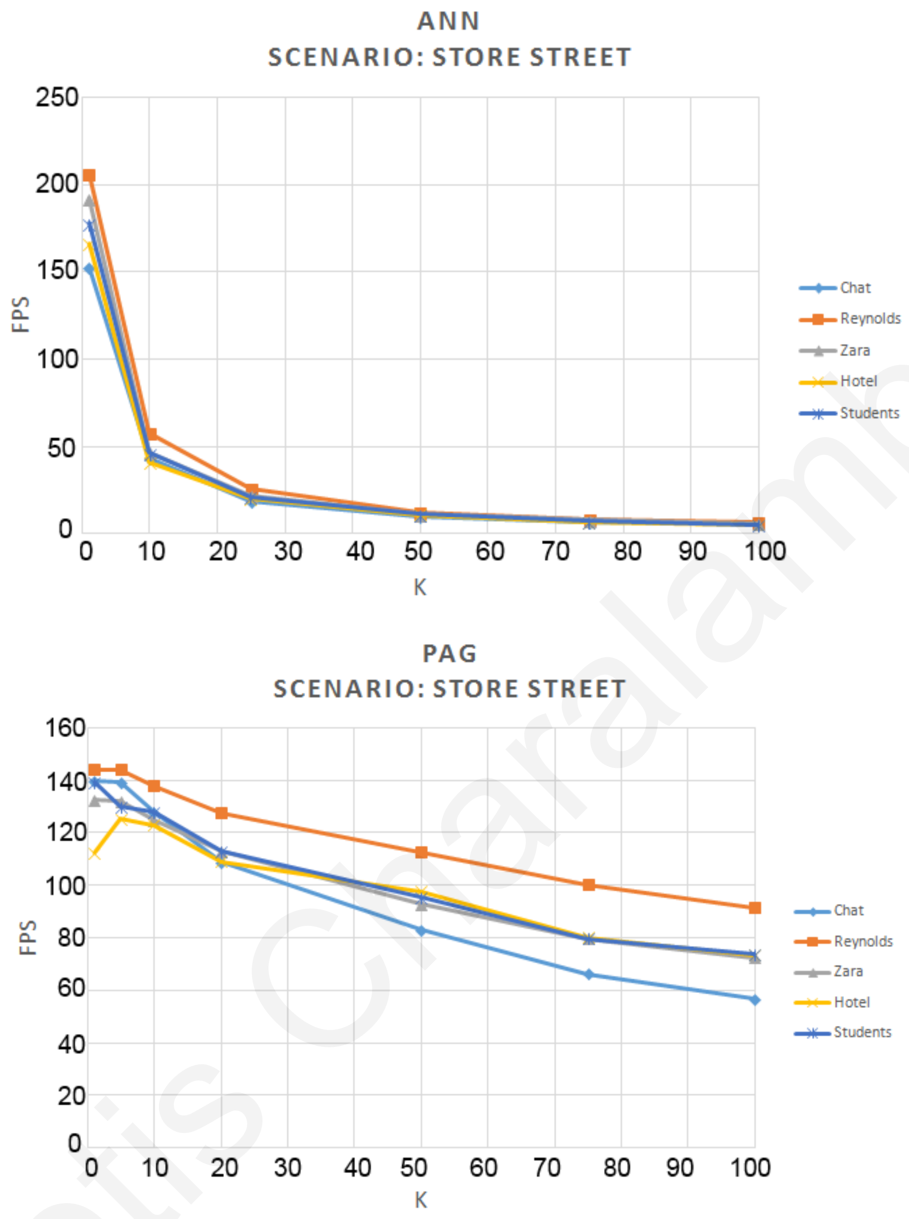


Figure 4.28: Timing comparison between the PAG and the kNN approach for a simulation of 32 agents in a busy street.

from the data. Even more so, we tested the PAG framework using the *jump thresholds* as described in Section 4.4. As we discussed, this threshold can be considered as a weight between selecting the PAG traversal on the kNN approach. A jump threshold value of 0.5 indicates that an agent

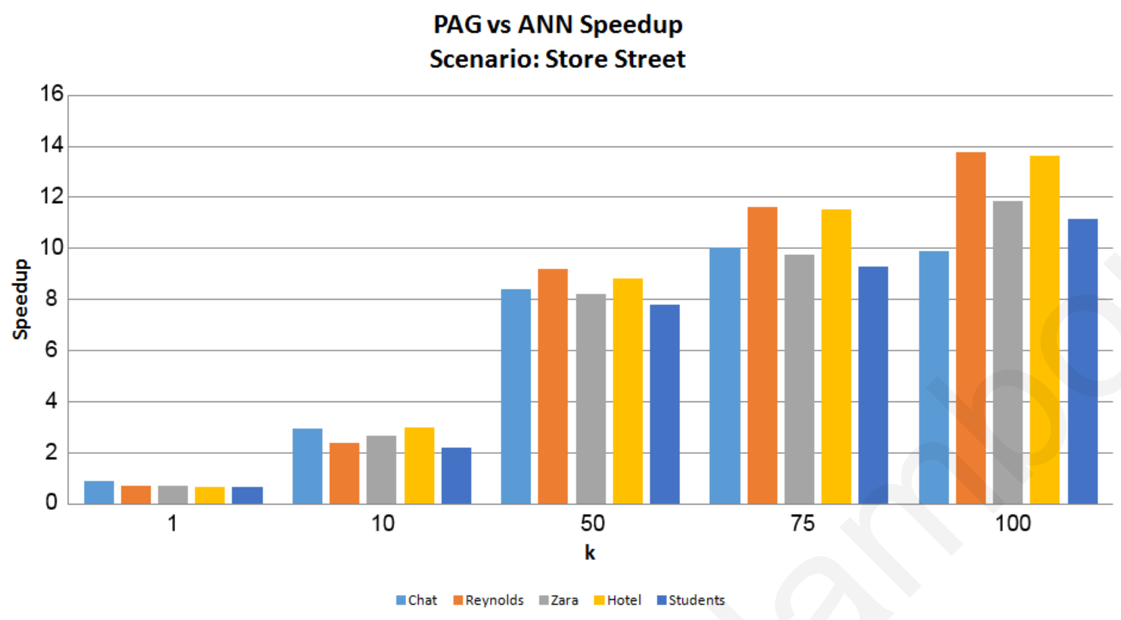


Figure 4.29: Speedup of the PAG over the *Full kNN* approach – PAG outperforms it by up to $x14$.

should not traverse to the next edge on the graph, if its current state (TPP) differs by 50% from the state stored on the currently visiting node.

kNN Algorithm Two approaches can be considered for the kNN simulator: a *Full kNN* approach where at every simulation step, each agent searches the database of TPPs even though its state is empty and *Partial kNN* where an agent searches the TPP database only when having a state with possible interactions, and for all other times (interaction free) it utilizes the IFDB as in the PAG. For most of the experiments described here, the *Partial kNN* approach was considered. We note that the *Full kNN* approach is significantly slower the PAG and the Partial kNN approaches. An example of simulation performance of the *Full kNN* against the PAG with different datasets can be seen in Figures 4.28 and 4.29. Notice that the PAG performance remains in more than double the real time frame rate for large k, whereas the *Full kNN's* performance drops below the real time mark for $k > 20$, even on this small scenario with just 32 agents.

Timing Performance As expected, the full PAG (i.e., PAG without jumping) outperforms all the other approaches by up to an order of magnitude. Figures 4.30 up to A.6 show the comparison in speed gains and collision quality over the kNN approach as k changes. On the top row of each figure, the left graph shows the absolute simulation frame rates that can be achieved by the method whereas the right one shows the relative speedup to kNN. A relative speedup over one indicates better performance. The absolute values in frames per second do not matter that much as the speedup achieved since the implementation could definitely benefit from a different programming environment, multi-threading and a GPU platform. Even under these conditions though, the proposed method can achieve more than real time performance ($> 25fps$) for a relatively large number of agents (150-180 in the graphs shown) even with a large number of nearest neighbors.

Collision Quality On the figures' bottom row, the collision quality is shown on the left; i.e., every how many simulation frames a collision occurs (higher indicates more rare collisions) and the relative collision quality is shown on the right. A relative quality near one or greater indicates good relative quality (*as compared to kNN*) but not necessarily good behavioural quality overall; collisions might be required if they exist in the source data also. The purpose behind the collision quality test is to test that under the same conditions and same configuration, the simulations act similarly; even if they are not perfect.

Comparison over different datasets and scenarios

Two scenario cases were considered, both with dense crowds:

- *Manhattan* with 180 agents (Figure 4.21) which basically simulates a city like environment
- *Chinese Hanzi* with 150 agents (Figures 4.24 and 4.25) which simulates agents in a more complex pattern.

Different datasets were used as the input of the simulations for both the PAG and the kNN approaches. The PAG outperforms the kNN approach in all the cases with performance increasing over k, with almost the same collision quality except on the *lee-chat* dataset. Also, a careful examination of the quality graphs indicates that the PAG is more stable as the number of examples is retrieved in contrast to the kNN approach indicating a more stable algorithm; edge traversal fixes wrong initial node selections. Details for a representative simulation are presented in the following paragraphs. A more complete and elaborate analysis for different scenarios can be read in Appendix A.

Students (lerner-students)

This is the largest real world dataset used in our experiments. As such, the PAG graph is of good quality with good connectivity and the resulting collision quality is similar to the kNN approach for both scenarios (Figures 4.30, 4.31) – ranging from -20% up to 60% quality improvement. The highest improvements (40% and 80%) are for $k = 1$, i.e., when we search for the nearest neighbor. Even though the PAG doesn't offer much performance improvements at this value for k, the quality seems much better; this is due to the action selection mechanism of the PAG against the kNN. Also, high jump thresholds that benefit frequent jumps on nodes of the graph due to low errors are very similar to kNN; both in speed and quality. As far as speedup is concerned, the PAG vastly outperforms kNN by 700-800% at 100 nearest neighbors.

Quality vs Performance

It's clear from the example described in the previous paragraphs and the ones in Appendix A, that as far as performance speedup S is concerned, PAG clearly outperforms kNN. If simulation quality Q (quality is debatable if collisions is the only thing that is considered) is important, one should weight out the benefits in order to select firstly the algorithm (PAG, PAG with jumps or kNN) and the k-best number of examples to be retrieved every time it's needed (node selection for PAG or example selection for kNN). One potential solution to that, is defining a performance measure $p(Q, S)$:

$$p(Q, S) = w_Q.Q + w_S.S \tag{4.4}$$

where w_Q and w_S define weights for the quality and performance speedups respectively.

4.6 Discussion

In this chapter the Perception-Action Graph was presented as a means to simulate realistic steering behaviours for crowds efficiently. As we showed in Section 4.5, the proposed method outperforms the simple kNN approach to data-driven crowd simulation and has similar (and some times better) collision quality. We also evaluated the approach by visual inspection. Clearly, one cannot just evaluate the quality of a crowd simulator (data-driven or otherwise) by collisions alone; for example a simulation with agents that just repel each other of stop abruptly resulting in a collision free simulation might have "better" collision quality but the resulting simulation looks like nothing in real life data. For these reasons, a multi-criteria approach to evaluating simulated crowds iin the presence of some reference data such as the ones in Table 4.1 is presented in Chapter 5.

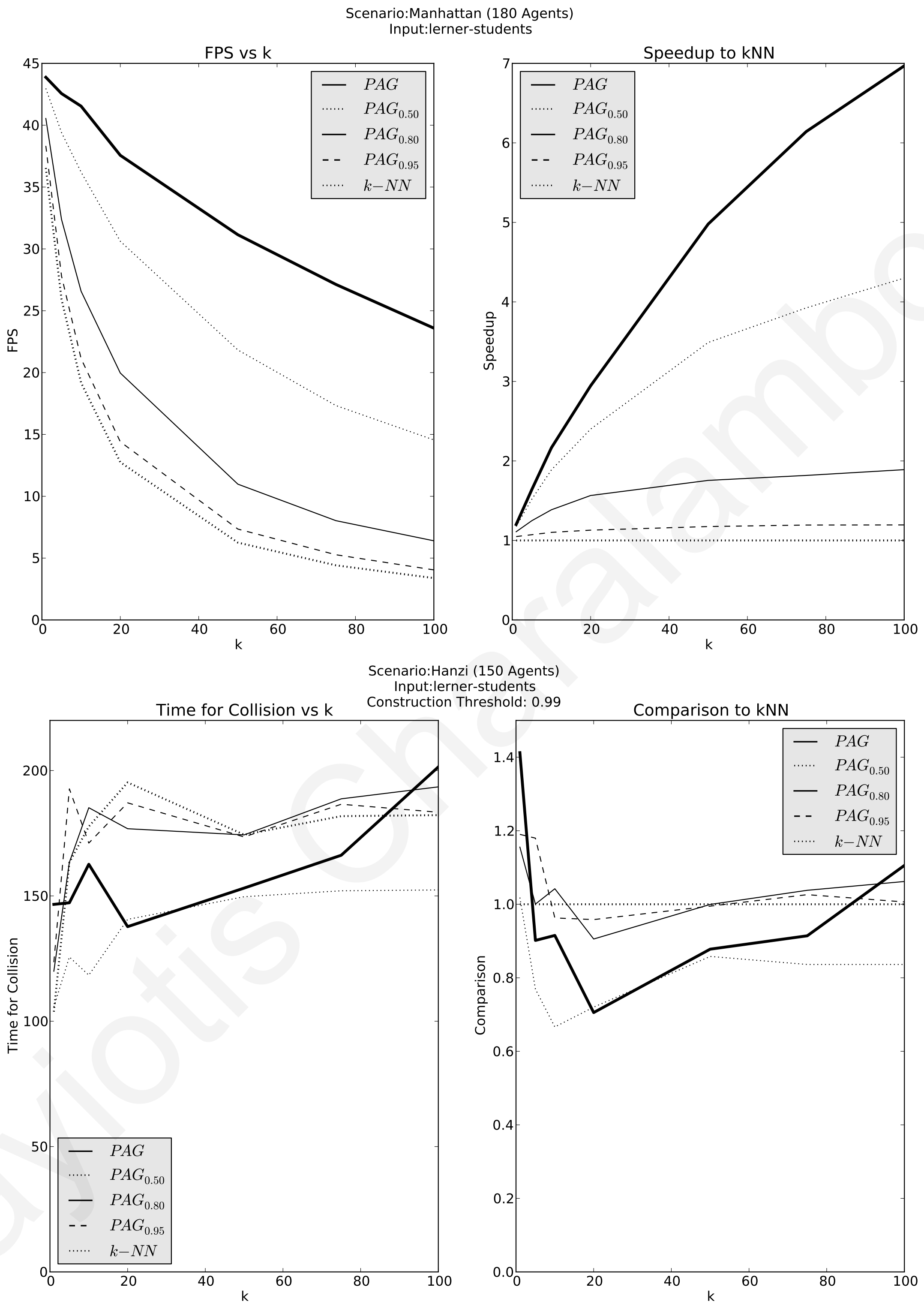


Figure 4.30: Comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 180 agents in a scenario similar to Section 4.5.2.5 using as input the *lerner-students* dataset.

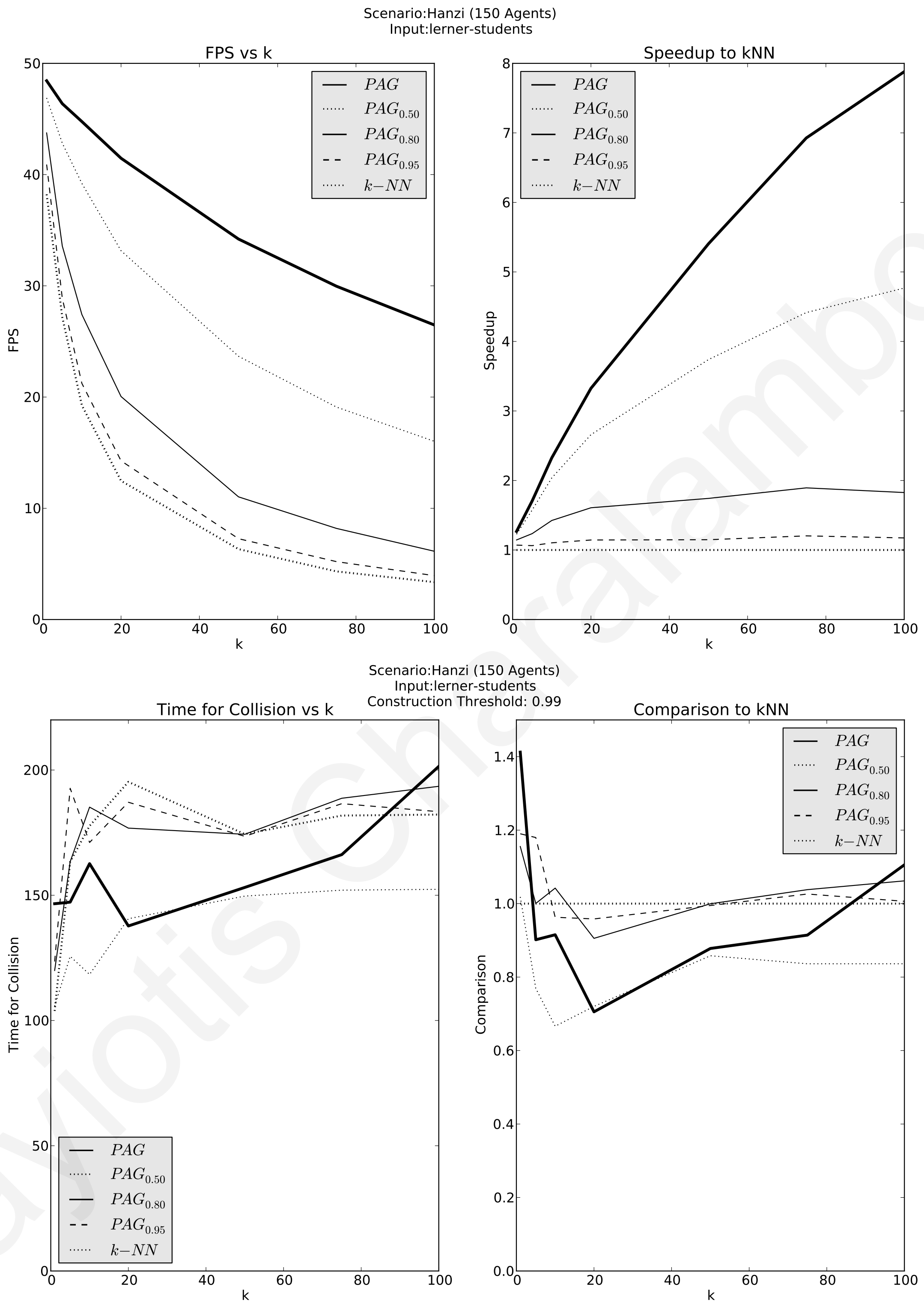


Figure 4.31: Comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 150 agents in a scenario similar to Section 4.5.2.6 with the lerner-students dataset.

Chapter 5

Data Driven Evaluation of Crowds

"Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid."
(Albert Einstein)

5.1 Introduction

Computer generated crowds are nowadays commonly used in films, video games, on-line communities and virtual environment applications (Section 5.1). Realistic simulation of such crowds is an important factor in the level of user immersion and the value of the conclusions one can draw from these simulations. Over the past twenty years, the field of computer graphics has experienced a dramatic increase in the number of tools, approaches and algorithms focusing on creating compelling crowd motions. Currently, several companies specializing in offline modeling and rendering offer tools to author crowd simulations [12, 18, 20, 57] and more recently several game engines have also recently developed modules to assist in modeling crowds [101]. As more approaches to generate virtual crowds are introduced, developing better methods to analyze, evaluate, and improve the quality of these simulations is becoming more important.

Complexity of crowd analysis and evaluation The complexity inherent in a simulation involving multiple interacting characters, makes evaluation a non-trivial, challenging problem. Some issues are easy to identify and measure such as characters colliding with each other or the environment, or characters going much too fast. Other issues are more subtle: e.g., jamming at narrow passages or unnecessary backtracking. Some crowd simulation systems avoid these types of problems, however the result might still not be satisfying to the eye. A crowd in a city, where each character walks directly to its target in the most efficient and "proper" way might end up looking robotic and spiritless; something that is completely different from what happens in real-life crowds where many phenomena appear such as people walking in groups, chatting, wandering from shop to shop and so on.

Evaluating crowds is currently a difficult and often neglected issue as far as simulation is concerned. Deciding on what is a correct simulation is most of the times a biased problem that depends on the point of view of the viewer. Additionally, evaluating crowds should take into account both local (micro) and/or global (macro) characteristics such as local collision and implicitly generated lane formations respectively. Finally, simulating a crowd typically involves a lot of components (as described in the previous chapters) and therefore care should be taken when evaluating them; i.e., an algorithm for behaviour should be judged solely on that ability and not other aspects such as rendering. There is still no acceptable generally accepted methodology in the crowd simulation literature for evaluating the ability of a simulator on capturing required behaviours.

Approaches Typically three approaches are followed to evaluate crowds: user based, statistical and data-driven. User based evaluations [74] is the typical approach since human beings are accustomed to real life crowds in their everyday lives and in the end they are going to be the final users/viewers of the system. This approach though pose some problems due to this human bias to crowds; i.e., if the task is evaluating the crowd's characters capability on navigation and the characters do not have human like appearance with human like animations, users might not rate the steering quality of the simulation correctly due to their real life biases. This means, that an approach where the appearance is abstracted (in the form of discs for example) should be selected to remove the bias, even though this representation might be misleading in the character's ability to have a plausible walking animation to achieve that steering behaviour. Statistical approaches on the other hand, try to measure some characteristics of the characters or the crowd as a whole such as speed histograms of people, interpersonal distances or distance to goal and then rate a simulation's quality in different scenarios such as the SteerBench suite [93]. This approach is still problematic though, since statistics are typically not linked to real world experience and the ranking could be objective to the selection of statistics. This is where data–driven techniques come in place, where user based experience (or real world data to be precise) are linked to the statistics. In these methods, simulations are compared to real world data such as videos of crowds or some user defined sketches of the desired behaviours. Some measures are defined and estimated on the training data and simulation data are tested on their capability of replicating these behaviours/measures.

To this extent, we propose a data-driven multi-criteria evaluation framework which compares a crowd simulation to user supplied data such as space/time trajectories of real life people. Here we are only considering steering behaviours of crowds; even though this approach can be adapted for other components also such as high level behaviour patterns.

Data-Driven Evaluation More specifically, a novel data-driven approach for analyzing the quality of crowd simulations is presented. This approach is data-driven in the sense that it takes as input a set of user-defined metrics and training data, either synthetic or from video footage of real crowds and evaluates new testing data on their resemblance to the input. Given a simulation, the crowd analysis problem is formulated as an anomaly detection problem and state-of-the-art outlier

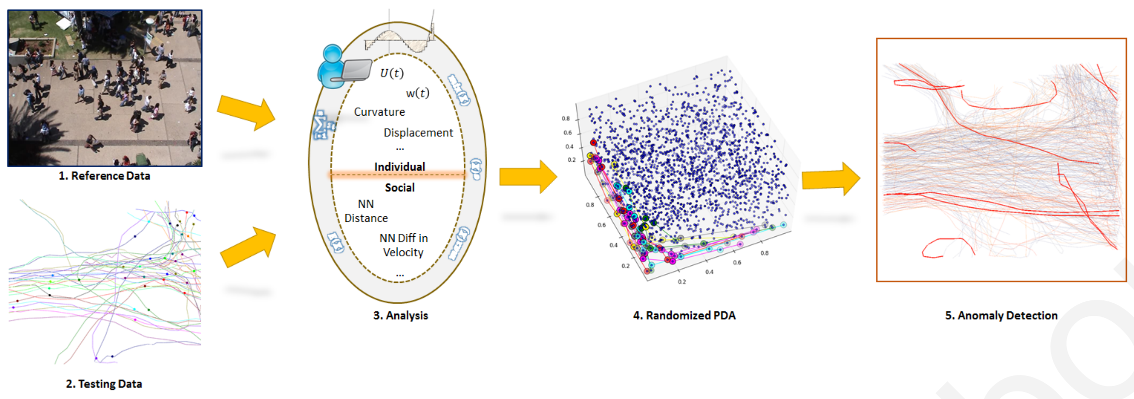


Figure 5.1: **System Overview** Reference data (1) from real crowds or relevant simulations are compared against testing data (2) from simulations the user wishes to analyze. Using a variety of metrics (3) the proposed framework automatically detects outliers at interactive rates using a randomized Pareto Depth Analysis (PDA) approach (4). The resulting analysis is then shown as a heatmap with the most anomalous paths in the simulation shown in bright red (5).

detection algorithms are exploited to address it. To that end, the concept of Pareto Optimality is employed (Section 5.3.1.1) due to its unique ability to automatically detect outliers under multiple evaluation criteria. This allows us to capture potential erroneous behaviours that are difficult to detect with conventional approaches. We demonstrate the applicability of our approach through several examples of simulated crowds produced by different techniques including simulations from commercial games.

Multiple criteria evaluation An important problem in detecting anomalies is to determine an appropriate metric to compare the simulated data to the reference data. However, rather than attempting to determine a single best criterion for evaluating a simulation, the proposed framework allows users to specify multiple criteria. We then use a recently proposed multi-criteria anomaly detection approach, the *Pareto Depth Analysis (PDA)* approach [29], to robustly compare simulations against real crowds. Unlike previous approaches for handling multiple criteria, PDA does not require users to choose arbitrary weights (or some times brute force weight searching) to balance the importance of these criteria. For example one could give more weight on a collision avoidance metric rather than a speed metric to emphasize more on collision free simulations rather than completely similar speed patterns to the reference data. Instead, the proposed method naturally defines outliers based on the concept of Pareto optimal fronts while at the same time scaling linearly with the number of evaluation criteria.

5.1.1 Contributions

A new data-driven crowd analysis framework is introduced, that formulates the problem as anomaly detection and applies Pareto Depth Analysis (PDA), a state-of-the-art anomaly detection technique from the field of machine learning. PDA is uniquely appropriate for the crowd analysis problem due to its ability to handle multiple criteria in a robust way. The main advantages of the proposed approach are:

- It enables unsupervised anomaly detection, i.e., there is no requirement on manually labeling the reference data with any additional information and therefore correct behaviour is implicitly defined by the data.
- By comparing a simulation against a known dataset it can identify both atypical behaviours as well as missing features from the simulation (Novelty Detection).
- Given a simulation consisting of mostly nominal behaviours, atypical behaviours (that is behaviours that differ from the mostly nominal ones) can be identified by using the simulation itself as reference (Outlier Detection).
- Through combining individual and social (or inter-personal) evaluation criteria, it allows the identification of issues related to the social behaviours of the agents (such as groups), which are normally much harder to detect.

- Through the introduction of randomized PDA, a variation of PDA, the analysis of crowd simulations is enabled at interactive rates since the original PDA algorithm is quite slow [29].

The rest of this chapter is organized as follows. In Section 5.2, the proposed crowd analysis framework is presented. Section 5.3 details the application of Pareto Depth Analysis and other anomaly detection algorithms in our framework. Section 5.4 shows the applicability of our framework on existing crowd simulation methods using both synthetic and real life examples. Finally, some conclusions and plans for further research are discussed in Section 5.5.

5.2 Method Overview

Given as input a set of reference data (real or simulated) that captures typical or desired crowd motion and a set of user selected evaluation criteria an outlier detection algorithm is trained which can then be used to filter any given testing set of trajectories. These data consist of space/time paths (trajectories) that track the positions of agents in time. Paths (or more precisely segments of these paths) that are found to be anomalous are then highlighted for the user using a heat map approach for further investigation. By using segments of the paths, the proposed approach works on local behaviours of each individual agent in the crowd; examining both its own behaviour and its relationship to neighboring agents that potentially affect its behaviour. As such, this method of simulation analysis can be viewed as an anomaly or outlier detection problem. [1] An overview of the proposed framework is shown in Figure 5.1.

As such, if the input is the testing data itself, the proposed framework acts as an *outlier detector*; i.e., it find parts of the simulation that differ from the general behaviour of all agents given a

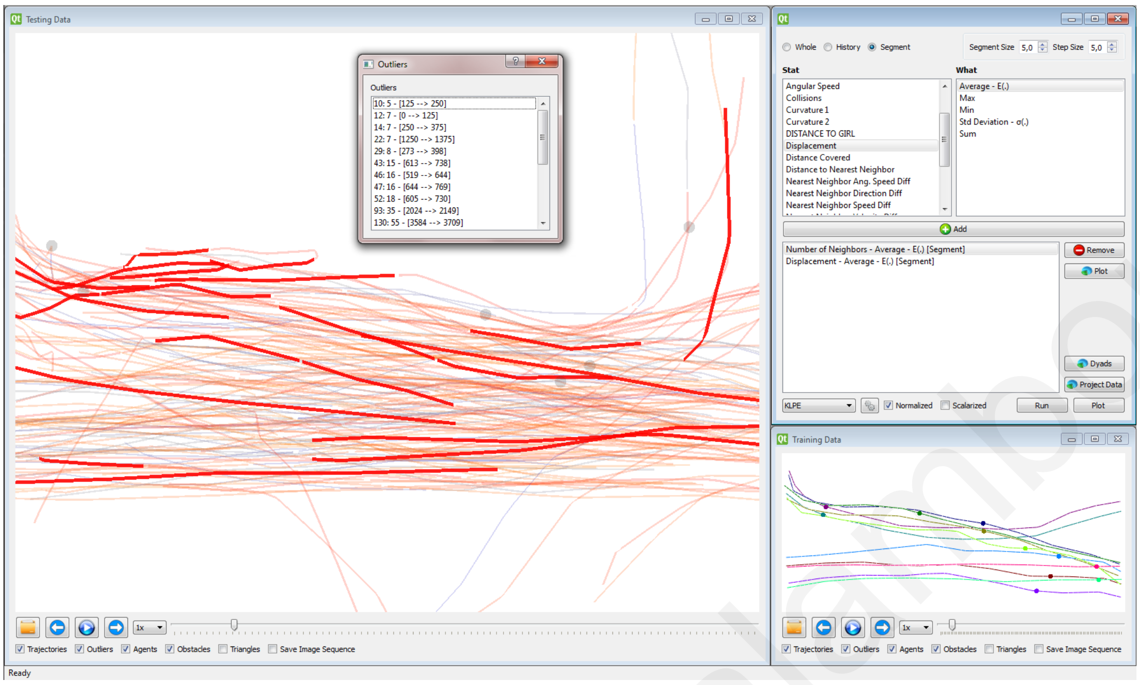
[1]Outlier detection refers to identifying anomalous data without any prior knowledge of which of the given data are nominal or not, whereas novelty detection algorithms try to classify new data given a set of data that is known to be nominal.

set user defined metrics. If on the other hand, different data are used from learning and testing; i.e., the learning data describe the behaviour that we want to have in the tested simulations, the system acts as a *novelty detector* and tags paths (or parts of them) as anomalous or not; accompanied with an anomaly score. In general, we assume that the reference data provided by the user contains mostly nominal behaviour. This assumption allows us to use unlabeled data, and avoids the need for a user annotating paths into good and bad.

5.2.1 Comparison Metrics

When training the proposed anomaly detection method, a user must specify a set of metrics to use for the evaluation. Following the approach presented in [93], metrics are split into two parts: a *difference/dissimilarity measure* and an *operator* (Figures 5.1 and 5.2). A difference measure seeks to capture numerically how far the state of the simulation agent is from a given agent in the reference data at a given timestep (examples include difference in speed or acceleration). An operator then aggregates the effect of a difference measure over many time steps (e.g. average, sum, min or max). Combining an operator with a difference measure will provide a complete metric, for example two paths may be compared based on their average speed or maximum acceleration.

Personal vs Inter-personal Measures Broadly speaking, the proposed approach can utilize two high-level types of difference measures: individual/personal and social/inter-personal (Figure 5.3). Individual similarity measures are based on properties of each agent in isolation. Examples include an agent's speed, acceleration, total displacement, path curvature, and other criteria that can also be defined as a function of the agent's path over time. In contrast, inter-personal metrics seek to capture an agent's relationship to its neighboring agents at any given timeframe. For example, the average distance to an agent's nearest neighbor may help reveal outliers in density. Likewise, the difference between an agent's velocity and the velocity of its nearby neighbors



Outliers

10: 5 - [125 --> 250]
12: 7 - [0 --> 125]
14: 7 - [250 --> 375]
22: 7 - [1250 --> 1375]
29: 8 - [273 --> 398]
43: 15 - [613 --> 738]
46: 16 - [519 --> 644]
47: 16 - [644 --> 769]
52: 18 - [605 --> 730]
93: 35 - [2024 --> 2149]
130: 55 - [3584 --> 3709]

Whole ○ History ○ [✓ Segment] — Segment Size 5,0 Step Size 5,0

Stat	What
Angular Speed	Average - E(.)
Collisions	Max
Curvature 1	Min
Curvature 2	Std Deviation - σ(.)
DISTANCE TO GIRL	Sum
Displacement	
Distance Covered	
Distance to Nearest Neighbor	
Nearest Neighbor Ang. Speed Diff	
Nearest Neighbor Direction Diff	
Nearest Neighbor Speed Diff	

Number of Neighbors - Average - E(.) [Segment]
Displacement - Average - E(.) [Segment]

KLPE — [✓] Normalized — [ ] Scalarized

Figure 5.2: **Anomaly Detection Tool** Screenshot of the proposed tool. A user selects if he/she wants to work on *segments* or the *whole* trajectory, the *difference measures* such as number of neighbors and displacement, *operators* such as average and the algorithm (kLPE, PDA, k-NN and One Class SVM are the ones used in this work). The proposed tool then highlights potential erroneous in the test data (left) behaviour as compared to the reference data (lower right).

can capture important aspects of social interactions between individuals (Figure 5.3) such as small scale groups.

Having both of these high level measures can help in identifying anomalies that couldn't be identified by just one of the two. For example, an input dataset of people walking in groups could have similar speed fingerprints as a simulation of agents walking alone and therefore it would be considered correct if no inter-personal difference measure is added.

In practice, we pre-calculate all the statistics (both individual and inter-personal) for each agent at the beginning of the process to improve on the run-time of the anomaly detection where only the differences for the selected measures need to be calculated. A list of similarity measures and operators tested in our framework are given in Tables 5.1 and 5.2 respectively.

Table 5.1: **Difference Metrics** The set of difference metrics used the proposed analysis framework. These metrics are divided into individualized and social metrics; with the latter referring to the relationship of the examined agent to its nearest neighbor or its neighborhood (i.e., a radial region around the agent).

Difference Measures	
Individual	Inter-personal (Social)
Speed	Nearest Neighbor Speed
Angular Speed	Neighborhood Speed
Acceleration	Nearest Neighbor Angular Speed
Angular Acceleration	Neighborhood Angular Speed
Number of Nearest Neighbors	Nearest Neighbor Direction
Distance to Nearest Neighbor	Neighborhood Direction
Number of Neighbors	Nearest Neighbor Velocity
Curvature	Neighborhood Velocity
Distance Covered	Angle to Nearest Neighbor
Angular Displacement	
Displacement	
Collisions	

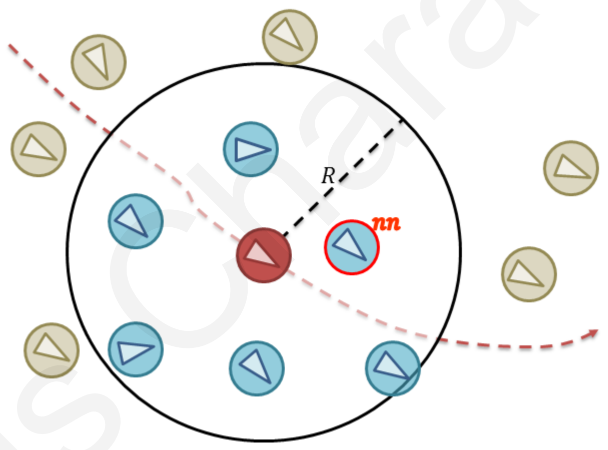


Figure 5.3: **Personal vs Inter-personal Measures** Agents in the radial region with radius R around the subject agent (red) are used in the inter-personal metrics (Table 5.1). The agent closest to the subject agent (marked as nn) is considered a special case and measures can be calculated relative to that agent also. Individual metrics consider only the subject agent.

5.2.2 Segments

Rather than analyzing entire trajectories at once, long trajectories can be split into smaller segments of equal temporal length. This allows for a finer analysis of the simulations so that local abnormalities are detected and pinpointed. These local abnormalities basically refer to local agent

Table 5.2: **Difference Operators** The set of operators used in the proposed framework. These are applied on the difference metrics in Table 5.1. For example, one could get the average difference in speeds of two space/time trajectories or the maximum curvature of all segments.

Operators		
Name	Symbol	Formula
Average	$\mu(X)$	$\mu(X) = \frac{1}{N}\sum_{i=1}^{n} x_i$
Standard Deviation	$\sigma(X)$	$\sigma(X) = \frac{1}{N}\sum_{i=1}^{n}(x_i - \mu(X))^2$
Sum	$sum(X)$	$sum(X) = \sum_{i=1}^{n} x_i$
Minimum	$min(X)$	$min(X) = x_j \iff \exists j \in [1, N]\text{where } x_j \leq x_i \forall i \neq j$
Maximum	$max(X)$	$min(X) = x_j \iff \exists j \in [1, N]\text{where } x_j \geq x_i \forall i \neq j$

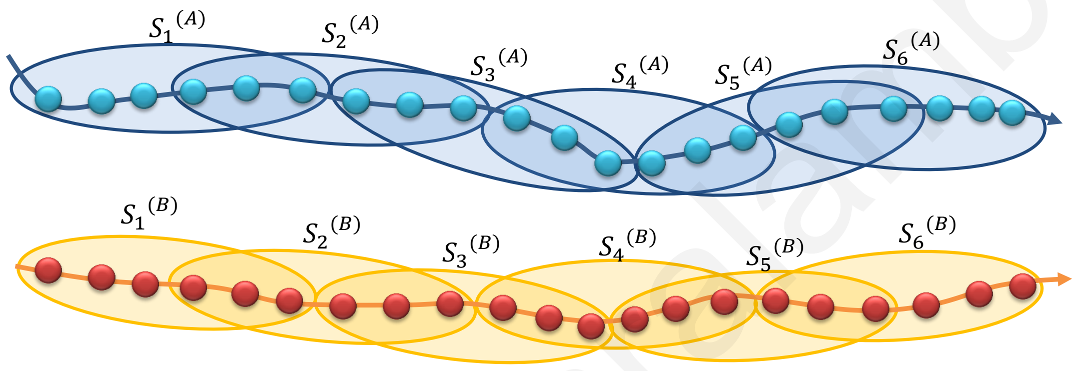


Figure 5.4: **Space/Time trajectory segments** Space/time trajectories are split into overlapping segments of equal length at equal sampling intervals to be analyzed by the proposed framework. Here for example we see two trajectories, split into segments of 6 samples (each sample differs some milliseconds from the previous one), retrieved every 3 samples – therefore each set of 3 samples belongs to 2 segments.

centric behaviours, contrary to what is being described in [29] where simple analysis with two measures (speed and shape) on whole trajectories in the same closed environment is performed.

In order not to miss anomalies which occur between segments we break up trajectories into overlapping segments; that is a segment for example can be of 5 seconds of length and segments could be retrieved every 1-2 seconds (Figure 5.4). Each of these segments can be treated as a unique trajectory for the purpose of detecting outliers. These trajectories are then evaluated via the user-specified metrics.

A segment can be described by both, local and global measures: for example one could describe a segment with the average speed of the segment itself and the average speed of the trajectory it belongs to. This adds a sense of personalization on the segment since it is part of something bigger; a segment of an agent that temporarily stops to avoid a potential collision (even though it's moving most of the time) is completely different than a segment of an agent that stops repeatedly.

Segment Representation Assuming l difference measures with each one denoted as m_j, $j \in [1, l]$, each segment of the n possible segments in the training data is represented as a vector:

$$\mathbf{s}_i = [{m_1}^{(i)}, {m_2}^{(i)}, \ldots, {m_l}^{(i)}]^T, i \in [1, n] \tag{5.1}$$

where the superscript denotes the segment and the subscript indicates the difference measure (notice that this notation is different than then one on Figure 5.4 where it is used for visualizing parts of trajectories – for the remainder of the chapter Equation (5.1) is the representation used). The set of segment representations $\mathbf{S}_T$ of the training data in essence represents the space of nominal behaviours and can be considered as a training database:

$$\mathbf{S}_T = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\} \tag{5.2}$$

Assuming a new testing segment $\mathbf{s}_t = \{{m_1}^{(t)}, {m_2}^{(t)}, \ldots, {m_l}^{(t)}\}$, $t \ni [1, n]$, we would like to see if it lies in the space defined by the nominal data $\mathbf{S}_T$. Typically, normalization is applied for each of the dimensions of the data to ensure relative differences.

5.2.3 Anomaly Detection

Once the simulation data has been divided into segments, reference data has been selected and the analysis metrics have been chosen, outliers can be found automatically using recently proposed machine learning techniques. For example, the k-LPE algorithm uses local p-value estimation

(LPE) based on a k-NN graph to provide outlier detection while maintaining a (user-specified) maximum false-alarm rate [110]. The proposed framework supports a number of different techniques in addition to k-LPE such as PDA [29], distance to the k nearest neighbor [3] and One Class SVM [87].

5.2.3.1 Distances between segments

Typically, the distance $d(i, j)$ between two samples $\mathbf{s}_i$ and $\mathbf{s}_j$ can found through their euclidean distance:

$$d(\mathbf{s}_i, \mathbf{s}_j) \equiv d(i, j) = \sqrt{\sum_{k=1}^{l} ({m_k}^{(i)} - {m_k}^{(j)})^2} \tag{5.3}$$

In the more general case, a different distance function for each of the difference measures could be applied (for example, we could have histograms as an additional operator in the operators in Table 5.2 and therefore distances between histograms should be considered). In this thesis, we address only scalar values since the operators return only scalarized values and therefore distance is simply defined by absolute or squared differences (euclidean). For completeness, in the general case, the distance between two samples $\mathbf{s}_i$ and $\mathbf{s}_j$ is defined as:

$$d(\mathbf{s}_i, \mathbf{s}_j) \equiv d(i, j) = \left(\sum_{k=1}^{l} d_k(i, j)^p \right)^{(1/p)} \tag{5.4}$$

Equation (5.4) is a generalization of the Minkowski distance of order p (*p-norm distance*) of segments $\mathbf{s}_i$ and $\mathbf{s}_j$. $d_k(i, j)$ is the distance between sample i and j on difference measure k (e.g. absolute difference, squared difference, histogram difference, etc). Setting p to 2 and the distance function as the absolute difference of the two segments we get the euclidean distance (Equation (5.3))

5.2.3.2 k-NN Based Approaches

The simplest approach to anomaly detection is comparing the testing sample $\mathbf{s}_t$ to its closest neighbors in the training dataset $\mathbf{S}_T$. One could denote the distance of $\mathbf{s}_t$ as the distance to the k_{th} neighbor or to the average distance of all k nearest neighbors with values over a threshold value denoting an anomalous value. This threshold value can be either user defined or estimated from the nominal data, i.e., it could be found for example by assigning a value close to the average distance of every training sample from its neighbors (for example add the standard deviation of distances to account for variability).

5.2.3.3 k-LPE

Localized p-value Estimation (k-LPE) is a recently proposed method [110] for anomaly detection based on k-Nearest Neighbour Graphs (k-NNG). Each point $\mathbf{s}_i$ on the graph (the training segments in our case), are assigned a score value $R_S(\mathbf{s}_i) \in [0, 1]$ based on the distances (Section 5.2.3.1) to their k-nearest neighbors. Assuming the k nearest neighbors of $\mathbf{s}_i$ are the ordered list of point $kNN(\mathbf{s}_i) = \{\mathbf{s}_{i,1}, \mathbf{s}_{i,2}, \dots, \mathbf{s}_{i,k}\}$, the score for each point is typically defined as:

$$R_S(\mathbf{s}_i) = d(\mathbf{s}_i, \mathbf{s}_{i,k}) \tag{5.5}$$

i.e., the distance to the k-th nearest neighbor.

Given a new test point, an anomaly score $p_K(\mathbf{s}_t)$ is estimated based on the following formula:

$$p_K(\mathbf{s}_t) = \frac{1}{n} \sum_{i=1} n \mathbb{I}_{\{R_S(\mathbf{s}_t) \leq R_S(\mathbf{s}_i)\}} \tag{5.6}$$

where $\mathbb{I}_{\{.\}}$ is the indicator function, i.e., it returns 1 if the condition is valid. Equation (5.6) basically indicates how many existing points on the kNNG of the training data have worse score than the testing point and therefore high values of p_K indicate a non anomalous point whereas low values (near 0) indicate potentially anomalous points. Typically a pre-defined significance level α (in most cases in our experiments $\alpha = 0.05$) controls the anomaly detection – a point is considered anomalous if $p_K(\mathbf{s}_t) \leq \alpha$. The significance value α controls the percentage of false alarm rates; when scores below level α are declared as anomalous the false alarm error is smaller than α asymptotically since k-LPE distributes anomaly scores uniformly in $[0, 1]$ (see Zhao et al. [110] for a proof). This value is a natural way of defining outliers and is easily understandable by users, in contrast to the threshold values in k-NN approaches (Section 5.2.3.2) where it is not clear in most cases how the threshold value should be assigned.

5.2.3.4 One Class SVM

One Class Support Vector Machines (SVM) proposed by Schölkopf et al. [87] is an extension of SVMs for classification problems; everything in the input dataset is considered to be nominal and therefore an "optimal" hyperplane is calculated to divide the space into two regions; nominal and anomalous (Figure 5.5). This approach differs from the previous approaches in the sense that the result if binary; nominal and anomalous whereas the other approaches return a score value. This method was not implemented by us; rather we used the implementation found in the *scikit-learn toolkit* [88].

5.2.3.5 k-Selection

In methods based on algorithms in Sections 5.2.3.2 and 5.2.3.3, k nearest neighbors are to be selected and be associated with the training and testing data. The choice of k plays a significant role on the results; a small value examines a very small subset of the data whereas a very large number could lead to over-training. We follow an approach similar to Hsao et al. [29], where we

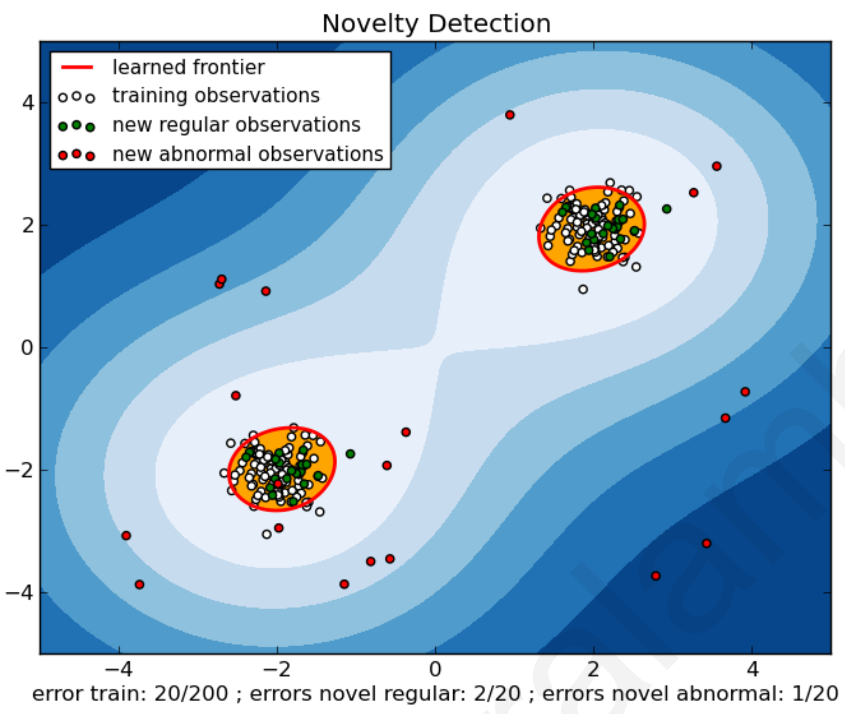


Figure 5.5: Decision boundaries (in red) as estimated by One Class SVM on random data. New test samples that fall into these regions are consider inliers (green dots) and all others (red dots) outliers. Image taken from `http://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html`

iterate over k and construct k-NN graphs for each k; that is, an edge is created for all k-nearest neighbours of each point. We stop this iteration either up to the moment the graph is completely connected or a maximum value is reached (this happens in clustered data). Typically, in a multi-criteria environment we find different k_i, $i \in [1, l]$ for each difference measure m_i; that is edges are created if a sample is in the nearest neighbors in the examined measure. Assuming we look at all the difference measures as one we set k as the average of all k_i (actually the nearest upper bound integer value):

$$k = \lceil \frac{1}{l} \sum_{i=1}^{l} k_i \rceil \tag{5.7}$$

5.2.3.6 Scalarization

The main issue with the all the above mentioned approaches is that it is assumed that all difference measures are of similar importance; i.e., speed is of equal importance as the velocity difference of neighboring agents. In most of the cases, this is not true and therefore weights need to be assigned for each measure. These weights are then used to *scalarize* the segments $\mathbf{s}_i$ into a single scalar representation s_i using convex combinations[2] of the measures:

$$s_i = \sum_{k=1}^{l} w_k . {m_k}^{(i)} \tag{5.8}$$

where $\sum_{k=1}^{l} w_k = 1$ and $w_i \geq 0, \forall i \in [1, l]$. These weights are either manually assigned, or alternatively a grid search on possible weights is performed to select the weights that help in predicting anomalies better. This last approach is performed using k-folds cross validation on the training data; the weights that predict correctly more number of inliers (since we do not know the outliers) are selected. Selecting these weights is very expensive since performance decreases exponentially with the number of difference measures. Having l difference measures and if n different weights are considered per measure, a total of n^l different weight should be considered for n-fold cross validation (which is a of $O(n^l)$ complexity). That would mean for example that for $l = 4$ and $n = 10$, a total of $4^{10} = 2^{20} = 1M$ different weights would need to be examined which is practically prohibiting for real time analysis (Figure 5.6).

Therefore, assuming the proper weights have been found and segment representations are salarized, the distance to the nearest neighbors (Equation (5.3)) reduces to just a single absolute difference:

[2]A convex combination is a linear combination of points in N-dimensional space where all weights are non-negative and sum up to 1

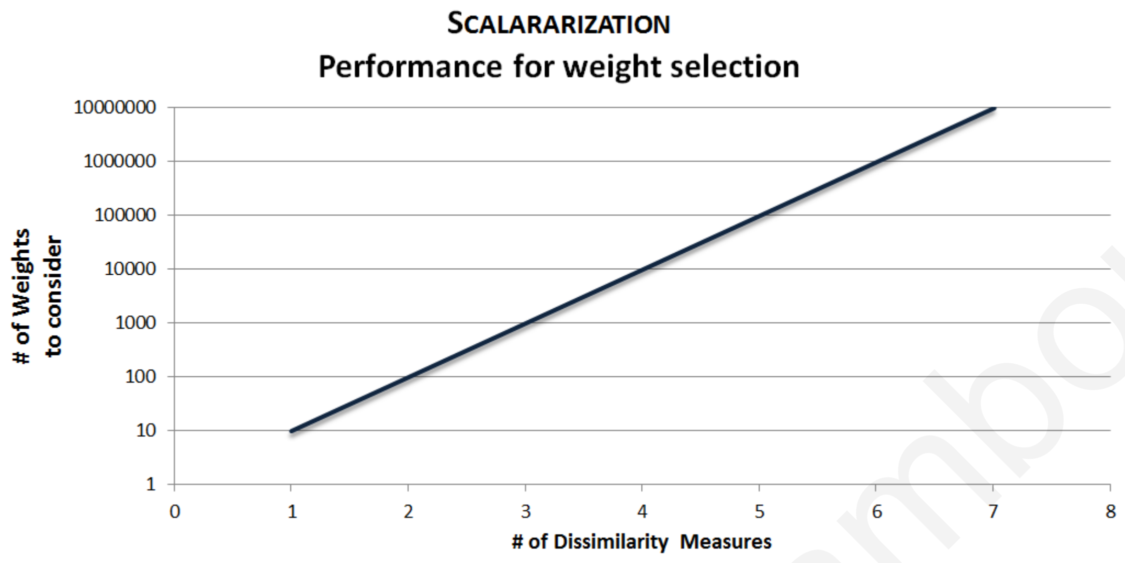


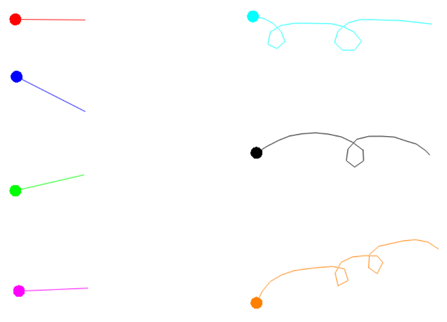
Figure 5.6: Performance of the scalarization drops exponentially with the addition of more difference measures when a grid search approach for the best weights is followed.

$$d(\mathbf{s}_i, \mathbf{s}_j) \equiv d(s_i, s_j) \equiv d(i, j) = |s_i - s_j| \tag{5.9}$$

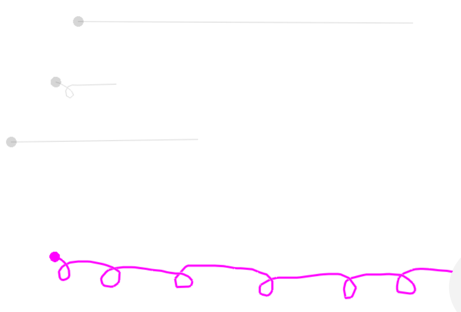
5.3 Multiple-Criteria Crowd Analysis

Often, multiple dissimilarity measures may be needed to capture a wider range of atypical behaviours. However, most anomaly detection approaches do not naturally handle multi criteria. A typical solution to this is to take a linear combination of multiple dissimilarity measures (e.g., average two metrics) as was discussed in Section 5.2.3.6. One of the main issues with this approach is finding the proper weights for each measure which is very expensive especially for more than two measures.

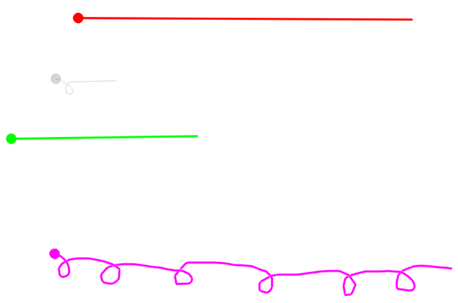
Scalarization failing example However, there are clear outliers which no linear combination of metrics can detect. Consider, for example, the trajectories in Figure 5.7. The trajectories in Figure 5.7a are the set of nominally acceptable trajectories whereas other images show which of the curves in a testing set (of only non nominal trajectories) are identified as outliers (the colored



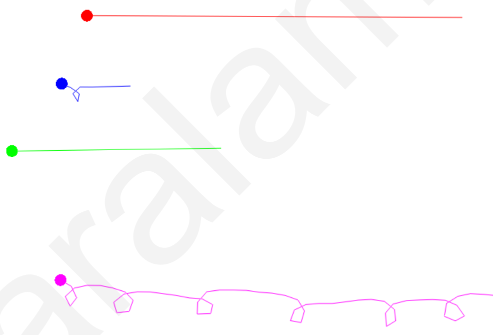
(a) Training



(b) Test - KLPE (Curvature)



(c) Test - KLPE (Displacement



(d) Test - PDA

Figure 5.7: **Hand-drawn Example** In this example, hand drawn data is used for testing and training. (a) The training data has short straight paths and long curvy paths. Both the curvature metric (b) and the length metric (c) fail to detect the short curved path as an outlier as it is no shorter nor more curved than any training data examples. (d) Only PDA correctly detects the short curly path as an outlier because no other short path is curly.

lines). The short, curved blue path in the testing data is no shorter nor more curved than the example paths in the training data and should be identified as an outlier. k-LPE with two different difference measures (curvature and displacement) fails to pinpoint that trajectory as an outlier (Figures 5.7b and 5.7c respectively), therefore *any linear combination* of the two metrics will fail to capture this path as an outlier (by using k-LPE). However, there are no other training paths that are simultaneously short and curved, so we know it is an outlier.

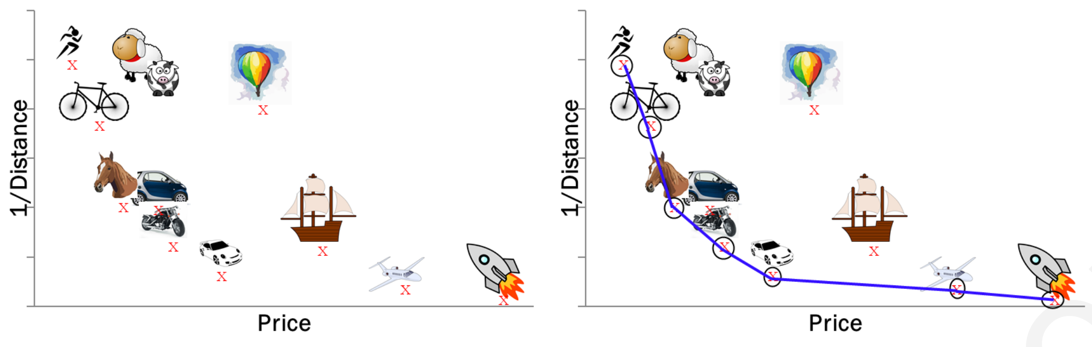


Figure 5.8: (left) Deciding on which transportation to use under two criteria (speed and price) gives a set of two dimensional items. (right) The set of all Pareto optimal points is called the Pareto Front of the items (the circled points).

5.3.1 Pareto Depth Analysis

5.3.1.1 Pareto Optimality

Pareto Optimality is the typical approach for defining optimality in a problem with multiple often conflicting criteria. An item is considered Pareto optimal if there is no other item that is better or equal in all the defined criteria. Take for example the problem of deciding out of a set of vehicles, which is the best one to use based on price and speed (see Figure 5.8). Both of these metrics are often very conflicting since in some cases increasing the performance of a vehicle changes its price and vice versa. Going on foot is the optimal solution as far as price is concerned even though it is the slowest approach, whereas a rocket is the fastest whilst being the most expensive at the same time. There is no other option where it is both cheaper and faster than going on foot or traveling with a rocket and therefore these items are considered Pareto optimal. Travelling with a balloon on the other hand is not Pareto optimal since there is at least one other item that is both cheaper and faster; the car and horse for example. The set of all Pareto optimal items is called the *Pareto Front* (also known as *Pareto Frontier* or *Pareto Set*) of the items (circled items in the right of Figure 5.8). The *Pareto Depth Analysis (PDA)* [29] method exploits the concept of Pareto Optimality as follows.

5.3.1.2 Dyads

First, relationships between all the training segments are found using *dyads* which encode the relationship between two segments over all the criteria – i.e., it is a vector representation of differences for each criterion. More formally, a dyad $\mathbf{D}_{i,j} \in \mathbb{R}^l$ between segments $\mathbf{s}_i$ and $\mathbf{s}_j$ is defined as:

$$\mathbf{D}_{i,j} = [d_1(i,j), d_2(i,j), \ldots, d_l(i,j)]^T \tag{5.10}$$

where l is the number of difference criteria and $d_k(i,j)$, $k \in [1,l]$ indicates the difference between segments i and j for criteria k (Section 5.2.3.1). In the proposed framework, $d_k(i,j)$ is typically absolute differences between the segments for criteria k.

The set of all possible dyads between segments are calculated and stored; if there are N samples in a dataset, a total of $\binom{N}{2}$ dyads are calculated – dyads between the sample and itself are not calculated. The set of all dyads $\mathbb{D}$ encodes the differences between each possible pair of training segments:

$$\mathbb{D} = \{\mathbf{D_{i,j}} : i \neq j \text{ and } i,j \in [1,N]\} \tag{5.11}$$

For example, in Figure 5.9 the set $\mathbb{D}$ of all dyads using two difference measures based on average angular acceleration and average speed ($d_1(i,j) = \Delta|avg(ang.acc)|$ and $d_2(i,j) = \Delta|avg(speed)|$ respectively) are plotted for a real-world dataset (Zara01).

5.3.1.3 Pareto Fronts

Pareto Fronts Construction Pareto Fronts are the set of dyads that are Pareto-optimal; i.e., no other dyads are more optimal than the dyads of the front. The Pareto fronts are computed using

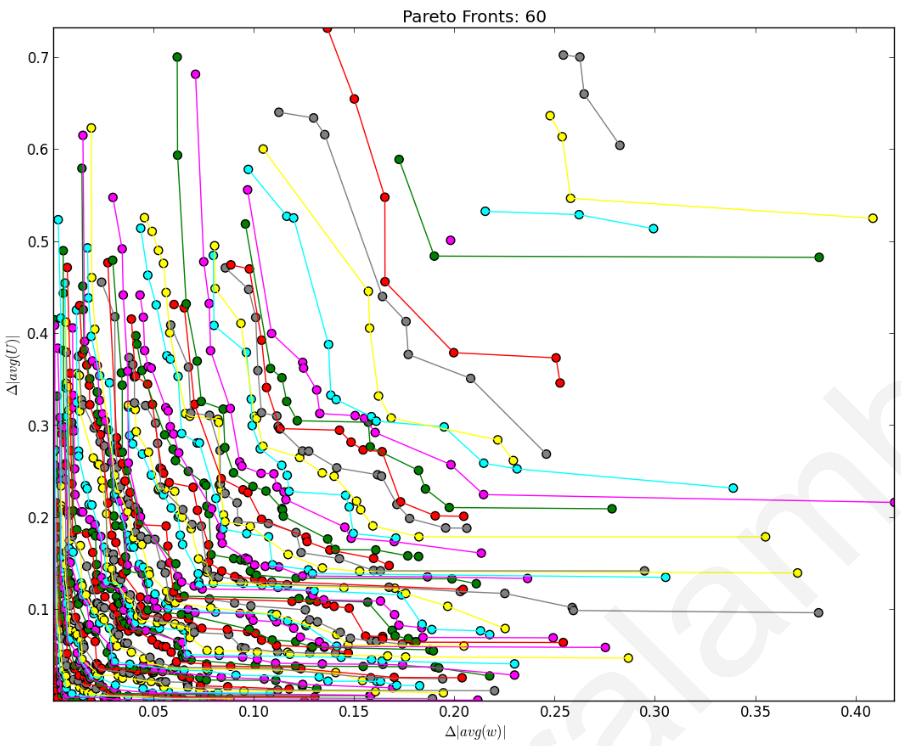


Figure 5.9: Pareto Fronts for the *Zara01* dataset for average speed and average angular speed. Test data that generate dyads that fall in shallow fronts (bottom left) are considered inliers, whereas outliers are concentrated on the deeper fronts (top right).

an iterative approach (Figure 5.10) where the first Pareto front of the dyads is calculated, then the dyads that lie on it are removed from $\mathbb{D}$ and the algorithm is applied to get the next Pareto-optimal front. For each one of these fronts, a rank is given based on the order they are found; i.e., the first Pareto front has rank 1, the second 2 and so on. This process continues until every dyad of $\mathbb{D}$ lies on a Pareto-Front. The lines in Figure 5.9 indicate the estimated Pareto Fronts for the aforementioned example of *Zara01*, with the fronts nearer to the origin being the shallower ones. The list of Pareto-fronts is then used to determine if a new data sample is an inlier or not.

Testing for abnormalities Finally for each new segment to be evaluated, a number of k_i, $i \in [1, l]$ closest matching training samples for each of the criteria is selected and $k = \sum_{j=1}^{l} k_j$ new dyads are generated. Dyads between the test sample and the k neighboring samples are calculated and the rank of the closest Pareto Front for every one of these dyads is found. These

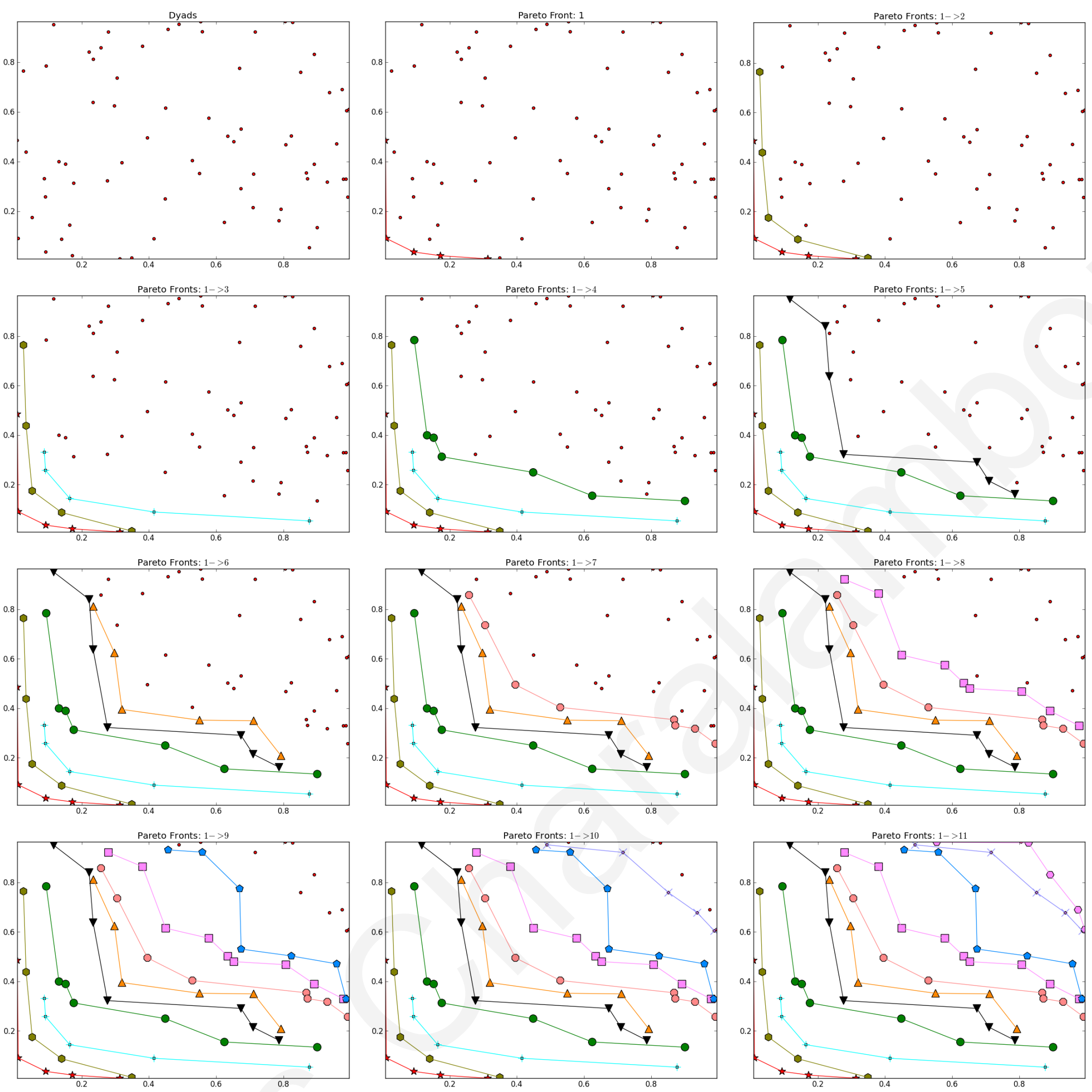


Figure 5.10: **Pareto Fronts Construction** 11 Pareto fronts are constructed one at a time for a random dataset of $602D$ points; that is the first Pareto front is found and the dyads lying on it are removed to calculate the remaining fronts recursively.

fronts indicate the depth of each dyad. By averaging these depths, an anomaly score is found which indicates how anomalous the new sample is. High values indicate deep fronts and therefore possible outliers, whereas low values (and therefore low fronts) indicate inliers.

5.3.1.4 Randomized PDA

In general, given N dyads there will be approximately $\binom{N}{2}$ dyads to construct Pareto-fronts over. The Pareto-fronts construction process is time consuming and the typical approach is if the difference measures are known in advance, then the fronts are pre-calculated and stored for later reuse. Since the proposed framework allows for difference measures (both in number and type) to be used; it is infeasible to pre-calcaulte all possible combinations of dyads and their fronts to be used on an interactive system. Instead, to alleviate this problem, we propose to use a Randomized PDA analysis, in which we construct Pareto fronts over a randomly chosen subset of the dyads. The sampling process is performed uniformly during the construction of the dyads and not the training samples to ensure that data from all the samples are used to construct the Pareto Fronts. We found that this approach works well in practice, and provide an empirical analysis of it in Section 5.4.2.

5.4 Results

We tested our framework across a variety of datasets described below. These datasets consist of both simulated data obtained from a variety of crowd simulations techniques, as well as data captured from real humans.

Grouping dataset: 24 people walk in a group (Figure 5.11b). The pedestrian trajectories were manually tracked and ortho-projected [51].

Zara01 & Zara02 datasets: A sparse crowd of people interacting at a commercial street Figure 5.11c. The first dataset consists of 147 manually tracked human trajectories and the second of 203 trajectories [51].

Crossing dataset: Two groups of 100 agents each cross paths while moving from opposite directions of an obstacle-free environment (Figure 5.12). The agents' trajectories were simulated using ORCA [104].

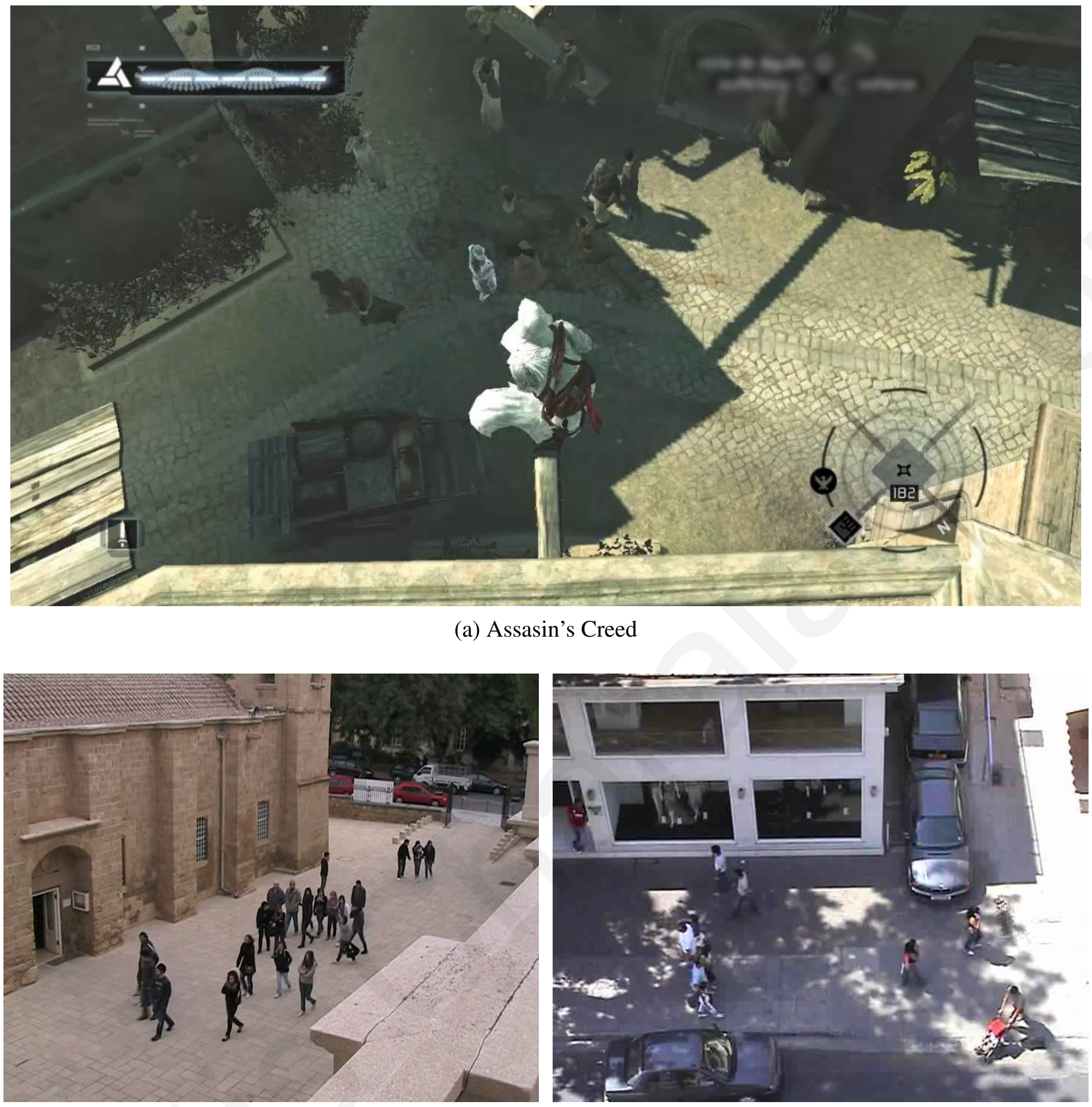
Assassin's Creed dataset: 35 non-playing characters (NPCs) were manually tracked from the popular game Assassin's Creed [100] (Figure 5.11a).

5.4.1 Crowd Analysis

We now highlight some resulting crowds analysis experiments performed using our framework.

Bidirectional Flow. In this experiment, we use the *Crossing* dataset both for training and testing purposes. As the ORCA simulated agents strive for locally efficient motions, they anticipate rather late and, hence, congestion evolves in the center of the environment leading to atypical behaviours. Using the average displacement from the start as a measure, our approach can detect visually apparent outliers, such as agents that are caught inside a stream moving in the opposite direction (see Figure 5.12). Furthermore, by focusing on segments rather than entire trajectories, we can capture the bottleneck at the center of the environment as well as other time-consuming situations.

Synthetic Group Agent. We introduced a synthetic agent in the *Grouping* dataset and tested this new dataset against the original one. The agent was simulated using a VO-based technique [15] and its parameters were tuned so that it can exhibit aggressive behaviour and become eager to reach to its goal. We used a combination of two criteria, one that focuses on the individual behaviour of the entities (speed) and one on the social (differences between an entity's speed and the average speed of its neighbors). As can be seen in Figure 5.13b, dyads associated with the synthetic agent concentrate around the deeper Pareto front, allowing us to automatically detect the presence of



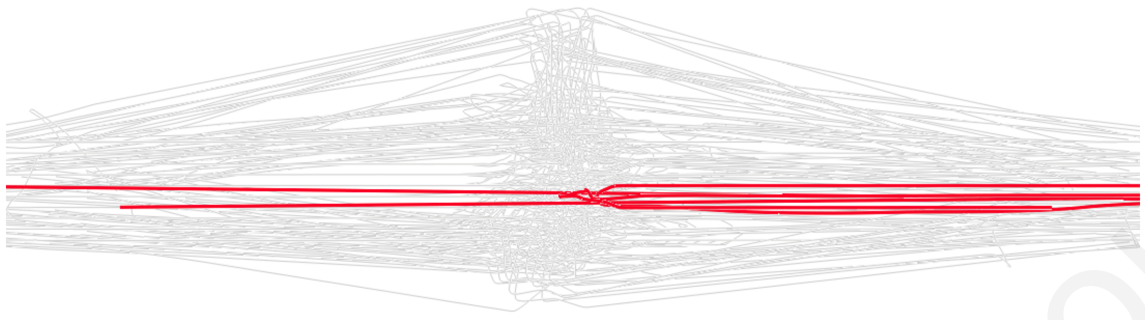
(a) Assasin's Creed

(b) Grouping

(c) Zara

Figure 5.11: **Pedestrian Datasets** Crowd datasets gathered on real and synthetic pedestrians. (a) In the *Assasin's Creed* dataset, 35 autonomous virtual characters are tracked (b) In the *Grouping* dataset, a group of 20 people walking down a street together. (c) In the *Zara* datasets hundreds of individuals walk past a tracking camera on a street corner.

the agent. Note that such an agent cannot be easily detected through visual inspection due to the diversity of behaviours that the other flock members exhibit. We refer the reader to Figure 5.13a and the companion video for simulation results.


(a) Trajectories

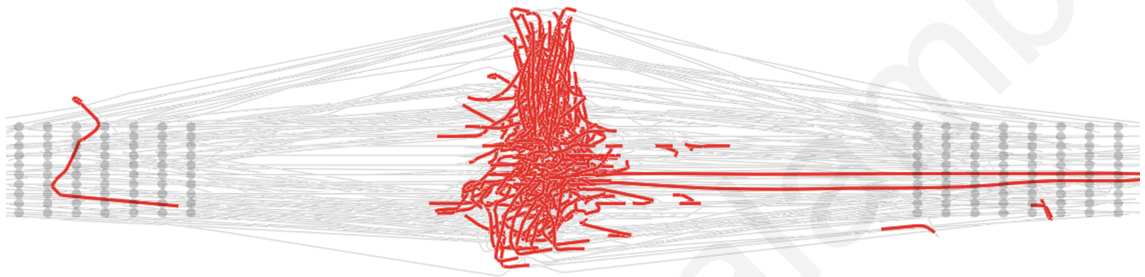
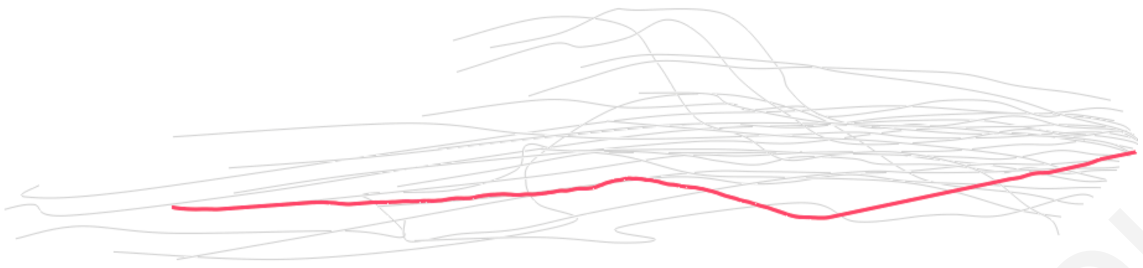
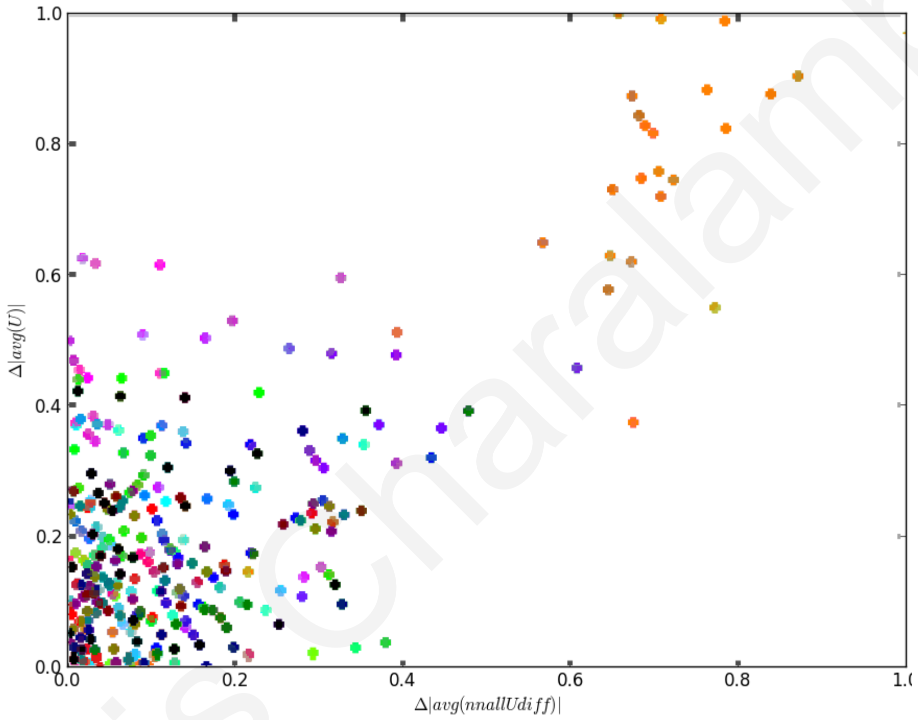

(b) Segments

Figure 5.12: **Crossing Scenario** 200 simulated agents walk past each other. (a) Anomalous trajectories are highlighted. In this case, two agents are found which get pushed backwards a long distance (b) Anomalous segments colored via a heatmap. Highly anomalous segments (dark red) are found primarily in the central congestion, i.e., in some agents who fail to find their goals quickly.

Interactions in a virtual city. In this experiment, we analyzed the quality of the manually tracked NPCs from the *Assassin's Creed* game (see Figure 5.14). The NPCs global motions are governed by waypoints along the medial axis of the environment, whereas a reactive technique is used to resolve local collisions between them. As such, the characters typically have to be very close to react to each other, which results in atypical behaviours, such as backward motions, oscillatory movements, etc.. We evaluated the quality of the path segments of the NPCs under two criteria: maximum curvature and average speed. Using k-LPE on each criterion independently, a number of different anomalous segments were detected. However, only the combination of the two criteria using our proposed randomized PDA implementation allowed us to characterize a wider corpus of segments as potentially erroneous.



(a) Trajectories



(b) Dyads

Figure 5.13: **Synthetic Agent** A synthetic agent was inserted into data of real pedestrians from the *grouping* scenario. (a) While the path looks visually similar, our approach was able to detect the synthetic agent as an outlier using a combination metrics of an speed and the difference between the agent's speed and it's neighbor's speed. (b) The dyads for the above example. The dyads for the outlying agent are concentrated on the top right (i.e., have a large Pareto depth).

5.4.2 Real-time Crowd Analysis

Randomized PDA As discussed in Section 5.3.1.4, using a randomized subset of the dyads in PDA to generate the Pareto fronts allows for the return of accurate crowd analysis results in real-time. Figure 5.15 demonstrates that the RMSE of the predicted anomaly scores compared to the

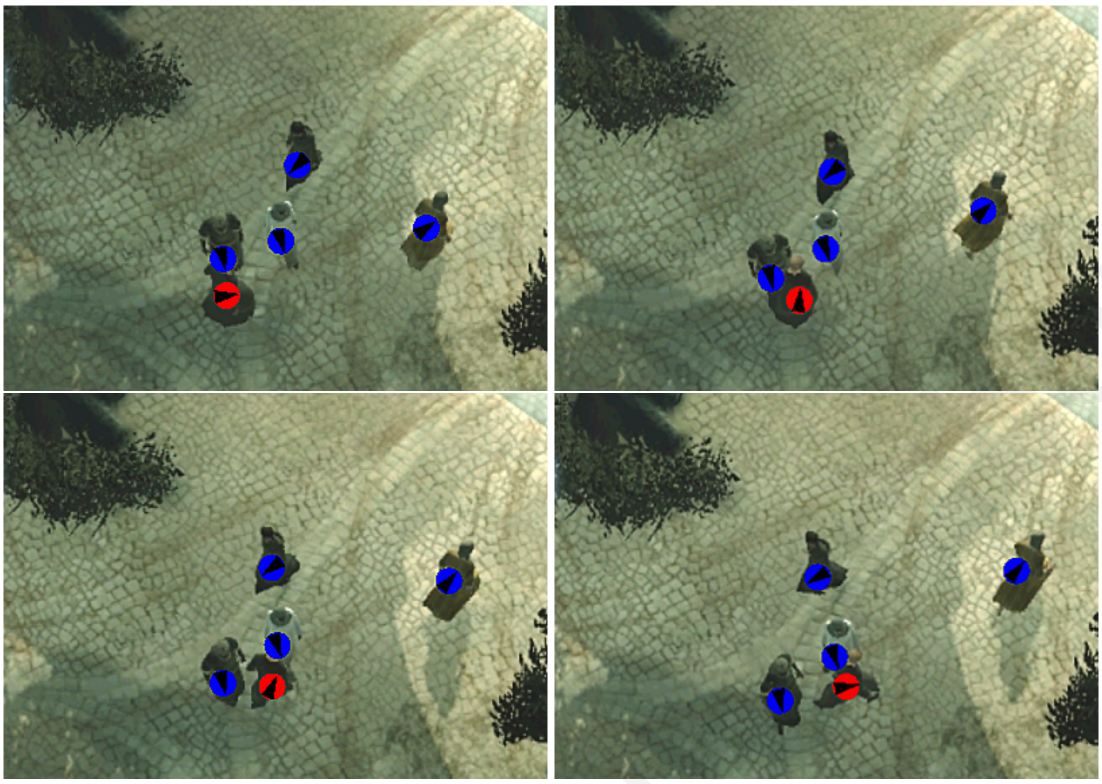
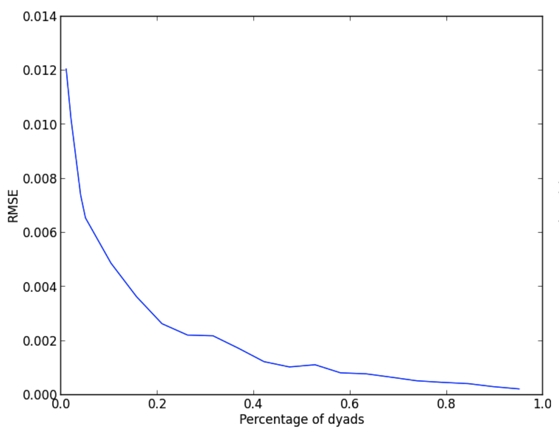


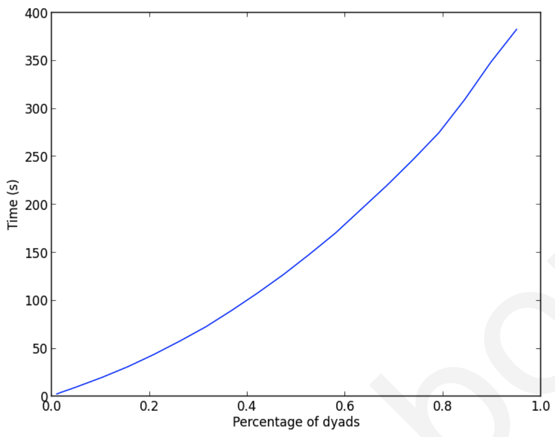
Figure 5.14: **Assassin's Creed** Example outlier detected in a simulated agent in the game Assassin's Creed (red circle). The outlying agent can be seen to spin around quickly as it tries to avoid the upcoming collision too late. This outlier was found using PDA evaluating path segments interns of maximum curvature and average speed.

scores retrieved by using all the possible dyads is low even for as few as 5% of the original dyads, converging to zero rapidly.

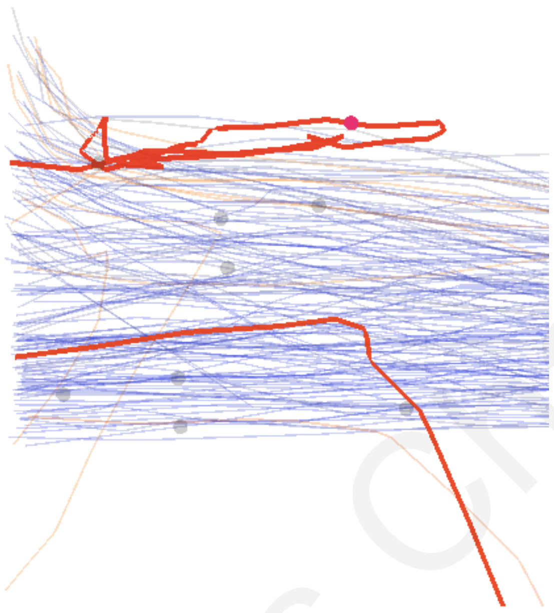
Given the interactive nature of our analysis tool, quick responsiveness to a user's requests is a very important feature. Overall, our system has real time performance for very large datasets. A summary of our runtime performance on various datasets is given in Table 5.3. Time refers to the total processing time to generate the new fronts from the selected dyads subset plus the total time to characterize the test data trajectories. The processing time is linear in both the dyads subset size and the number of metrics selected. The numbers in the first and second columns are the number of trajectories in the corresponding datasets. All results were obtained on an Intel i7 CPU @ 3.50 GHz processor.



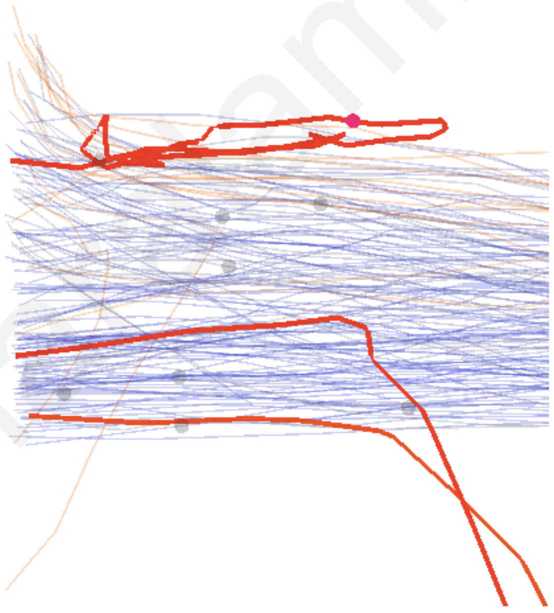
(a) Error Rate



(b) Runtime



(c) Full PDA (5.5 min)



(d) Randomized PDA (15 s)

Figure 5.15: **Randomized PDA** Our randomized PDA approach returns similar results to full PDA with much less runtime. (a) The RMSE on the *Zara* dataset (using 4 anomaly measures) quickly approaches 0 with a small percentage of dyads. (b) Runtime of the randomized PDA is vastly improved with less dyads (c-d) Heatmap coloring of outliers (red is highly anomalous) (c) Computing outliers with the full PDA algorithm takes several minutes. (d) Our randomized approach using only 10% of the dyads produces very similar anomaly scores in just a few seconds.

5.5 Discussion and Future Work

We have introduced a novel crowd analysis and evaluation method. By expressing the problem as outlier detection, simulated trajectories can be measured against nominal data to identify

			Total Time (s)				
Training	Testing	Metrics	20%	40%	60%	80%	100%
Zara02 (203)	Zara01 (147)	2	5.2	11.3	18.5	26.4	35.1
Arxiepiskopi (24)	GroupM (261)	2	0.9	1.3	1.5	1.8	2.1
Arxiepiskopi (24)	GroupM (261)	4	2.8	4.5	6.2	7.7	9.7
Crossing Sim (200)	Perpendicular Sim (180)	2	8.9	17.6	26.0	37.7	47.3
Zara01 (147)	Assasin's Creed (35)	10	31.2	97.8	200.3	332.1	486

Table 5.3: Timing results for various datasets. By using a small percentage of the possible input dyads we gain in performance without compromising quality. Increasing the percentage of the dyads has a linear effect on performance cost.

possibly problematic parts. Using Pareto Depth Analysis, a recently introduced machine learning technique, we have been able to combine multiple criteria in each evaluation, broadening the range of effects we can detect. We have also shown how to use randomization in order to accelerate the training of PDA and make the method usable even within an interactive environment.

Our approach has some limitations that we would like to address in the future. In our current implementation, we use a randomized selection of training dyads allowing us to assess the quality of a simulation at interactive rates. However, the analysis of very large crowds, consisting of thousands of agents (see, e.g., [64]) can become very slow.

Another limitation, inherent to any data-driven approach, is that the reference training data may be noisy and also not cover the entire spectrum of behaviours that the testing entities can exhibit. This can result in false-positive outliers, that is, behaviours missing from the training set may be considered as anomalous. In the future, we would like to investigate ways of accounting for such incomplete and noisy nature of the training data. One option is to incorporate fuzziness/uncertainty in our dissimilarity measures using an approach similar to the one proposed in [22]. In addition, we can enlarge the input training dataset by blending between samples following the spatiotemporal approach of Ju et al. [34].

Selecting the metrics manually could be a burden to the user. Instead of letting the user decide on them, we are considering an automatic approach where the appropriate metrics are selected based on the training data. Additionally, generating and searching in high dimensional Pareto fronts is a very time consuming task, therefore a good alternative would be to find ways of projecting the data into a lower dimensional manifold aiming to keep the distance relationships between the segments as much as possible. Additionally, a GPU search based approach to find the Pareto fronts is considered.

Finally, we believe that the concept of Pareto optimality can be exploited in other domains besides crowd simulation such as in character animation. As it stands, our proposed framework can help researchers from the pedestrian and crowd simulation community to author better simulation algorithms, as it provides an automatic way of detecting possible erroneous behaviours in simulations. The entertainment games industry can also exploit our technique to further improve the planning AI of the NPCs.

Chapter 6

Behaviour Annotation

One important issue in data driven crowd simulation is the selection of appropriate data, from a potentially large databases of input trajectories, that will convey the required character and behaviour of each simulated agent. In this work, an approach for identifying and extracting trajectories with similar behaviour patterns is presented. The proposed method samples annotated trajectories to extract examples over the input using both local and global statistics. These examples are then used to train a classifier that annotates trajectory segments into six different behaviours such as group walking and talking interactions between people. Early results show high accuracy in classification. This methodology can be used for a multitude of applications, such as evaluation of crowd simulators, data selection, surveillance and tracking of pedestrians.

6.1 Introduction

Crowds of pedestrians are an important feature of both virtual and real worlds. Various scientific fields have been concerned with studying crowds. Some, such as Computer Vision, study them in attempt to understand the intents of real people captured by sensors. Others, such as Computer Graphics, in attempt to convey the desired intend in the simulation of virtual people.

Being able to extract higher level understanding of pedestrians from their trajectories can have a wide array of applications such as crowd synthesis, crowd evaluation and outlier detection. Having a method for automatic labelling of trajectories, or parts of them, with tags such as walking, standing still, talking to another person or as being part of a group, real and virtual crowds can be better understood. Example applications are in the area of data-driven crowd synthesis, or for evaluation of any type of simulation as compared to real data. As far as crowd synthesis is concerned, trajectories from large databases could be extracted so that they exhibit a particular behaviour pattern and then used to train a data driven crowd simulator resulting in more believable crowds. Typical methods to evaluate a virtual crowd consist of quantitative approaches such as gathering statistics from the simulated trajectories and getting scores for various tasks such as time to reaching goals, minimum time to collision, average speeds etc. or qualitative approaches that employ typically user evaluations. By comparing higher level characteristics of the simulation and to real world data, we believe that better evaluation algorithms can be developed.

A set of features that help in high level classification of trajectories is proposed based on local and global characteristics of trajectories. This selection of the two type of features is capable of capturing behaviours that only one the two set of features would fail to do so due to both local interactions between pedestrians and global goals of individuals.

6.2 Methodology

In the proposed classification scheme, users are requested to tag people in a video into six categories (Table 6.1):

- *Walk*: people walking by themselves,
- *Wander*: relatively slow walkers wandering around,

Table 6.1: **Different classes of pedestrians** The proposed system was used to classify pedestrians into 6 distinct categories, ranging from walking behaviours (walk, wander and walk fast) to simple individual (wait) to the more social/group ones (meet-and-talk to moving in groups).

Classes		
Walking	Assertive	Social
Walk Wander Walk Fast	Wait	Group (walking) Meet and talk

- *Walk Fast*: people walking in a hurry,
- *Wait*: people standing in place or walking up and down waiting for someone/something
- *Meet and Talk*:people forming groups to talk and
- *Group*: people moving around in group formations

Even though some of the classes are very similar with subtle differences (such as the walk, wander and walk fast classes), a person can differentiate between them. We note here that a single class per person is assigned in this test case, just for simplicity reasons. If the pedestrian trajectories are assigned multiple classes at different or even the same time intervals, classification would surely benefit but at a higher cost in the manual tagging phase.

After the tagging phase, each trajectory is split into smaller fixed time sub-trajectories which are overlapping (Figure 6.1). Different sampling schemes were considered here with varying segment sizes; the results presented in this work consist of 4s trajectories spaced at 1s intervals (100 frames per trajectory for a 25fps video). Each segment S_i, is considered to represent the class of the segment it belongs to. For each one of these segments, a feature vector consisting of 12 values were calculated (see Table 6.2). These values are split into two categories: history and segment values. History values are statistics that represent the entire history of the pedestrian up to the end of the segment (from t_1 up to $t_{n+seg.size}$ in Figure 6.1, where $seg.size$ is the trajectory segment size), whereas the segment values are statistics of the local segment only (from t_n

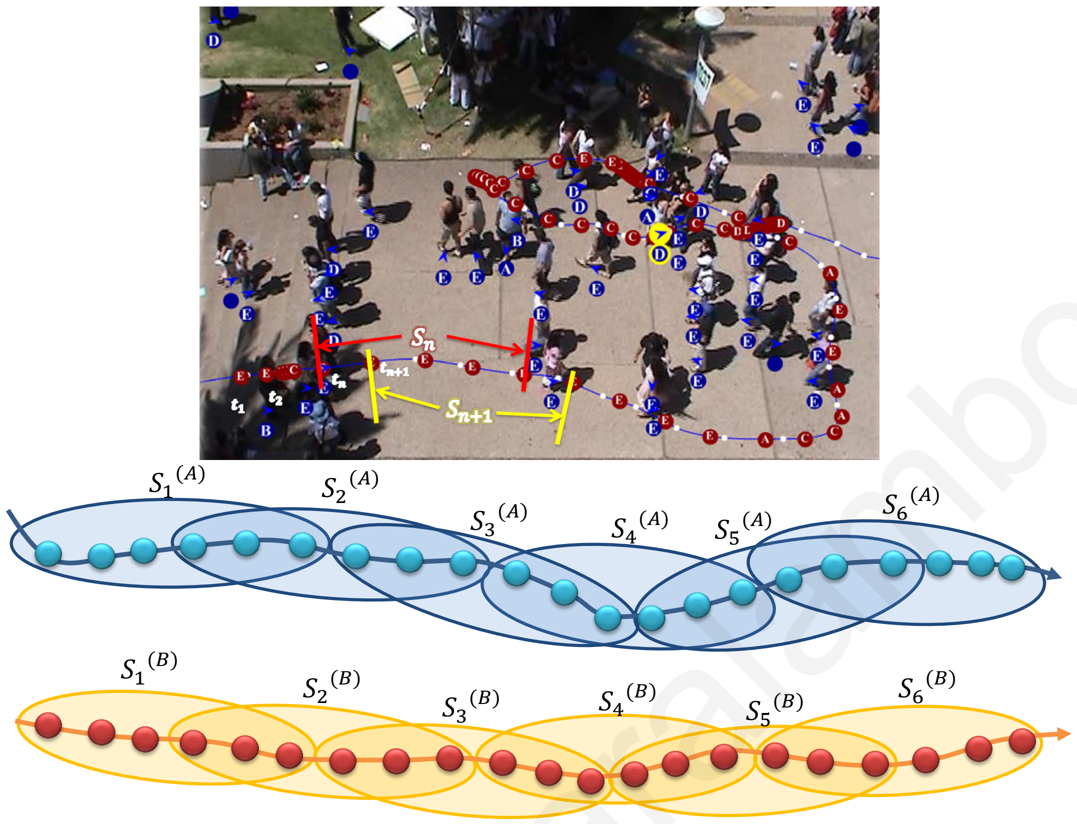


Figure 6.1: **Segments** A trajectory is segmented into smaller overlapping trajectories S_n. Each one of these segments is used as an example of a particular class of behaviour.

up to $t_{n+seg.size}$). We chose this scheme, since we believe that the behaviour of a person is on one side consistent with the history of his behaviour and on the other side on the the local interactions/actions he takes at any given time. By taking into account only one of the two, we have found that the classification accuracy dropped significantly.

This processing of the data gives clusters of examples that are representative (according to the input data) of the previously mentioned classes. A visualization of the input data per class is shown in Figure 6.2; each column represents a feature and is normalized to the maximum value in all the classes. This figure represents the distribution of values for each feature in each class; median values are represented with horizontal lines and the curves represent the distribution of values (typically Gaussian) reflected in both directions. The area between the dotted lines in each

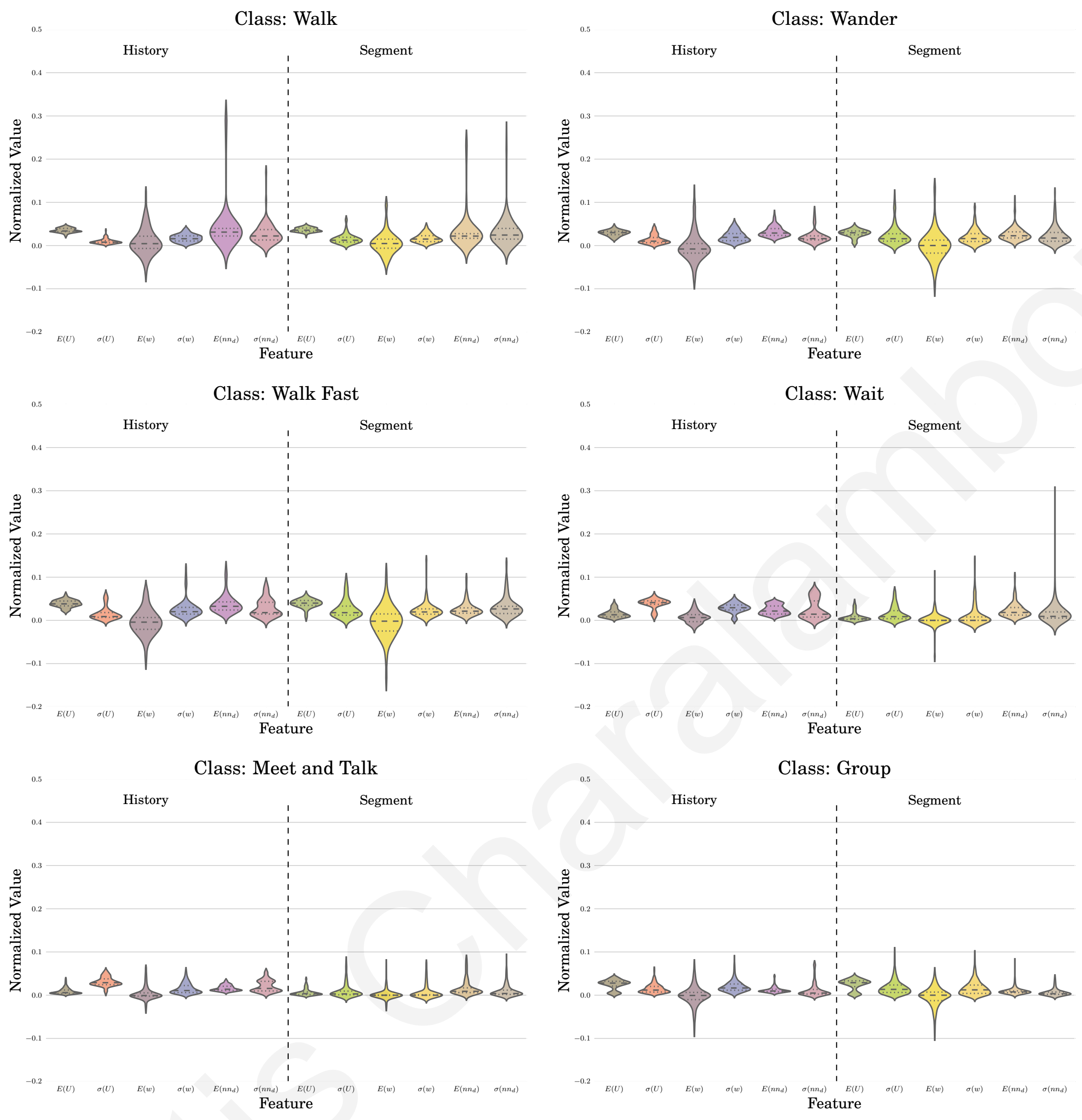


Figure 6.2: **Classes and their example distributions** Distributions of values for the data in the tested dataset for each one of the six classes. Each violin diagram represents a class, and each column of a diagram is a feature.

feature represent 75% of the values. The first row of figures represents single people walking but with slightly different values; the human tagger was able to differentiate between a wanderer and people walking in normal or fast pace. The *group* behaviour is similar to the walking behaviours, but as it can be observed people in groups move slower to other walkers and additionally they have smaller interpersonal distances, both in the history and segment parts.

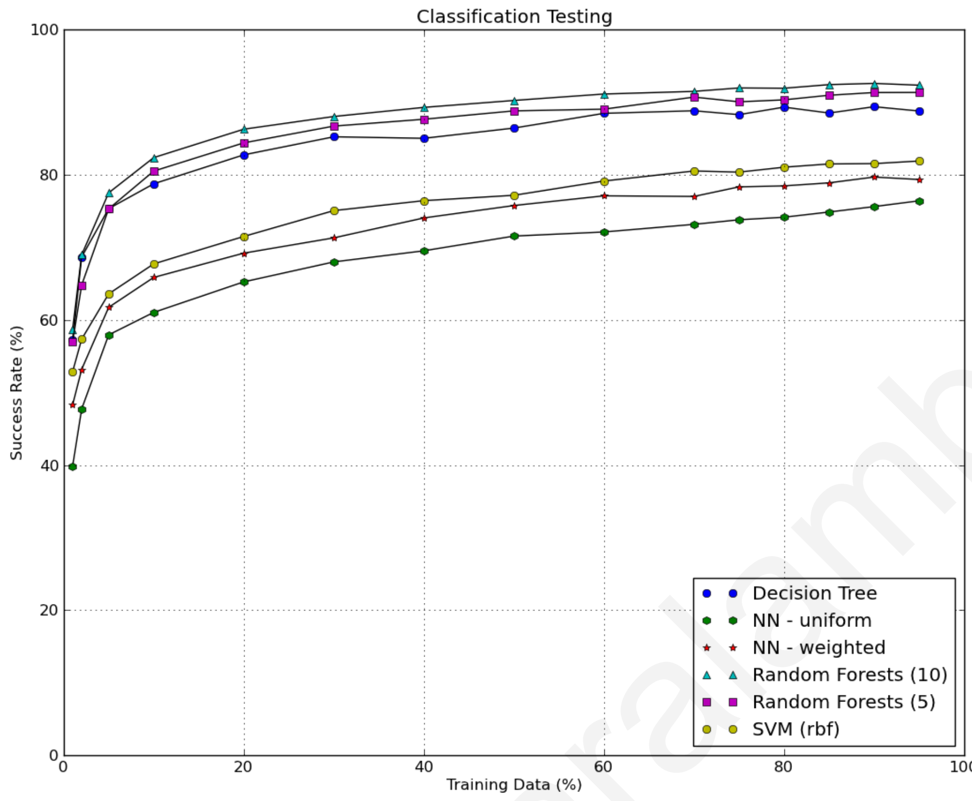


Figure 6.3: **Performance of different classifiers** Different classifiers performance measured using the F1 score for different training data sizes. Tree based algorithms get the best scores, with the Random Forest algorithm outperforming all others.

These examples were then used to train various classifiers, and the best one amongst them was selected as the final classification algorithm (Figure 6.3).

6.3 Experiments

To test and evaluate our methodology, pedestrians from a crowd outside a store were tracked (*Zara2* dataset in Table 6.3) and each trajectory was assigned a single class out of six possible by a user (Table 6.3). These trajectories are then segmented into smaller overlapping example segments of 4 seconds length (a) and for each of these examples the statistics in Table 6.2 were generated (as described in Section 6.2). We run two types of test to evaluate our approach: on one hand we run quantitative experiments to verify the quality of various classifiers on the input

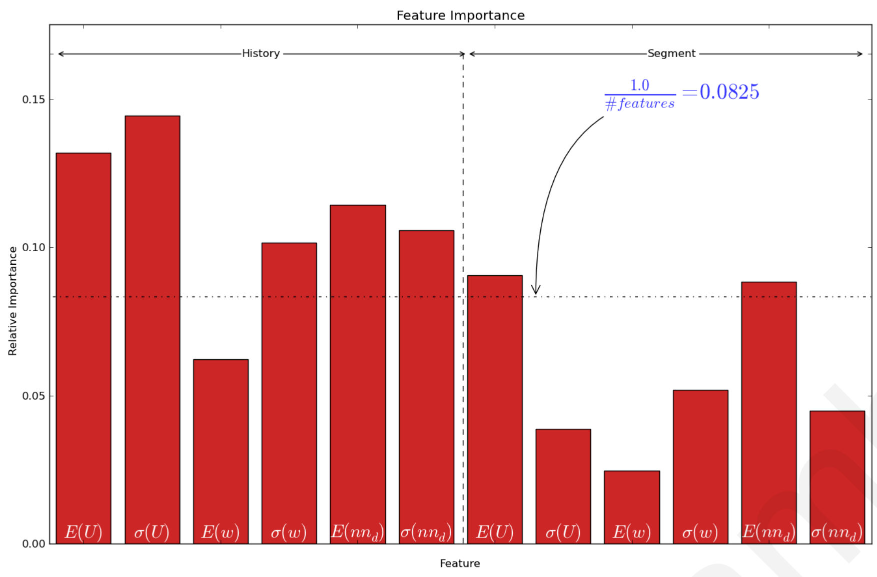


Figure 6.4: **Feature importance** Feature importance in the classification as this is estimated by the Random Forests algorithm. Observe that most of the features contribute to the classification.

data and on the other hand we run qualitative experiments by applying the classifier on untagged sources that exhibit different characteristics.

To test the classifiers, the input examples were split into training and testing data of various sizes and were run several times with different training and testing data. For each of these experiments the F_1 score was calculated – high values near 1 mean better classification accuracy. We have found that the Random Forests classifier [2] with 10 random decision trees was the best choice, with $> 90\%$ prediction accuracy. The Random Forests classifier was also used to find each individual feature's importance (Figure 6.4); most of the selected features contribute to the classification which partially explains the accuracy of the algorithm.

A Random Forest classifier was used to annotate the behaviour of various people in various pre-tracked video sources; these trajectories were tracked using either our own manual crowd tracker or they were taken from various online datasets such as the ones by [48, 51, 75]. Additionally, data from a synthetic dataset generated from Reynold's OpenSteer framework [82] were also used.

Table 6.2: **Statistics used as features** The statistics used in this work consist of 12 distinct scalar values for speed, rotational speed and distance to nearest neighbor. For each segment of a trajectory, average and standard deviation values were found for both the segment and the history of the trajectory.

	Statistic	$E(.)$	$\sigma(.)$	Description
History	$U(t)$	✓	✓	Speed
	$w(t)$	✓	✓	Rotational Speed
	$nn_d(t)$	✓	✓	Distance to nearest neighbor
Segment	$U(t)$	✓	✓	Speed
	$w(t)$	✓	✓	Rotational Speed
	$nn_{dist}(t)$	✓	✓	Distance to nearest neighbor

Table 6.3: **Datasets** Trajectories in the *Zara2* dataset were extracted and examples were generated with each one of them assigned to a single class out of six. The columns from *Zara1* up to *reynolds–pedestrian* indicate the number of examples per class as they were classified by the Random Forest classifier using as training data the *Zara2* dataset.

		Input					# Examples per class				
ID	Name	Zara2	Zara1	Zara3	arxiepiskopi	students	biwi-hotel	biwi-eth	lee-chat5	lee-stroll	reynolds–pedestrian
–	Walk	122	93	89	29	192	89	67	0	0	17
A	Wander	241	158	165	76	766	151	131	40	356	3650
B	Walk Fast	100	64	44	6	134	166	146	0	0	0
C	Wait	261	12	43	19	351	162	183	39	20	5
D	Meet and Talk	608	238	199	38	1312	343	27	45	11	96
E	Group	1733	901	911	292	2681	334	272	0	1	182

The leftmost column in Table 6.3 shows the number of training examples belonging to each of the six categories. All the other columns display the number of segments in various test data as they were clustered by the proposed system. In order to verify the results using these data, the trajectories should be manually classified similarly to the input using users and then the classification accuracy could be compared to the ground truth which in the end; be it a simulator or a video tracking tool is the final judge on quality. As a first step, the class annotations for the different trajectories (subtrajectories) are exported and displayed for fast verification (Figure 6.5).

Datasets *Zara1* and *Zara3* (from [51,53]) are videos from the same area as the training dataset (*Zara2*) and were taken with a few minutes of difference and were used for sanity check – the area is the same and has the same scale as the input. The automatic tagging results were very

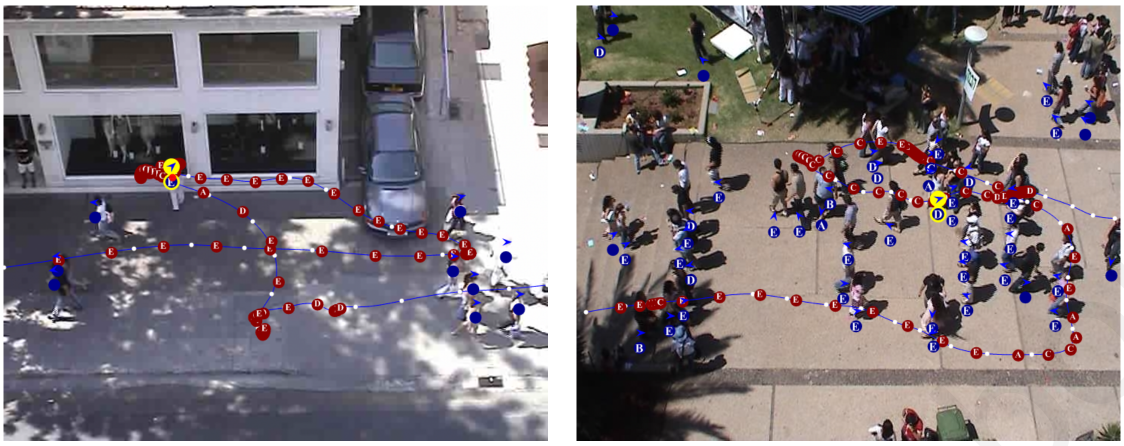


Figure 6.5: **Examples on videos of people.** Trajectories are annotated using the proposed methodology. Here the annotations for a single trajectories are presented with each letter indicating one of the six classes.

promising as the system succeeded in identifying people in these categories. Please consult the accompanying video for the tracking results on some of these videos. As it can be seen in Figure 6.6, the distribution of the segments into three supergroups (Walkers, Wait and Groups) for the three zara dataset are very similar especially in the groups of people, which is a possible indication of typical groups vs single persons distributions in that area.

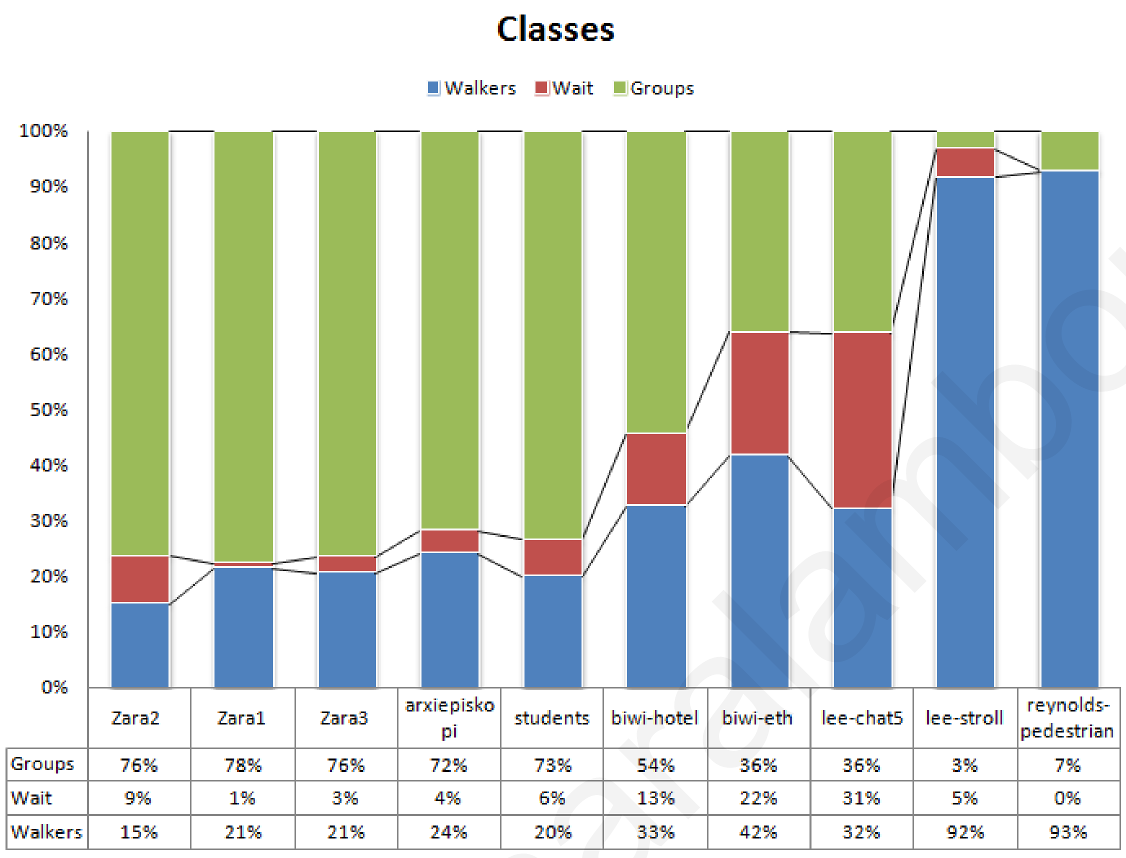
The *arxiepiskopi* dataset consists of flocking behaviour of people in an archaelogical site, and even though the input data didn't have flocking behaviour in large numbers, the proposed method managed to capture that most people in that video were in groups. The *students* dataset on the other hand consisted of a dense crowd of students in a university that exhibited a number of complex behaviour patterns which were concurrently present. These behaviours were also present in the training data but at a lower degree. The accompanying video contains a part of the student video and the tags for the students - as can be noticed in the video even though the system mistags some of the pedestrians, it does so for similar classes such as the Meet and Talk and Group behaviours. This could be due to either the complexity of the test data or due to incorrect ortho-projection of the data which could lead to imprecise measurements. The first issue could be handled with more training data or the selection of different features.

The *biwi* dataset (from [75]) consisted of people walking outside a hotel and the ETH building. These videos contained similar behaviour to the zara datasets but with different distributions; for example the hotel video had a relatively large number of people waiting for the bus an were standing still for large periods.

The *lee-chat5* dataset (from [48]) consisted of 8 people walking, forming small groups and splitting which was successfully captured by the system. *lee-stroll* on the other hand consisted mostly of people wandering around, independently of each other which is quite prevalent in the classification (Figure 6.6). The synthetic dataset (*reynolds-pedestrian*) was generated using the pedestrian simulator in the Opensteer framework and consisted of 50 agents walking around a circle. Since the maximum assigned speed for the agents was relatively low and agents were moving around in a wavy manner, most of the segments were tagged as wanderers. Some of the subtrajectories were identified as temporal groups that lasted a few seconds due to the slow speed of the agents and their temporal proximity (see Figure 6.2).

6.3.1 Evaluating crowd simulations using higher level annotations

Higher level evaluation By having a higher level understanding of crowds through nehaviour annotations, more intuitive comparisons between crowd simulation can be performed. Take for example the Perception-Action Graph (PAG) data-driven crowd simulation framework presented in Chapter 4. The PAG can simulate crowd simulations using as input data from videos of crowds or well behaved simulations. We can compare input and output of the simulator, by tracing both input and the output simulation trajectories. Input data are classified manually by users, segments



	Zara2	Zara1	Zara3	arxiepisko pi	students	biwi-hotel	biwi-eth	lee-chat5	lee-stroll	reynolds-pedestrian
Groups	76%	78%	76%	72%	73%	54%	36%	36%	3%	7%
Wait	9%	1%	3%	4%	6%	13%	22%	31%	5%	0%
Walkers	15%	21%	21%	24%	20%	33%	42%	32%	92%	93%

Figure 6.6: **Classification results** Distribution of the different classes for the datasets in Table 6.3. Classes were grouped together for visualization purposes – *Walkers* consist of classes Walk, Walk Fast and Wander, *Wait* is class Wait and *Group* consists of classes Group and Meet and Talk.

are created and statistics over the segments are found to train the proposed classifier. By segmenting simulations and collecting the same statistics as the training phase, we can annotate them with higher level descriptors to get an overall image of the simulated crowd; that is how many segments belong to walkers or small groups. This approach for evaluating is similar to what a human observer would comment on observing a simulation such as "this simulation has more groups and therefore looks more realistic (or vice versa)".

Evaluating the PAG framework Notice for example the results in Figure 6.7. We generated simulations using the PAG framework using as input the *zara2* dataset. Each simulation was run with different parameters such as the jump thresholds, graph construction threshold, number of nearest neighbors for searches etc., similarly to the experiments performed in Section 4.5. Then, we annotated these simulations using the approach presented in this chapter into the 6 groups presented in Table 6.1.

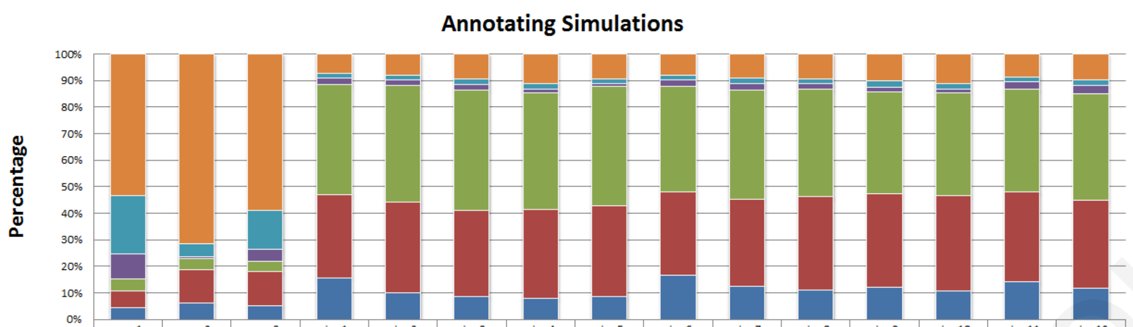
As it can clearly be observed in the graph, the simulation framework failed to capture group behaviour of people such as people walking together. Most of the trajectory segments in the source data belong to people in small groups ($> 70\%$), whereas most of the simulations exhibit group behaviour on only $7 - 11\%$ of the total segments. This is an indication that the framework, should take into account interpersonal distances for example during transitions on the graph. One alternative, is to modify the optimization function (Equation (4.3)) to account for formations – set target to be a group formation for example.

Secondly, there are a lot of fast walkers in almost all the simulations ($> 40\%$), which is a results of the selection mechanism applied in the framework when there are no interactions; actions that drive agents towards goals are selected even though this is not always the case so that agents do not stay in the same place for a long time.

Finally, the PAG framework managed to capture all of the input behaviours to some extent, even though it didn't manage to capture the input behaviour distribution in its entirety. It's worth noting here the stability of the algorithm on capturing similar behaviours with different parameters; most of the simulation have similar behaviour distributions between them. This agrees with the results in Section 4.5 where collision quality of the PAG over different k-nearest neighbours remained similar.

6.4 Discussion

A method to identify and tag classes of people, using local and global trajectory statistics has been presented. This tagging algorithm identifies people with similar behaviours and not people



	zara1	zara2	zara3	sim-1	sim-2	sim-3	sim-4	sim-5	sim-6	sim-7	sim-8	sim-9	sim-10	sim-11	sim-12
🟧 groups	54%	72%	59%	7%	8%	10%	11%	9%	8%	9%	10%	10%	11%	9%	10%
🟦 meet and talk	22%	5%	15%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
🟪 wait	9%	1%	4%	2%	2%	2%	1%	1%	3%	3%	2%	2%	1%	3%	3%
🟩 walk fast	5%	4%	4%	41%	44%	45%	44%	45%	40%	41%	40%	38%	39%	39%	40%
🟥 wander	6%	13%	13%	31%	34%	33%	33%	34%	31%	33%	35%	35%	36%	34%	33%
🟦 walk	4%	6%	5%	15%	10%	8%	8%	8%	17%	12%	11%	12%	10%	14%	12%

Figure 6.7: **Annotating simulations for evaluation** By comparing higher level annotations of real world crowds and simulations we can gain a more intuitive comparison between them compared to statistical approaches that is similar to what a human observer would say.

with similar trajectories as most trajectory similarity algorithms do. This is achieved due to the selection of both local and global statistics. We see a lot of potential applications of the proposed method.

Motion Mining for data driven crowd simulation: typically a designer wishes to develop a crowd simulation having a particular behaviour. By using the proposed scheme, a designer could select some people in source videos and the system would search the entire database of examples to mine similarly behaved people to use in a data driven crowd simulation system.

Evaluation for crowd simulations is a very important issue. Typical crowd evaluation systems take into account various numerical statistics of the simulation and evaluate the performance / quality of the simulator according to these. Other researchers use input from users to characterize their systems. We believe that a system that combines both is most appropriate; this system can tag crowd simulation data with higher level constructs that describe each individual characters behaviour patterns and at the same time define the overall behaviour of the crowd such as a crowd of small groups of 2 or 3 people, fast walkers etc. These constructs are defined by users and are learned by the classification algorithm, which in turn associates a users preconceived knowledge of what is important in defining crowd's behaviour.

Motion Synthesis This method can also help in *Motion Synthesis*; since at simulation time the crowd simulator will be able to decide what animation to play during movement. If for example an agent is in the Wait class, then an Idle animation could be played, or play the appropriate animation during the transition from idle to walking.

Video tracking Another possible use of this method would be in *video tracking*, where future positions of tracked pedestrians could be predicted according to their higher level behaviour description. Relative to this area would be the *surveillance*, such as a system capable of identifying outlier trajectories by finding people with strange transitions from one class to another.

As feature work we are considering on looking into other features also in addition to the ones presented here, similar to the ones in Chapter 5. In Figure 6.4 we observe that some of the features such as $\sigma(U(t))$ and $\mathbf{E}(w(t))$ for the local trajectory segments have less impact and could be replaced with other statistics such as the average number of neighbors around a person or the angle to final destination (which could indicate desire to reach a goal). These and some other statistics are currently on testing phase. Also, the tagging of the training data is purely empirical to the user of the system, since it is left to the user to decide and differentiate between the classes. A more detailed and expensive tagging procedure such as multiple tags per trajectory or multiple user tagging would surely benefit classification. We have found, that even though a single class for the whole trajectory was assigned, the classification accuracy of the proposed scheme is quite high. This is due to the fact that the examples that are generated use both global and local statistics of the trajectory which can capture subtle statistics that can differentiate between the classes.

Chapter 7

Conclusions and Future Work

7.1 Contributions

As part of this thesis, novel data-driven techniques to synthesize, evaluate and analyze agent based virtual crowds have been proposed, implemented and tested. All of the proposed methods employ data from real or simulated crowds; data implicitly define an underlying crowd behaviour that can be both simulated and compared to.

Crowd Synthesis A novel data-driven simulation framework – The Perception-Action Graph [7] has been presented to efficiently simulate virtual crowds (Chapter 4) that mimic the steering behaviour of real world crowds. The proposed framework successfully manages to capture the underlying crowd behaviour; i.e., given different examples of crowds for the same scenario, different behaviour patterns emerge (Section 4.5.2). The proposed approach outperforms traditional data-driven simulation techniques based on k-Nearest Neighbor queries by up an order of magnitude (Section 4.5.4). Quality measured by collision rate is constant and assuming that enough data are present, it is better than a traditional data-driven approach as it is discussed in Section 4.5. In most of the conducted experiments, quality is at least similar to traditional approaches. These results come to due to the pre-processing of the data in a graph structure that handles interactions between agents during simulation without requiring continuous searches in databases of examples. The processed data consist of Temporal Perception Patterns (TPPs) – a novel agent based state representation which captures temporal changes in the surrounding stimuli (such as other agents, static obstacles and relative movement to the agent).

Crowd Evaluation Evaluating the behaviour of simulated crowds has not been studied extensively in the literature. Crowd evaluation typically is typically performed either though user evaluations or by measuring some statistical measures over the simulations. This thesis addresses the crowd evaluation problem using a robust data-driven approach (Chapter 5) based on modern outlier detection techniques, including Pareto Depth Analysis. By following a data-driven approach, we manage to capture the benefits of using both user evaluations and statistics; i.e., a quantitative method of ranking behaviour that uses the implicit knowledge that real humans have of a crowd or virtual humans.

Simulations are compared to data from well behaved crowds; i.e., crowds that exhibit behaviour patterns that are desirable by the designers of a crowd system. The proposed approach is both multi-criteria and agent based in the sense that local and global characteristics of simulated data are evaluated using multiple statistical metrics from both the data and the simulations. The proposed metrics belong in two categories; individual that take into account the local properties of trajectory segments by themselves and inter-personal that take into account relationships between the agents. This approach can be used to non only successfully identify abnormal behaviours but also behaviours from the example data that the simulation system could not capture and therefore can help in developing better crowd simulation systems.

Higher Level Annotation of Crowds Finally, a method to extract higher level knowledge of pedestrian behaviour is proposed (Chapter 6) that successfully identifies and tags higher level behaviours such as walking, group formations and interactions. This approach can characterize both simulation and real world crowds with higher level annotations that can help in analyzing, describing and comparing crowds in an intuitive way. By annotating data of real crowds with higher level information better simulations and evaluations can be defined and studied. The proposed method is based on Random Forests and employs local and global features of trajectory segments to distinguish with a high success rate between six different classes of behaviours.

Crowd Simulation Tools and Data We believe that research in any field cannot move fast enough if there is no collaboration between scientists in the field. As such, we are planning on releasing all the tools, code and data developed over the course of the thesis to the community so that others can benefit from our work. To our knowledge, no other data-driven crowd simulation system is currently available to the community. Additionally, we are also planning on releasing the crowd evaluation and annotation tools to help other researchers debug their own crowd simulators or even compare their evaluation algorithms to our own. Finally, we plan to release data from our work on our website. Currently tools and data can be downloaded from our website (`http://graphics.cs.ucy.ac.cy`)[1] .

7.2 Future Work

All of the aforementioned techniques, open up the road for more research on data-driven virtual crowds. Even more so, we believe that these techniques can be used and exploited in other contexts also and not just virtual crowds. As part of future work, we are considering many different possibilities ranging from expanding each individual components separately to fusing them together into a complete crowd autoring framework. More specifically:

[1]Currently data and software can be found under: `https://graphics.cs.ucy.ac.cy/research/downloads/crowd-data`

Improving PAG Simulation *Reinforcement learning* could be used to enhance the proposed framework. In reinforcement learning based environment, decisions/actions are penalized or rewarded during simulation time and therefore this kind of an approach could be applied during the PAG traversal; edges could be rewarded according to collisions, error to the defined optimization function (Equation (4.3)) or even the evaluation scores defined in Chapter 5.

Data from the high level annotation (Chapter 6) should be used to enhance the simulation results produced by the proposed simulation framework (Chapter 4); i.e., during walks of the PAG graph actions should be selected based also on higher level behaviours. One potential alternative is to generate different PAGs for each high level behaviour and use the PAG corresponding to the current high level annotations. Additionally, data from different sources can be combined to improve the PAG and IFDB quality. Finally, the IFDB queries are based only on the current trajectory segment contrary to the PAG traversal which depends indirectly from previously selected actions. To improve on this, we could generate a graph like structure similar to the IFDB which will interconnect similar trajectory segments where traversal would be based on a different objective function than the one presented in Equation (4.3).

Data Requirements - State Space Exploration Studies should be performed to identify the amount of data that is needed to capture and simulate the required bahaviours. In one hand, it was observed in some of the datasets (e.g. the Reynolds dataset in Table 4.1) that by selecting just a subset of the data, most of the behaviours could be captured and adding more data to them simply did not improve simulation results. In a lot of the cases on the other hand we do not have enough data, since given a limited video of crowds it is very difficult and/or impossible to capture the entire state space. Therefore, a way to generate new states and actions based on the existing data could help in the quality of any data-driven simulator. In the case of the PAG framework for example, novel TPPs could be generated to fill up empty regions of the graph. These novel

TPPs could be either generated automatically without any user intervention or a designer could intervene to create new data by exploring the TPP space.

Crowd Authoring One of the most important topics in crowd simulation is *crowd authoring* (or scenario editing); i.e., the ability to easily author crowds exhibiting desired behaviours. A good direction to follow is looking at data-driven methods to author crowds using the methods proposed in this thesis. The main idea we are considering consists of mining the appropriate crowd data based on desired behaviours using semantic descriptions from the crowd analysis techniques proposed here (Chapter 6), then synthesizing these behaviours using the PAG framework (Chapter 4). This authoring framework should also incorporate detection of abnormal behaviours for easy and intuitive debugging of the crowd behaviours during synthesis (Chapter 5).

Simulating Local Groups Groups of pedestrians usually form in the real world, and as indicated in Chapter 6, most of the people in the real world ($60 - 75\%$) are parts of groups of 2–3 persons. Our framework cannot simulate these grouping behaviours as is; therefore approaches to extend the PAG framework to handle this crowds either implicitly without adding any rules or explicitly by adding constraints on the simulation. Another way of achieving small scale groups is by adding a data-driven high level controller that will be responsible for formations or groups and will again be learned from data of real world crowds. The higher level annotation framework proposed in Chapter 6 should be employed both for the learning phase and the simulation phase of such a data-driven crowd simulator.

Automatic Crowd Evaluation Evaluating the behaviour of crowds can be cumbersome for a non experienced user, especially if he needs to define a multitude of metrics. To this end, we should investigate automatic ways of selecting the appropriate metrics based on the source data. This means finding what kind of metrics describe the input data the best and then applying the proposed evaluation framework.

Improved Performance of Crowd Evaluation Generating and searching in high dimensional Pareto fronts is a very time consuming task, therefore a good alternative would be to find ways of projecting the data into a lower dimensional manifold aiming to keep the distance relationships between the segments as much as possible. We are also considering approaches to increase the performance of the PDA approach based on GPUs, such as representation of the Pareto space as textures.

Bibliography

[1] AHMED, N., NATARAJAN, T., AND RAO, K. Discrete cosine transform. *IEEE Transactions on Computers* (1974), 90–93.

[2] BREIMAN, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[3] BYERS, S., AND RAFTERY, A. E. Nearest-neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association 93*, 442 (1998), 577–584.

[4] CHARALAMBOUS, P., AND CHRYSANTHOU, Y. Learning Crowd Behavior. In *Workshop on Crowd Simulation, Collocated with the 23rd International Conference on Computer Animation and Social Agents (CASA2010)* (2010).

[5] CHARALAMBOUS, P., AND CHRYSANTHOU, Y. Learning Crowd Steering Behaviors from Examples. In *Proceedings of the Third International Conference on Motion in Games* (Berlin, Heidelberg, 2010), MIG'10, Springer-Verlag, pp. 35–35.

[6] CHARALAMBOUS, P., AND CHRYSANTHOU, Y. Classifying Pedestrian Behaviour using Random Forests, 2013. Poster presented at the 6th International Conference on Motion in Games 2013.

[7] CHARALAMBOUS, P., AND CHRYSANTHOU, Y. The PAG Crowd: A Graph Based Approach for Efficient Data-Driven Crowd Simulation. *Computer Graphics Forum* (2014). Under print.

[8] CHARALAMBOUS, P., ILIADOU, H., APOSTOLOU, C., AND CHRYSANTHOU, Y. Reconstruction of Everyday Life in 19^{th} Century Nicosia. In *Proceedings of the 4th International Conference on Progress in Cultural Heritage Preservation* (Berlin, Heidelberg, 2012), EuroMed'12, Springer-Verlag, pp. 568–577.

[9] CHARALAMBOUS, P., KARAMOUZAS, I., GUY, S., AND CHRYSANTHOU, Y. Data-Driven Crowd Analysis using Pareto Optimality. In preparation, 2014.

[10] CHENNEY, S. Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), Eurographics Association Aire-la-Ville, Switzerland, Switzerland, pp. 233–242.

[11] COURTY, N., AND CORPETTI, T. Crowd motion capture. *Computer Animation and Virtual Worlds 18*, 4-5 (2007), 361–370.

[12] Autodesk Softimage Crowdfx Plugin. `http://www.autodesk.com/products/autodesk-softimage/overview`.

[13] DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

[14] DRORI, I., COHEN-OR, D., AND YESHURUN, H. Fragment-based image completion. *ACM Trans. Graph. 22*, 3 (2003), 303–312.

[15] FIORINI, P., AND SHILLER, Z. Motion planning in dynamic environments using Velocity Obstacles. *International Journal of Robotics Research 17* (1998), 760–772.

[16] FRAILE, R., AND MAYBANK, S. Vehicle trajectory approximation and classification. In *British Machine Vision Conference* (1998).

[17] FUNGE, J., TU, X., AND TERZOPOULOS, D. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Siggraph 1999, Computer Graphics Proceedings* (Los Angeles, 1999), A. Rockwood, Ed., Addison Wesley Longman, pp. 29–38.

[18] Autodesk Gameware Navigation. `http://gameware.autodesk.com/navigation`.

[19] GERAERTS, R., AND OVERMARS, M. H. The corridor map method: A general framework for real-time high-quality path planning: Research articles. *Comput. Animat. Virtual Worlds 18*, 2 (May 2007), 107–119.

[20] Golaem Crowd Simulation for Autodest Maya. `http://www.golaem.com/`.

[21] GUY, S. J., CHHUGANI, J., CURTIS, S., DUBEY, P., LIN, M., AND MANOCHA, D. PLEdestrians: a least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), Eurographics Association, pp. 119–128.

[22] GUY, S. J., VAN DEN BERG, J., LIU, W., LAU, R., LIN, M. C., AND MANOCHA, D. A statistical similarity measure for aggregate crowd dynamics. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 190:1–190:11.

[23] HART, P., NILSSON, N., AND RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on 4*, 2 (1968), 100–107.

[24] HART, P., NILSSON, N., AND RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on 4*, 2 (july 1968), 100 –107.

[25] HEIGEAS, L., LUCIANI, A., THOLLOT, J., AND CASTAGNÉ, N. A physically-based particle model of emergent crowd behaviors. In *Graphicon* (2003).

[26] HELBING, D., AND MOLNÁR, P. Social force model for pedestrian dynamics. *Phys. Rev. E 51*, 5 (May 1995), 4282–4286.

[27] HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. Image analogies. In *SIGGRAPH '01: Proc. of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 327–340.

[28] HILDENBRANDT, H., CARERE, C., AND HEMELRIJK, C. K. Self-organized aerial displays of thousands of starlings: a model. *Behavioral Ecology 21*, 6 (2010), 1349–1359.

[29] HSIAO, K.-J., XU, K., CALDER, J., AND HERO, A. Multi-criteria anomaly detection using pareto depth analysis. In *Advances in Neural Information Processing Systems 25* (2012), pp. 854–862.

[30] HSU, E., PULLI, K., AND POPOVIC, J. Style translation for human motion. *ACM Trans. Graph. 24*, 3 (2005), 1082–1089.

[31] HUGHES, R. L. The Flow of Human Crowds. *Annual Review of Fluid Mechanics 35* (2003), 169–182.

[32] ILIADOU, H., CHARALAMBOUS, P., APOSTOLOU, C., AND CHRYSANTHOU, Y. Reviving Nicosia of the XIXth Century. In *11th International Conference on Urban History* (2012), EAUH'12.

[33] IRONI, R., COHEN-OR, D., AND LISCHINSKI, D. Colorization by example. In *Rendering Techniques* (2005), pp. 201–210.

[34] JU, E., CHOI, M., PARK, M., LEE, J., LEE, K., AND TAKAHASHI, S. Morphable crowds. *ACM Transactions on Graphics (TOG) 29*, 6 (2010), 140.

[35] KAPADIA, M., WANG, M., SINGH, S., REINMAN, G., AND FALOUTSOS, P. Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), pp. 53–62.

[36] KARAMOUZAS, I., AND OVERMARS, M. Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 394–406.

[37] KAVRAKI, L., SVESTKA, P., LATOMBE, J.-C., AND OVERMARS, M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on 12*, 4 (1996), 566–580.

[38] KIM, M., HWANG, Y., HYUN, K., AND LEE, J. Tiling motion patches. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation* (2012), Eurographics Association, pp. 117–126.

[39] KOVAR, L., GLEICHER, M., AND PIGHIN, F. Motion graphs. In *ACM Transactions on Graphics (TOG)* (2002), vol. 21, ACM, pp. 473–482.

[40] KOVAR, L., GLEICHER, M., AND PIGHIN, F. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 473–482.

[41] KUSHLEYEV, A., MELLINGER, D., POWERS, C., AND KUMAR, V. Towards a swarm of agile micro quadrotors. *Autonomous Robots 35*, 4 (2013), 287–300.

[42] KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3 (2003), 277–286.

[43] KWON, T., LEE, K. H., LEE, J., AND TAKAHASHI, S. Group motion editing. *ACM Transactions on Graphics (TOG) 27*, 3 (Aug. 2008), 80:1–80:8.

[44] Autodesk Kynapse - Artificial Intelligence Middleware for Games. `http://gameware.autodesk.com/kynapse`.

[45] LAI, Y.-C., CHENNEY, S., AND FAN, S. Group motion graphs. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 281–290.

[46] LAMARCHE, F., AND DONIKIAN, S. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Comput. Graph. Forum 23*, 3 (2004), 509–518.

[47] LEE, J.-G., HAN, J., LI, X., AND GONZALEZ, H. TraClass: Trajectory Classification Using Hierarchical Region-based and Trajectory-based Clustering. *Proc. VLDB Endow. 1*, 1 (Aug. 2008), 1081–1094.

[48] LEE, K., CHOI, M., HONG, Q., AND LEE, J. Group behavior from video: a data-driven approach to crowd simulation. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), 109–118.

[49] LEE, K. H., CHOI, M. G., AND LEE, J. Motion patches: building blocks for virtual environments annotated with motion data. In *ACM Transactions on Graphics (TOG)* (2006), vol. 25, ACM, pp. 898–906.

[50] Legion Science in Motion. `http://www.legion.com`.

[51] LERNER, A., CHRYSANTHOU, Y., AND LISCHINSKI, D. Crowds by Example. *Computer Graphics Forum 26*, 3 (2007), 655–664.

[52] LERNER, A., CHRYSANTHOU, Y., SHAMIR, A., AND COHEN-OR, D. Context-dependent crowd evaluation. *Computer Graphics Forum 29*, 7 (2010), 2197 – 2206.

[53] LERNER, A., FITUSI, E., CHRYSANTHOU, Y., AND COHEN-OR, D. Fitting behaviors to pedestrian simulations. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), ACM, pp. 199–208.

[54] LI, Y., CHRISTIE, M., SIRET, O., KULPA, R., AND PETTRÉ, J. Cloning crowd motions. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2012), SCA '12, Eurographics Association, pp. 201–210.

[55] LIM, I. S., AND THALMANN, D. A vector-space representation of motion data for example-based motion synthesis. In *Proceedings of the IFIP TC5/WG5.10 DEFORM'2000 Workshop and AVATARS'2000 Workshop on Deformable Avatars* (Deventer, The Netherlands, The Netherlands, 2001), DEFORM '00/AVATARS '00, Kluwer, B.V., pp. 169–179.

[56] LOYALL, A. B., BATES, J., AND MITCHELL, T. Believable agents: Building interactive personalities. Tech. rep., 1997.

[57] Massive Software - Simulating Life. `http://www.massivesoftware.com/`.

[58] Autodesk Maya. `http://www.autodesk.com/maya`.

[59] McDonnell, R., Larkin, M., Dobbyn, S., Collins, S., and O'Sullivan, C. Clone attack! perception of crowd variety. *ACM Transactions on Graphics (TOG) 27*, 3 (Aug. 2008), 26:1–26:8.

[60] Metoyer, R. A., and Hodgins, J. K. Reactive pedestrian path following from examples. In *Proc. of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)* (Washington, DC, 2003), IEEE Comp. Soc., p. 149.

[61] Moussaïd, M., Helbing, D., Garnier, S., Johansson, A., Combe, M., and Theraulaz, G. Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proceedings of the Royal Society B: Biological Sciences 276*, 1668 (2009), 2755.

[62] Moussaïd, M., Perozo, N., Garnier, S., Helbing, D., and Theraulaz, G. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PLoS One 5*, 4 (2010), e10047.

[63] Musse, S. R., Jung, C. R., Braun, A., and Junior, J. J. Simulating the motion of virtual agents based on examples. In *ACM/EG Symposium on Computer Animation, Short Papers* (Vienna, Austria, 2006).

[64] Narain, R., Golas, A., Curtis, S., and Lin, M. C. Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics (TOG) 28*, 5 (Dec. 2009), 122:1–122:8.

[65] Nørretranders, T. *The user illusion: Cutting consciousness down to size.* Viking, 1991.

[66] ONDŘEJ, J., PETTRÉ, J., OLIVIER, A., AND DONIKIAN, S. A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 123.

[67] OWENS, J., AND HUNTER, A. Application of the self-organising map to trajectory classification. In *Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on* (2000), pp. 77–83.

[68] PARIS, S., PETTRÉ, J., AND DONIKIAN, S. Pedestrian Reactive Navigation for Crowd Simulation: a Predictive Approach. *Computer Graphics Forum 26*, 3 (2007), 665–674.

[69] PARSEVAL DES CHÉNES, M. A. Mémoire sur les séries et sur l'intégration compléte d'une équation aux différences partielles linéaire du second ordre, á coefficients constants. *Mémoires présentés par divers savants 1* (1806), 638–648.

[70] PEJSA, T., AND PANDZIC, I. State of the art in example-based motion synthesis for virtual characters in interactive applications. *Computer Graphics Forum 29*, 1 (2010), 202–226.

[71] PELECHANO, N., ALLBECK, J., AND BADLER, N. *Virtual Crowds: Methods, Simulation, and Control*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2008.

[72] PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association, pp. 99–108.

[73] PELECHANO, N., O'BRIEN, K., SILVERMAN, B., AND BADLER, N. Crowd simulation incorporating agent psychological models, roles and communication. In *First International Workshop on Crowd Simulation, (V-CROWDS '05).* (November 2005), pp. 24–25.

[74] PELECHANO, N., STOCKER, C., ALLBECK, J., AND BADLER, N. Being a part of the crowd: towards validating VR crowds using presence. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1* (2008), International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, pp. 136–142.

[75] PELLEGRINI, S., ESS, A., AND VAN GOOL, L. Improving data association by joint modeling of pedestrian trajectories and groupings. *Computer VisionECCV 2010* (2010), 452–465.

[76] PERLIN, K., AND GOLDBERG, A. Improv: A System for Scripting Interactive Actors in Virtual Worlds. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 205–216.

[77] PETTRÉ, J., CIECHOMSKI, P. D. H., MAÏM, J., YERSIN, B., LAUMOND, J.-P., AND THALMANN, D. Real-time navigating crowds: scalable simulation and rendering. *Computer Animation and Virtual Worlds 17*, 3-4 (2006), 445–455.

[78] PETTRÉ, J., ONDŘEJ, J., OLIVIER, A., CRETUAL, A., AND DONIKIAN, S. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), ACM, pp. 189–198.

[79] REITSMA, P. S., AND POLLARD, N. S. Evaluating motion graphs for character animation. *ACM Transactions on Graphics (TOG) 26*, 4 (2007), 18.

[80] REN, L., SHAKHNAROVICH, G., HODGINS, J., PFISTER, H., AND VIOLA, P. Learning silhouette features for control of human motion. *ACM Transactions on Graphics (TOG) 24*,

4 (2005), 1303–1331.

[81] REYNOLDS, C. Flocks, herds, and schools: A distributed behavioral model. *Comp. Graph. 21*, 4 (1987), 25–34.

[82] REYNOLDS, C. Steering behaviors for autonomous characters. *Game Developers Conference 1999* (1999).

[83] REYNOLDS, C. Big fast crowds on PS3. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames* (New York, NY, USA, 2006), Sandbox '06, ACM, pp. 113–121.

[84] SANKARANARAYANAN, J., ALBORZI, H., AND SAMET, H. Efficient query processing on spatial networks. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems* (New York, NY, USA, 2005), GIS '05, ACM, pp. 200–209.

[85] SCHNITMAN, Y., CASPI, Y., COHEN-OR, D., AND LISCHINSKI, D. Inducing semantic segmentation from an example. In *ACCV (2)* (2006), pp. 373–384.

[86] SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. Video textures. In *Proc. of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 489–498.

[87] SCHÖLKOPF, B., WILLIAMSON, R. C., SMOLA, A. J., SHAWE-TAYLOR, J., AND PLATT, J. Support vector method for novelty detection. *Advances in neural information processing systems 12*, 4 (2000), 582–588.

[88] scikit-learn: Machine Learning in Python. `http://scikit-learn.org`.

[89] SHAO, W., AND TERZOPOULOS, D. Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 19–28.

[90] SHARF, A., ALEXA, M., AND COHEN-OR, D. Context-based surface completion. *ACM Trans. Graph. 23*, 3 (2004), 878–887.

[91] SHIMODA, S., KURODA, Y., AND IAGNEMMA, K. Potential field navigation of high speed unmanned ground vehicles on uneven terrain. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on* (2005), IEEE, pp. 2828–2833.

[92] SHOULSON, A., MARSHAK, N., KAPADIA, M., AND BADLER, N. I. ADAPT: the agent development and prototyping testbed. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2013), ACM, pp. 9–18.

[93] SINGH, S., KAPADIA, M., FALOUTSOS, P., AND REINMAN, G. Steerbench: a benchmark suite for evaluating steering behaviors. *Comput. Animat. Virtual Worlds 20*, 5/6 (2009), 533–548.

[94] SINGH, S., NAIK, M., KAPADIA, M., FALOUTSOS, P., AND REINMAN, G. Watch Out! A Framework for Evaluating Steering Behaviors. In *Motion in Games: First International Workshop, MIG 2008, Utrecht, the Netherlands, June 14-17, 2008, Revised Papers* (2008), Springer-Verlag New York Inc, p. 200.

[95] The Soul of The Sims, by Will Wright. http://www.donhopkins.com/home/images/Sims/.

[96] TERZOPOULOS, D., TU, X., AND GRZESZCZUK, R. Artificial fishes: autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artif. Life 1*, 4 (1994), 327–351.

[97] THALMANN, D., AND MUSSE, S. R. *Crowd Simulation.* Springer, 2007.

[98] TORRENS, P., LI, X., AND GRIFFIN, W. A. Building agent-based walking models by machine-learning on diverse databases of space-time trajectory samples. *Transactions in GIS 15* (2011), 67–94.

[99] TREUILLE, A., COOPER, S., AND POPOVIC, Z. Continuum crowds. *ACM Trans. Graph. 25*, 3 (2006), 1160–1168.

[100] UBISOFT MONTREAL. Assassin's Creed II, 2009.

[101] Unity Game Engine. `http://unity3d.com/unity`.

[102] VAN BASTEN, B., JANSEN, S., AND KARAMOUZAS, I. Exploiting motion capture to enhance avoidance behaviour in games. *Motion in Games* (2009), 29–40.

[103] VAN DEN BERG, J., FERGUSON, D., AND KUFFNER, J. Anytime path planning and replanning in dynamic environments. In *Robotics and Automation, (ICRA) 2006* (may 2006), pp. 2366 –2371.

[104] VAN DEN BERG, J., GUY, S. J., LIN, M., AND MANOCHA, D. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium ISRR* (2011), vol. 70 of *Springer Tracts in Advanced Robotics*, Springer, pp. 3–19.

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

[106] Warren, C. W. Global path planning using artificial potential fields. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on* (1989), IEEE, pp. 316–321.

[107] Yersin, B., Maim, J., Pettré, J., and Thalmann, D. Crowd patches: populating large-scale virtual environments for real-time applications. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (2009), ACM, pp. 207–214.

[108] Yu, Q., and Terzopoulos, D. A decision network framework for the behavioral animation of virtual humans. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 119–128.

[109] Zhang, Y., Correa, C., and Ma, K. Graph-based fire synthesis. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM, pp. 187–194.

[110] Zhao, M., and Saligrama, V. Anomaly detection with score functions based on nearest neighbor graphs. In *Advances in Neural Information Processing Systems* (2009).

Appendices

Appendix A

Simulation Performance of PAG

The following paragraphs describe timing and quality results for the PAG simulation framework as these are described in Section 4.5.4. Each experiment compares timing and collision quality of the same dataset against two different scenarios.

A.1 Pedestrians 1 (lerner-zara)

In both scenarios, (Figures A.1, A.2) the PAG approach outperforms kNN having at worst 10% quality decrease and at best 50% quality increase. At a value of 100 for k, collision behaviour is similar with a performance speedup of 700-750%. Notice also that collision behaviour is stable over different k, indicating a well connected graph with enough actions to resolve possible interactions even if the initial node selection is not the best possible.

A.2 Pedestrians 2 (eth-hotel)

As the number of nearest neighbors k increases, the PAG collision quality reaches that of the kNN, but never (at least up to 100 nearest neighbors) surpassing it (Figures A.4) A.3) . In both cases, PAG performance increases as k increases (up to 750-800%). Another interesting observation, is that the collision quality remains more or less the same (bottom left graphs in Figures A.3 and A.4) with the PAG in both cases; even though we search for more examples and the quality of the kNN drops over the number of k. This indicates that probably the closest match is the most appropriate in most cases. Also, this indicates; as in most of the other experiments that the PAG is a more stable algorithm; it's behaviour is almost independent on the number of examples retrieved due to the edge traversal.

A.3 Small Groups (lee-chat)

This a very small dataset consisting of people talking. Due to its size, the PAG graph has very few connections, resulting in bad collision quality due to few edge choices and few nodes (Figures A.5 and A.6). Collision quality increases as k increases indicating that probably a brute search over all nodes would be better suited for this small dataset; i.e., states found in the simulation can't be found in the source data and therefore actions in different states can help solve the interaction as collision free as possible. Also, notice that by introducing a jump threshold the PAG can get similar collision behaviour to the kNN with significant speed gains since a problematic situation is identified and a different node is selected to handle the issue. Notice again that the PAG has stable behaviour over different number of examples whereas kNN and variations of PAG with different jump thresholds have variations in their quality. Performance increases by more than 800%, keeping the simulation real time (¿25 fps) for even large k in contrast to the kNN approach which drops below real time for $k = 5$.

A.4 Simulated Data (reynolds-pedestrians)

Data in this dataset consist of agents in a simulated environment avoiding each other; mostly from the opposite direction. The quality results indicate that the behaviour in this simulation is fairly simple as by just a few number of retrieved examples, collisions occur at almost the same rate in most of the approaches. In the simper Manhattan like scenario (Figure A.7), quality is improved by up to 40% if the PAG is employed with a threshold and up 10% for full PAG, indicating that quality increases by using history (i.e., the effect of traversing edges). In the more complex Chinese Hanzi scenario though (Figure A.8), where paths are more irregular and are intersecting at different angles, quality drops by up to 25% for PAG without thresholds but still remains stable whereas by introducing a jump threshold it can pass kNN by 10-40%. Performance is again over 700% with simulation being real time over all k.

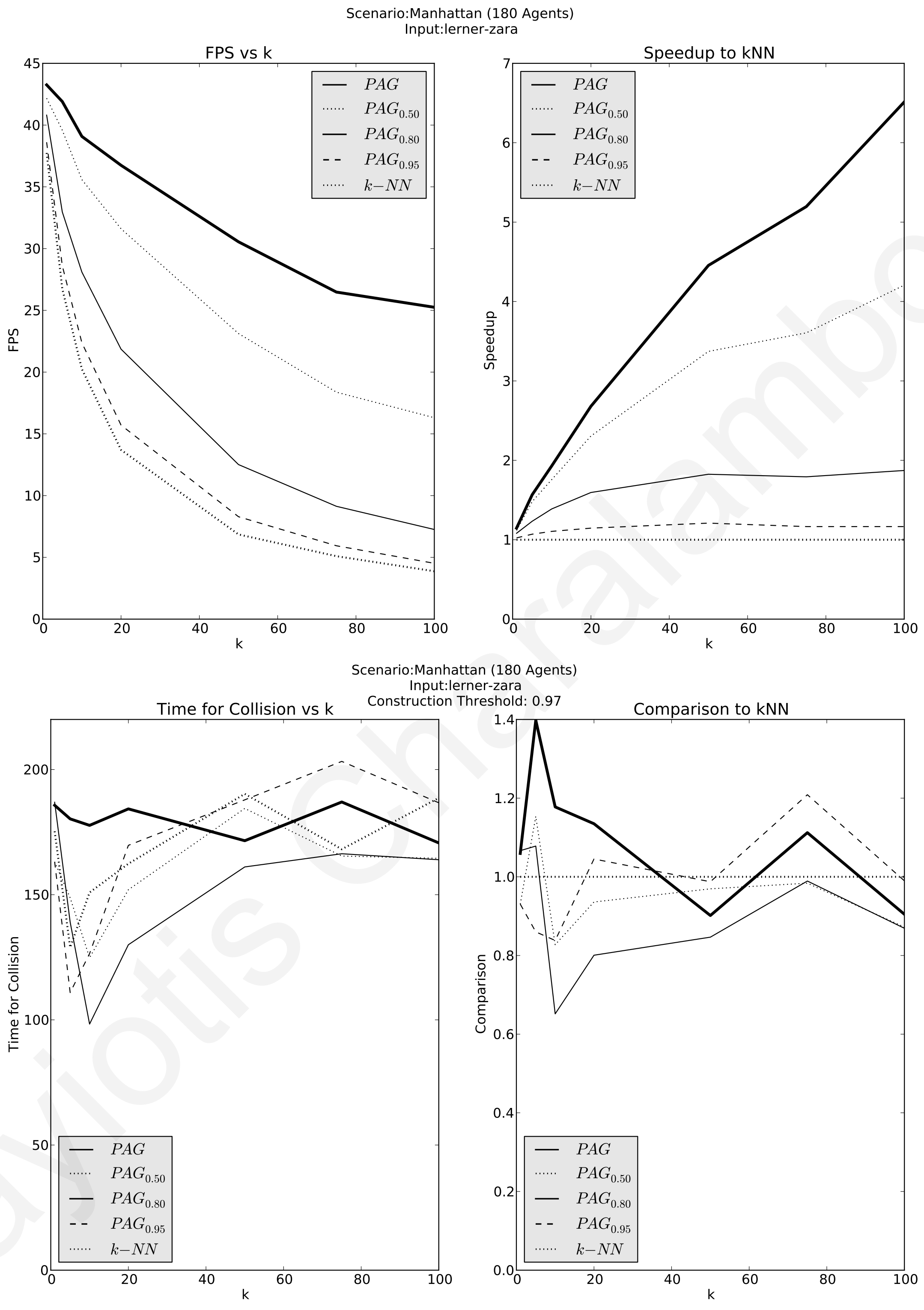


Figure A.1: Comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 180 agents in a scenario similar to Section 4.5.2.5 with the lerner-zara dataset. The PAG approach outperforms kNN up to 8 times with similar collision behaviour.

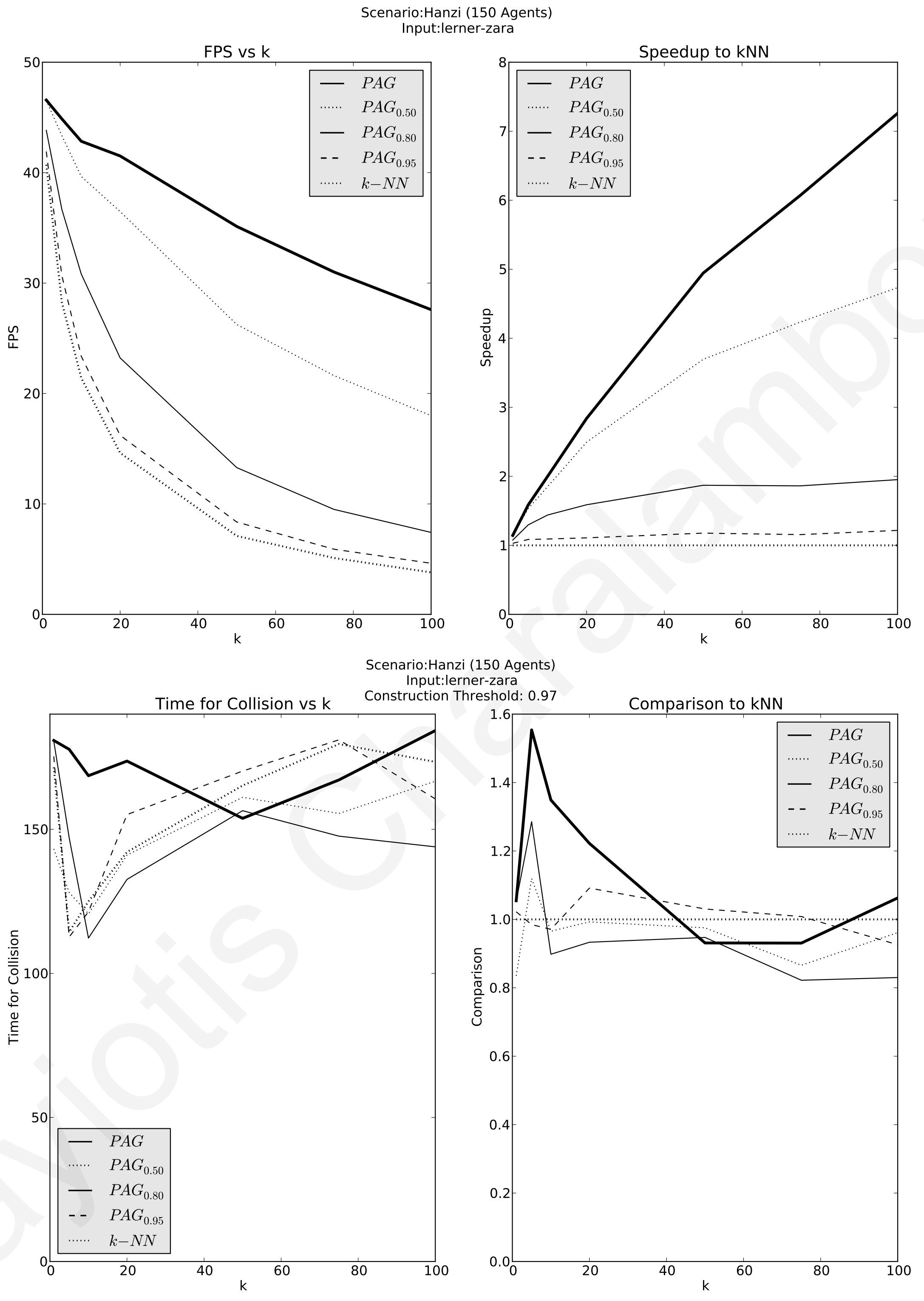


Figure A.2: Comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 150 agents in a scenario similar to Section 4.5.2.6 using as input the *lerner-zara* dataset. The PAG approach outperforms kNN up to 8 times with similar collision behaviour.

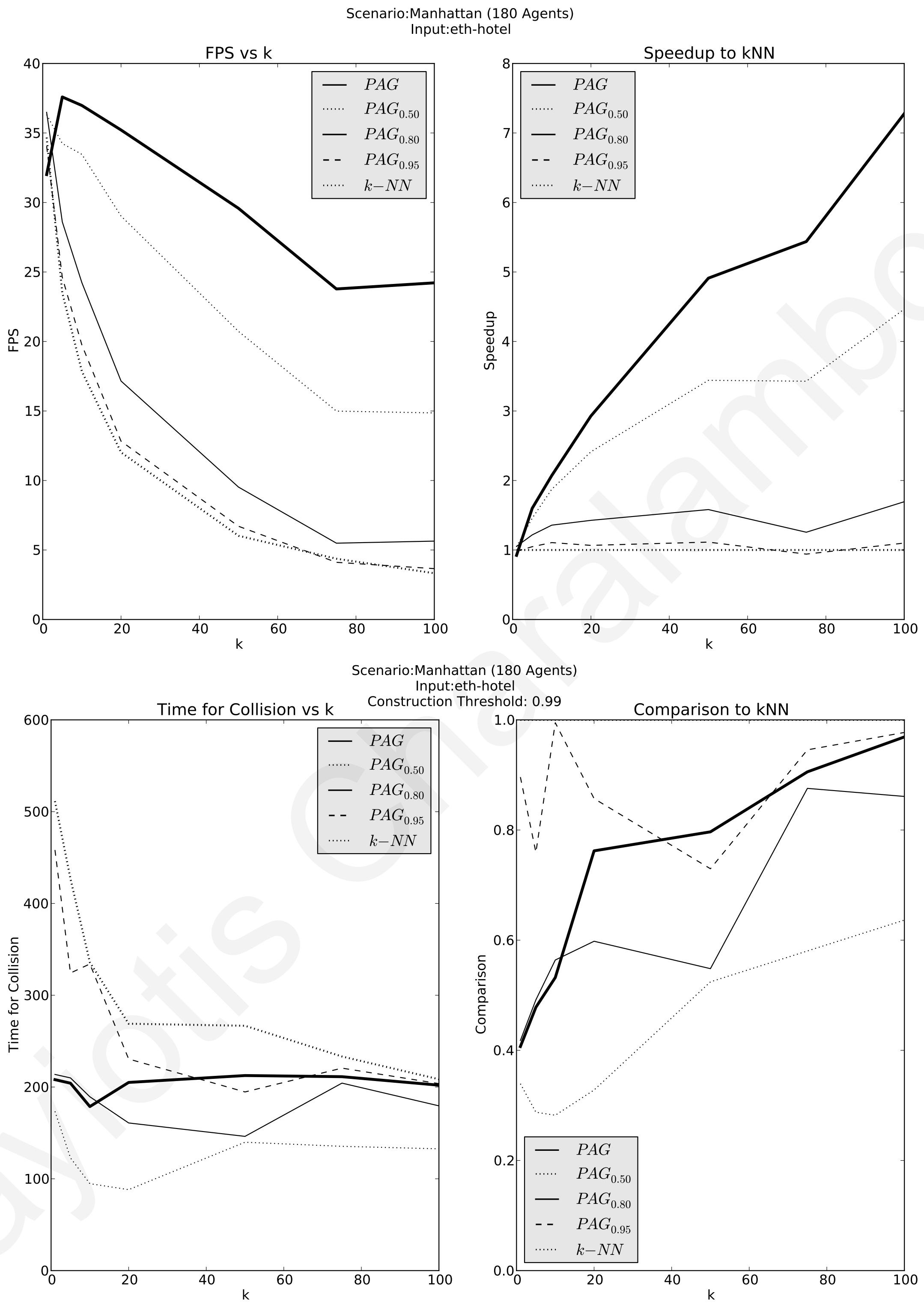


Figure A.3: Comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 180 agents in a scenario similar to Section 4.5.2.5 with the eth-hotel dataset.

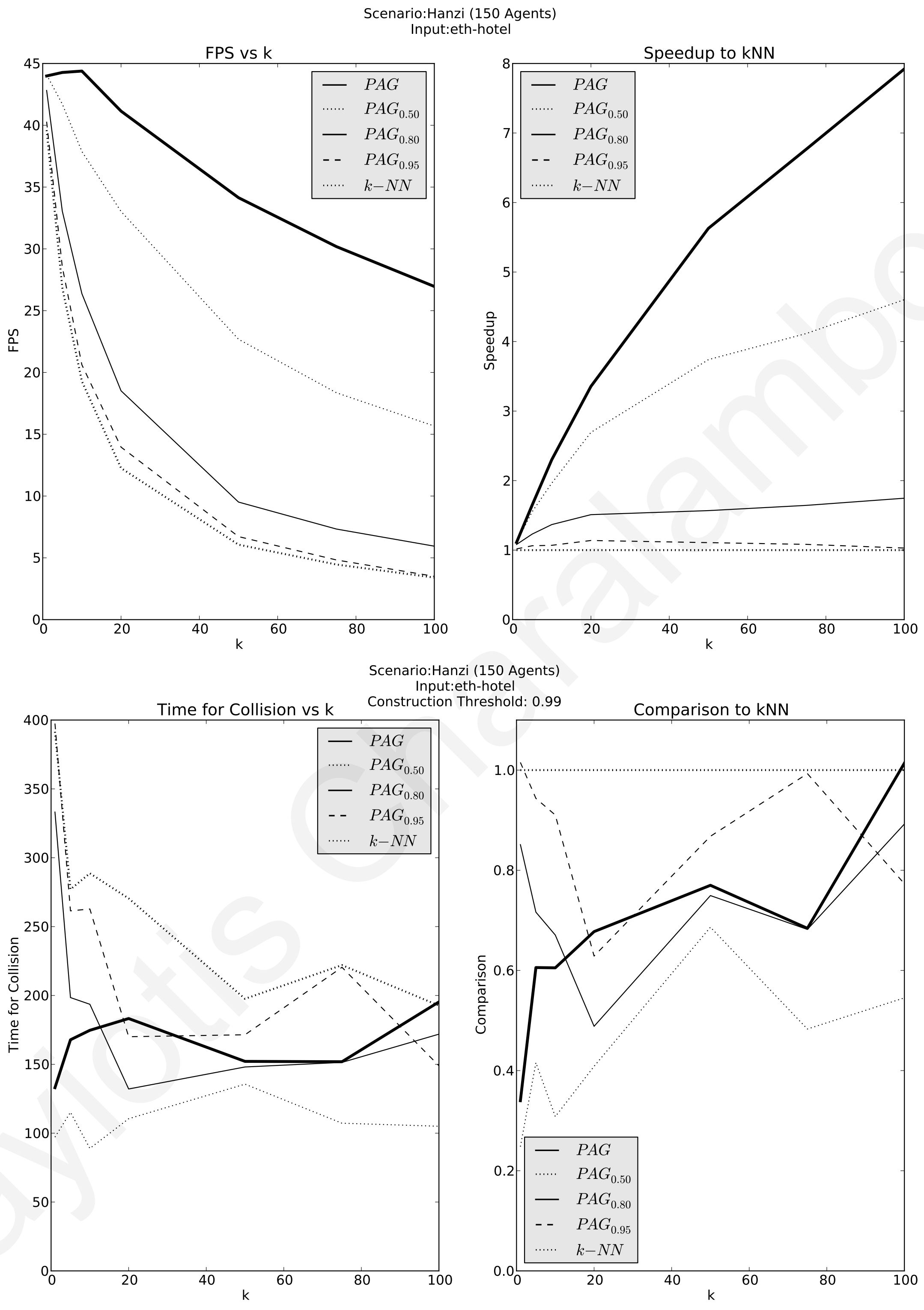


Figure A.4: Timing comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 150 agents in a scenario similar to Section 4.5.2.6 with the eth-hotel dataset.

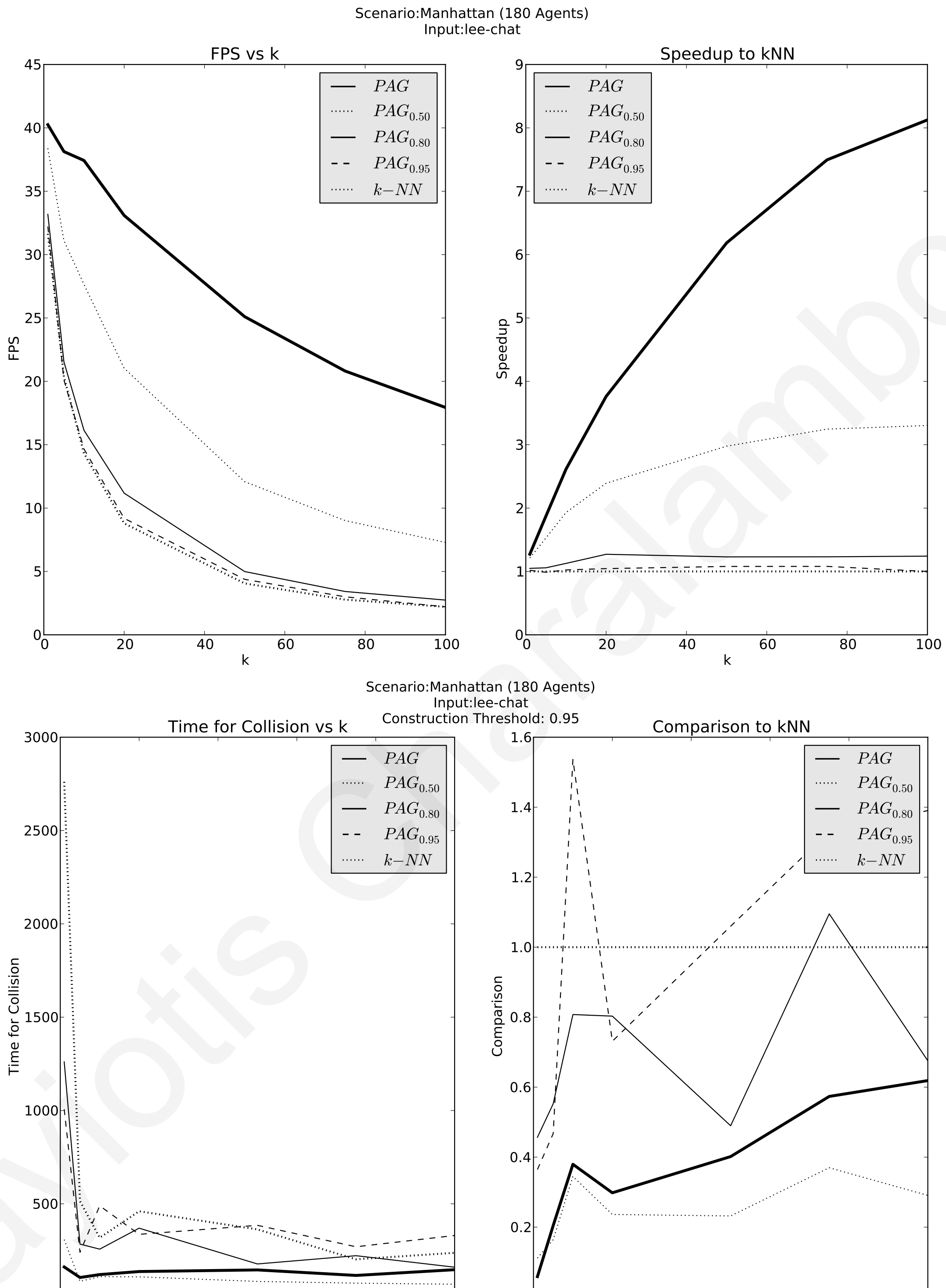
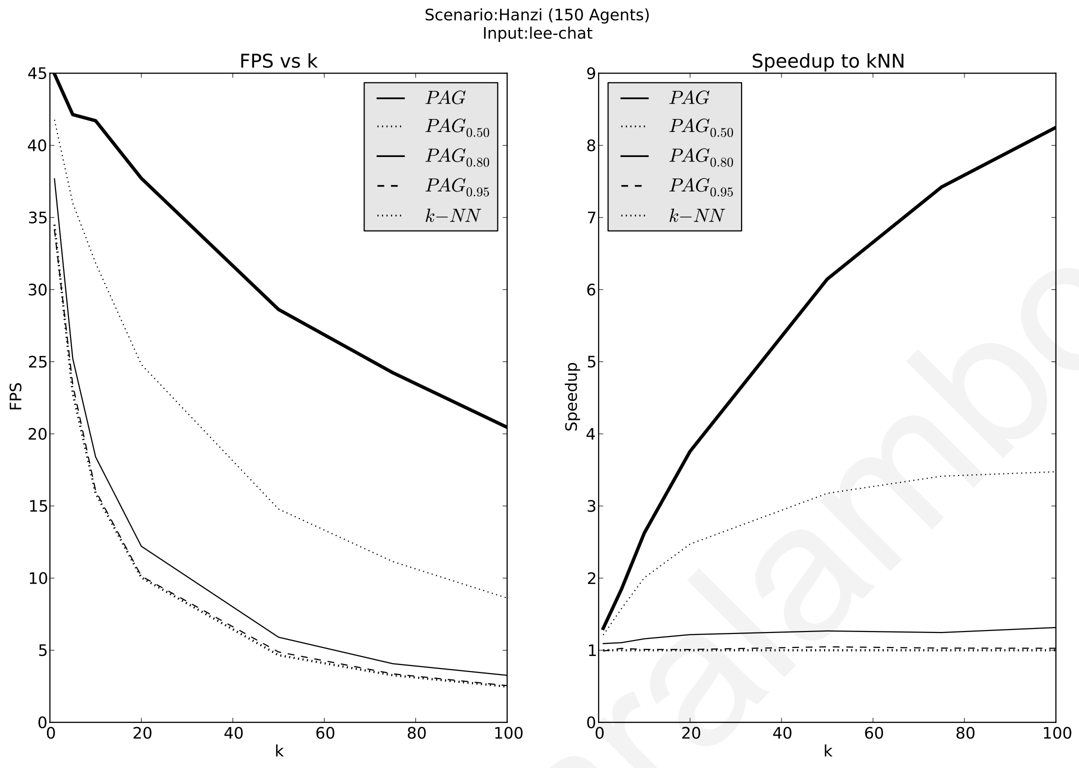


Figure A.5: Comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 180 agents in a scenario similar to Section 4.5.2.5 with the lee-chat dataset.



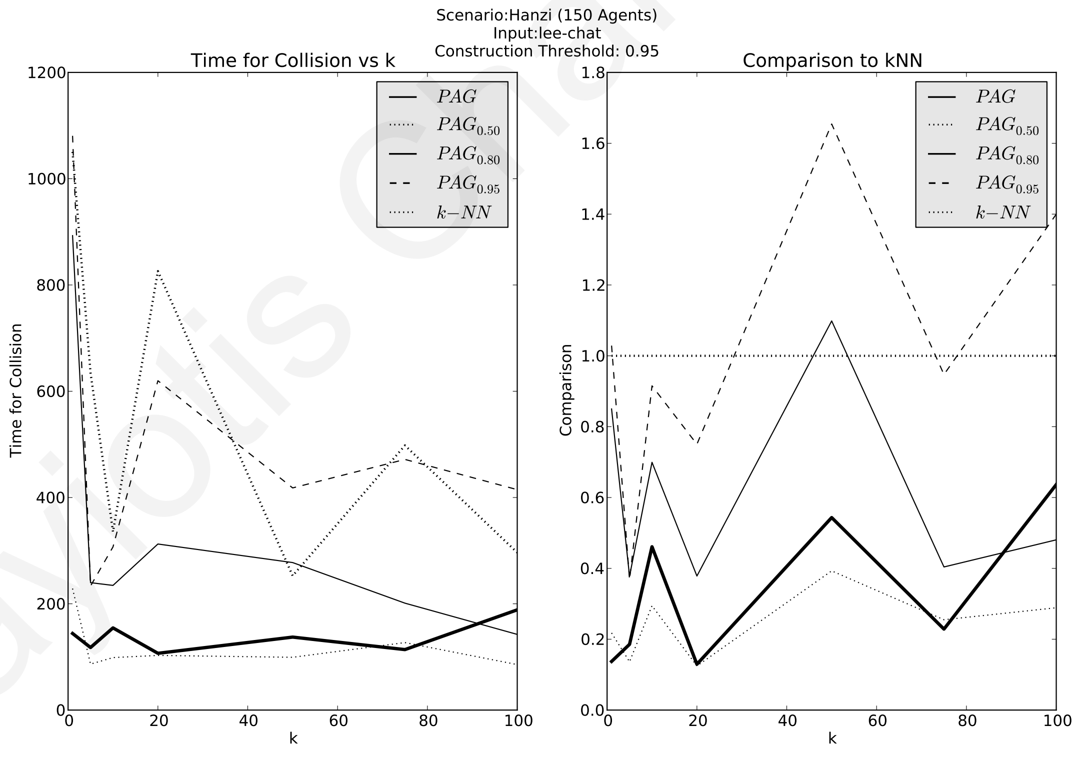


Figure A.6: Comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 150 agents in a scenario similar to Section 4.5.2.6 with the lee-chat dataset.

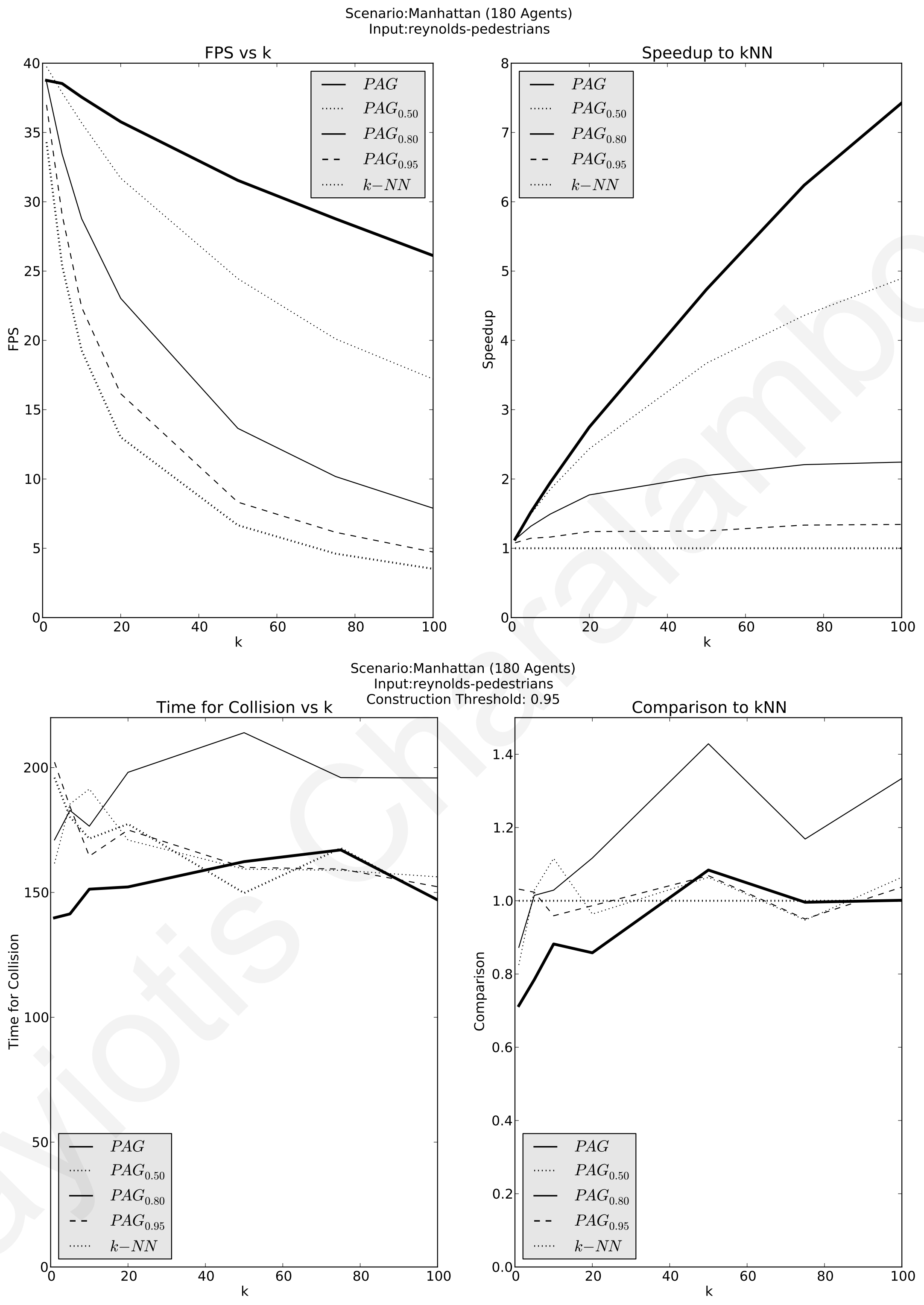


Figure A.7: Comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 180 agents in a scenario similar to Section 4.5.2.5 with the synthetic reynolds-hotel dataset. The PAG approach outperforms kNN up to 8 times.

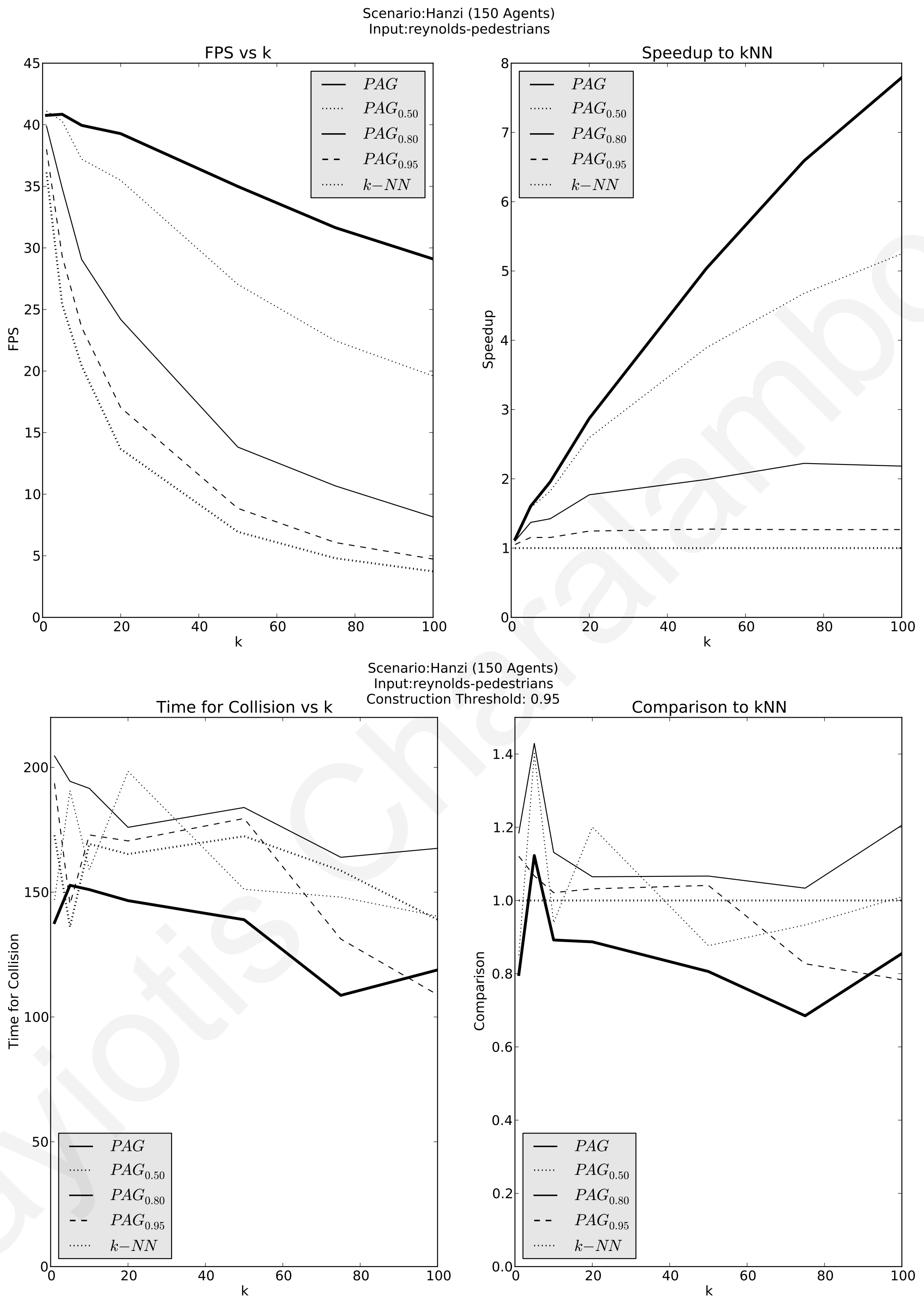


Figure A.8: Timing comparison between the PAG against itself (with varying jump thresholds) and the kNN approach. Simulation is 150 agents in a scenario similar to Section 4.5.2.6 with the synthetic reynolds-hotel dataset. The PAG approach outperforms kNN up to 8 times.