

**DOCTORAL PROGRAM IN
INFORMATION AND COMMUNICATION TECHNOLOGY**

**Doctoral candidate
Niccolò Bisagno**

Cycle	32
Thesis	On simulating and predicting pedestrian trajectories in a crowd
Advisor	Nicola Conci (University of Trento)

1. List of publications

Journal publications and book chapters

- Conci, Nicola, Niccolò Bisagno, and Andrea Cavallaro. "On modeling and analyzing crowds from videos." Computer Vision for Assistive Healthcare. Academic Press, 2018. 319-336.

Conference publications

- Bisagno, Niccolò, Bo Zhang, and Nicola Conci. "Group LSTM: Group trajectory prediction in crowded scenarios." Proceedings of the European conference on computer vision (ECCV). 2018.
- Bisagno, Niccolò, Nicola Conci, and Bo Zhang. "Data-Driven crowd simulation." 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 2017.
- Bisagno, Niccolò, and Nicola Conci. "Virtual camera modeling for multi-view simulation of surveillance scenes." 2018 26th European Signal Processing Conference (EUSIPCO). IEEE, 2018.
- Bisagno, Niccolò, Nicola Conci, and Bernhard Rinner. "Dynamic Camera Network Reconfiguration for Crowd Surveillance." Proceedings of the 12th International Conference on Distributed Smart Cameras (ICDSC). 2018.
- Bisagno, Niccolò, Nicola Garau, Andrea Montagner, and Nicola Conci. "Virtual Crowds: An LSTM-Based Framework for Crowd Simulation." International Conference on Image Analysis and Processing (ICIAP). Springer, Cham, 2019.

Abstracts and demos

- Rossi, Giulia, Niccolò Bisagno, and Aronne Armanini. "Dry granular flows of monodisperse particles: an optical method to compute the particles flow field." EGU General Assembly Conference Abstracts. Vol. 20. 2018.
- Bisagno, Niccolò, and Cristian Iacovlev. "Camera network optimization: maximize coverage in a 3D virtual environment." Proceedings of the 13th International Conference on Distributed Smart Cameras (ICDSC). 2019.

Under review

- Bisagno, Niccolò, Cristiano Saltori, Bo Zhang, and Nicola Conci. "Embedding group and obstacle information in LSTM networks for human trajectory prediction in crowded scenes", Under review in Computer Vision and Image Understanding (CVIU), 2020.
- Zaeemzadeh, Alireza, Niccolò Bisagno, Nicola Conci, Nazanin Rahnavard, and Mubarak Shah. "Out-of-Distribution Detection Using Union of 1-Dimensional Subspaces". Submitted to International Conference on Machine Learning (ICML), 2020.

- Song, Yue, Niccolò Bisagno, Syed Zohaib Hassan, and Nicola Conci. “AG-GAN: An Attentive Group-Aware GAN for pedestrian trajectory prediction” Submitted to IEEE Transactions on Image Processing (TIP), 2020.

2. Research/study activities

During the first year of the Ph.D. I focused on modeling the high-level behavior of the crowd in a simulated environment. In the second year, I shifted my research activities toward the prediction of pedestrian trajectories using deep learning models. In the third year, especially during my period abroad in the USA, I focused my research work on deep learning techniques applied to open set networks. Throughout the three years, I have engaged in multiple teaching activities. I taught the lab session of the Computer Vision master course for 3 years and I supervised many projects and thesis by several students. A summary of my research activities can be found here below.

Best research poster/demo awards

- Bisagno, Niccolò. “Crowd simulation and surveillance”. ICT Days, Trento, Italy, March 1-10, 2017
- Bisagno, Niccolò, Nicola Garau, and Andrea Montagner. “Modeling and Observing Virtual Crowds”. GTTI - Thematic Meeting on Multimedia Signal Processing, Cavalese (TN), Italy, January 21-23, 2018

Mobility grant

- Awarded the scholarship for “International mobility for doctoral research abroad” by the University of Trento

Research and training periods spent abroad

- April 2018, Universität Klagenfurt, Austria. Working on distributed camera systems in a simulated environment under the supervision of Prof. Bernhard Rinner
- March-July 2019, University of Central Florida, USA. Conducting research open-set deep networks under the supervision of Prof. Mubarak Shah

Conferences and Summer schools

- 2017 GTTI - Thematic Meeting on Multimedia Signal Processing
- 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)
- 2017 IEEE-EURASIP S3P-2017 Summer School: “Signal processing meets deep learning”
- 2018 GTTI - Thematic Meeting on Multimedia Signal Processing
- 2018 Joint Annual Meeting of GTTI-CNIT
- 2018 12th International Conference on Distributed Smart Cameras (ICDSC)
- 2018 26th European Signal Processing Conference (EUSIPCO)
- 2018 European Conference on Computer Vision (ECCV)
- 2019 Computer Vision and Pattern Recognition (CVPR)
- 2019 13th International Conference on Distributed Smart Cameras (ICDSC)
- 2019 20th International Conference on Image Analysis and Processing (ICIAP)

Teaching activities

Teaching

- Fall 2017-2018-2019, Teaching assistant for the Computer Vision and Multimedia Analysis course: teaching and development of lab sessions for the master course.

Thesis co-supervisor

Master thesis

- Garau, Nicola. “Simulating virtual crowds using LSTM networks”, 2018
- Montagner, Andrea. “Motion prediction and personalisation in virtual crowds”, 2018
- Hassan, Syed Zohaib. “Trajectory prediction and simulation via generative adversarial networks”, 2018

Bachelor thesis

- Formolo, Giulia. “Valutazione della coerenza del moto nella simulazione di scene affollate”
- Saltori, Cristiano. “Deep convolutional neural network for people counting in crowded scenes.”
- Teatini, Alex. “Un approccio basato su simulazione per analisi video di scene affollate”
- Iacovlev, Cristian. “Camera network optimization: maximize coverage in a 3D virtual environment”
- Scandolara, Alberto. “Sviluppo di un’estensione grafica in unity del simulatore di folle Menge”
- De Pellegrini, Martin. “Simulazione e virtualizzazione di algoritmi di tracciamento oggetti in ambiente Unity3D”
- Cracco, Davide. “Metriche di valutazione per l’analisi della qualità della simulazione di scene affollate”
- Barbisan, Luca. “Sviluppo di un simulatore di folle in Unity3D”



UNIVERSITÀ DEGLI STUDI DI TRENTO

Department of Information Engineering and
Computer Science - DISI

ICT International Doctoral School

FINAL DISSERTATION

ON SIMULATING AND PREDICTING
PEDESTRIAN TRAJECTORIES IN A
CROWD

Supervisor
Nicola Conci

Candidate
Niccoló Bisagno

Academic year 2018/2019

Ringraziamenti

Vorrei ringraziare:

- *il mio advisor Nicola per i tuoi consigli e supporto, tecnico e umano. Sono stati quasi 4 anni di tanti alti e alcuni bassi, ma, come dici tu, il dottorato (e la vita) sono come una sinusoidale e quando si arriva in basso si tornerà sempre ad un nuovo alto.*
- *Nicola e Andre. Siete stati miei studenti, tesisti, colleghi, compagni di mattine e notti a risolvere bug e problemi, ma soprattutto amici. Grazie per avermi sopportato e per continuare a sopportarmi. Continuerò a essere quello "che fa le cose".*
- *Genc, Mesay, Claus, Giulia, Alain, Trobi, Giacomino, Giulio, Bru e gli altri "MASTER TLC", tutti gli "Stiamo Malon" e i Pilipini. Avete reso le lunghe giornate passate a Povo estremamente piacevoli e ricche di belle relazioni.*
- *Rol, Tomino, Mike, Simo, Gio, Beppo e tutti i BlueBears. La pallacanestro unisce e avvicina persone speciali in maniera unica. Anche qualche birretta aiuta.*
- *i miei amici del vittoriese: gli Age of Nerdiress, la Consu, i vecchi Animatori, quelli del Jebbab, la Zia Fernanda. Tornare a trovarvi regala sempre qualcosa di nuovo.*
- *Elisabetta, perchè insieme a me hai combattuto più battaglie di tutti, anche se alla fine abbiamo perso.*
- *Berto, Luca, Dodo, Isi, Della. Amici, compagni di viaggio, di sventure e avventure, di cose belle e cose brutte. Magari qualche volta ci perdiamo un po', ma quando serve ci siamo sempre.*
- *mio fratello Tommaso, con Elena e i miei nipoti, Damiano e Stefano. Siete per me un luogo di conforto e un grande punto di riferimento.*
- *la mia Mamma e il mio Papà, che mi supportano e mi vogliono bene incondizionatamente. Siete il mio porto sicuro, vi voglio bene.*

Contents

Introduction	4
Context	4
Problems and solutions	4
Trajectory Prediction in Crowded Scenarios	5
Crowd simulation	6
Crowd simulation Applications	7
Toward anomaly detection in the wild: out of distribution detection	8
Structure of the thesis	9
1 Trajectory Prediction in Crowded Scenarios	10
1.1 Introduction	10
1.2 Contribution	12
1.3 Problem Statement	12
1.4 Related Work	13
1.4.1 Human behavior prediction	13
1.4.2 RNN-based Trajectory Forecasting Network	13
1.4.3 Group analysis in crowds	14
1.4.4 Obstacle avoidance	15
1.4.5 A baseline approach: the Social LSTM	15
1.5 Group-LSTM	17
1.5.1 Pedestrian trajectory clustering	17
1.5.2 Group trajectory prediction	17
1.6 Obstacle-LSTM	19
1.6.1 Obstacle Sampling	19
1.6.2 Obstacle Tensor	21
1.6.3 Pedestrian trajectory prediction with obstacles	23
1.7 Group-GAN	24
1.7.1 Overall model	24
1.7.2 LSTM-based Generative Adversarial Networks	24
1.7.3 Group Pooling	26
1.7.4 Attention Mechanism	27
1.8 Implementation details	29
1.8.1 Group and Obstacle LSTM	29
1.8.2 Group GAN	30
1.9 Results	30
1.9.1 Quantitative Results	30

1.9.2	Qualitative Results	35
1.10	Conclusions and future work	39
2	Crowd Simulation	40
2.1	Introduction	40
2.1.1	Properties of a good crowd simulator	40
2.1.2	The advantages of a crowd simulator	41
2.1.3	Simulation-driven analysis	42
2.1.4	From real to synthetic crowds: data-driven crowd simulation	42
2.2	Data-driven crowd simulation	43
2.2.1	Introduction	43
2.2.2	Related work	44
2.2.3	Methods	45
2.2.4	Experimental results	48
2.2.5	Future work	49
2.3	Virtual Crowds	51
2.3.1	Introduction	51
2.3.2	Related work	52
2.3.3	The proposed model	53
2.3.4	Implementation details	56
2.3.5	Results	57
2.3.6	Future work	58
2.4	Virtual camera modeling for multi-view simulation of surveillance scenes	58
2.4.1	Introduction	60
2.4.2	Camera model description	61
2.4.3	Pan-Tilt-Zoom Camera	65
2.4.4	Results	66
2.4.5	Future work	67
2.5	Conclusions	67
2.5.1	Open Issues	70
3	Crowd Simulation Application: Surveillance	72
3.1	Introduction	72
3.2	Related Work	73
3.3	Dynamic Camera Network Reconfiguration	74
3.3.1	Observation Model	74
3.3.2	Camera Models	76
3.3.3	Reconfiguration Objective	78
3.3.4	Update Function	78
3.3.5	Local Camera Decision	79
3.3.6	Evaluation Metrics	79
3.4	Experimental Results	80
3.4.1	Quantitative Results	80
3.4.2	Qualitative Results	81
3.5	Conclusions	83

4	Toward anomaly detection in the wild: out of distribution detection	84
4.1	Introduction	85
4.2	Problem Statement and Related Work	86
4.3	Union of 1-dimensional subspaces as the in-distribution structure . .	87
	4.3.1 Enforcing the Structure	90
4.4	Out-of-distribution Detection Test	92
4.5	Experiments	95
4.6	Conclusions	101
5	Conclusions and Future Directions	102
	Bibliografia	104
	List of Publications	118

Introduction

Context

Crowds of people are gathering at multiple venues, such as concerts, political rallies, as well as in commercial malls, or just simply walking on the streets. More and more people are flocking to live in urban areas, thus generating a lot of scenarios of crowds. As a consequence, there is an increasing demand for automatic tools that can analyze and predict the behavior of crowds to ensure safety.

Crowd motion analysis is a key feature in surveillance and monitoring applications, providing useful hints about potential threats to safety and security in urban and public spaces. It is well known that people gatherings are generally difficult to model, due to the diversity of the agents composing the crowd. Each individual is unique, being driven not only by the destination but also by personality traits and attitude.

The domain of crowd analysis has been widely investigated in the literature [39]. However, crowd gatherings have sometimes resulted in dangerous scenarios in recent years, such as stampedes or during dangerous situations as shown in Fig. 1.

To take a step toward ensuring the safety of crowds, in this work we investigate two main research problems: we try to predict each person future position and we try to understand which are the key factors for simulating crowds. Predicting in advance how a mass of people will fare in a given space would help in ensuring the safety of public gatherings.



Figure 1: Examples of crowded situations: (a) Recent rallies of the Sardine movement in Italy (*credits: Ansa*); (b) People gathering in a mall for the Black Friday (*credits: Repubblica*); (c) A snapshot from the terrorist's attack during the Boston marathon in 2013 (*credits: Reuters*)

Problems and solutions

This thesis focuses the attention on two specific problems related to the crowd analysis: trajectory prediction and crowd simulation. It also shows possible applications of

crowd simulation for surveillance. Finally, we describe a possible approach to use neural networks for anomaly detection.

Trajectory prediction in crowded scenarios

Understanding the future state of a crowd of people is key for multiple applications like security and retail planning. The prediction of the future state of a crowd can be carried on at two different levels:

- *global prediction*, where a crowd is seen and its movements are forecast as a unique entity
- *local prediction*, where we focus on predicting the individual trajectory and future position of a single pedestrian in the crowd

In recent years, the research focused on predicting each individual trajectory because it allows for better analysis and it is more applicable for different sparsity of the crowd.



Figure 2: Example of a pedestrian trajectory prediction problem in Grand Central Station, New York. In green, we highlight the trajectory that we have observed so far. In red, some of the possible future paths of the pedestrian of interest. Which path will he take among all the available ones? In blue, groups of socially related pedestrians are highlighted. How different is the interaction between socially related and unrelated pedestrians walking in a crowded scene?

In the literature [117, 3], solving the trajectory prediction task means determining which will be the future positions of a pedestrian in the next time frame, given the

observation of the previous time frame. This observation describes the pedestrian’s previous positions as well as its relative position with respect to other pedestrians and the presence of obstacles in the environment. However existing approaches, exploit neither the existing social relationships (e.g. couples walking together) between pedestrians nor the interactions with obstacles in the environment to improve the prediction task.

To improve the prediction of pedestrian’s trajectory on a *local level*, we model and exploit the social relationship between pedestrians and the interactions between each pedestrian and the environment. We propose two approaches:

- **a Group-LSTM [24] and an Obstacle-LSTM**, which are based on Long Short Term Memory (LSTM) networks. On the one hand, the Group LSTM explicitly models the different interactions between socially related and unrelated pedestrians. On the other hand, the Obstacle LSTM focuses on modeling the interaction between each pedestrian and the environment. When combined, Obstacle and Group LSTM show an improvement in prediction accuracy and they provide a realistic model in which each pedestrian navigates an environment depending on the status of obstacles and other pedestrians in its surroundings.
- **a Group-GAN**. This approach is based on Generative Adversarial Networks (GAN). It exploits both social relationships and it uses an attention module to focus on the most relevant parts of observed trajectories to improve the accuracy of the prediction.

Crowd simulation

In the domain of crowd analysis, there is a lack of a common framework for testing, evaluating and comparing algorithms. This is due to the fact that annotating and collecting crowded scene’s footages present several issues such as privacy of the recorded subjects, videos’ quality, cameras’ positions, and the annotation of the ground truth, which has to be done manually. A realistic simulator of crowds must provide the following features:

- *Realistic high-level behavior*. The simulation should be able to consistently reproduce the behavior of the crowd in a variety of situations and environments, like in scenes populated with different densities or in emergency situations.
- *Realistic, smooth and adaptable motion*. Agents in the simulation should be able to fluently move and change the motion behavior, like going from walking to running or to stopping, depending on the situation.
- *Appearance modeling*. The simulated crowd should be rendered in a photorealistic way, such that algorithms applied to real and synthetic images would produce consistent results.

About *modeling the realistic high-level behavior*, previous works [64, 146, 147, 148] usually focused on managing the crowd either on a micro (single pedestrian in a small neighborhood) or on a macro scale (the crowd of people). Moreover, these models relied on empirically defined functions to define the movement of pedestrians. To improve these approaches we propose:

- a **data-driven crowd simulation** approach [22], which combines an empirical model for local pedestrian simulation and it learns general tendencies over time to reproduce the macro scale behavior of a crowd. Combining these two approaches allows us to better represent the crowd both on a micro and on a macro level.
- **Virtual Crowds: an LSTM-based framework for crowd simulation** [23], which relies on an LSTM network to model pedestrian movement instead of an empirically defined function. Using neural networks to model pedestrian behavior and tendencies would allow for a better-refined problem without hard-coded rules.

Appearance modeling of the output of the simulator is one of the key components which would allow using synthetic images for real-world applications. A recent trend in research is to leverage advanced simulation frameworks for the implementation and validation of video surveillance and ambient intelligence algorithms. However, in order to guarantee a seamless transferability between the virtual and real worlds, the simulator is required to represent the real-world target scenario in the best way possible. To tackle this issue, we propose

- **Virtual camera modeling for multi-view simulation of surveillance scenes**[20]. A solution for the problem related to camera modeling and control, which discusses how noise and distortions can be handled, and implements an engine for camera motion control in terms of pan, tilt, and zoom, with particular attention to the video surveillance scenario.

Crowd Simulation Applications

Simulation environments carry a few advantages with respect to real scenes such as:

- *replicability of the scene*: the same experiments with different camera configurations can be run multiple times in a controlled environment.
- *low cost of scene annotation*: the ground truth of images does not need to be manually annotated, but it is provided natively by the 3D engine instead.
- *no privacy issues*, which are a key issue when it comes to using and sharing dataset with footages of real pedestrians

We leverage on the features of our simulation environment to tackle the problem of crowd surveillance. When dealing with crowded scenes, a reconfigurable camera system can be deployed to provide a better understanding of the scene to prevent dangerous scenarios. The goal of such a system is to maximize the scene coverage while focusing on the most crowded areas. Such camera networks aim to focus the attention on critical areas of the crowd while ensuring an acceptable level of attention also on less critical areas. About crowd surveillance, we propose

- **Dynamic Camera Network Reconfiguration for Crowd Surveillance** [21] which uses a novel network control approach to explore the trade-off between

target resolution and coverage in heterogeneous networks consisting of fixed, PTZ (Pan Tilt and Zoom cameras), and UAV-based cameras. In our approach, we model the scene and the camera network in a *simulated environment*, where we estimate the state of the crowd by merging the contributions of the individual cameras and we let cameras locally decide on their next position parameters. Leveraging on the simulator, the replicability of crowd movements allows to test different camera reconfiguration policies to achieve the best coverage.

Toward anomaly detection in the wild: out of distribution detection

One of the most difficult challenges that arise when dealing with trajectory predictions as well as with simulation is the presence of anomalies. According to the Oxford Dictionary, an anomaly is defined as *"something that deviates from what is standard, normal, or expected"*. Due to their nature, anomalies occur rarely and are thus difficult to predict or to be captured in a model. Empirically defined models and especially neural networks rely on examples to learn a pattern, but anomalies are rare events thus not providing enough samples for common algorithms.

Such kind of problem can be present in crowded scenarios (e.g. identify an anomaly in a crowd, like an unattended object or a person running), as well as in real-world deep learning applications, such as autonomous driving or object identification, the trained network will often encounter *out-of-distribution* (OOD) samples.

OOD has been a focus of research in the image classification domain [13, 101]. The research problem focuses on how to detect and identify samples at test time, which have never been seen by the network at training time as shown in Fig. 3.

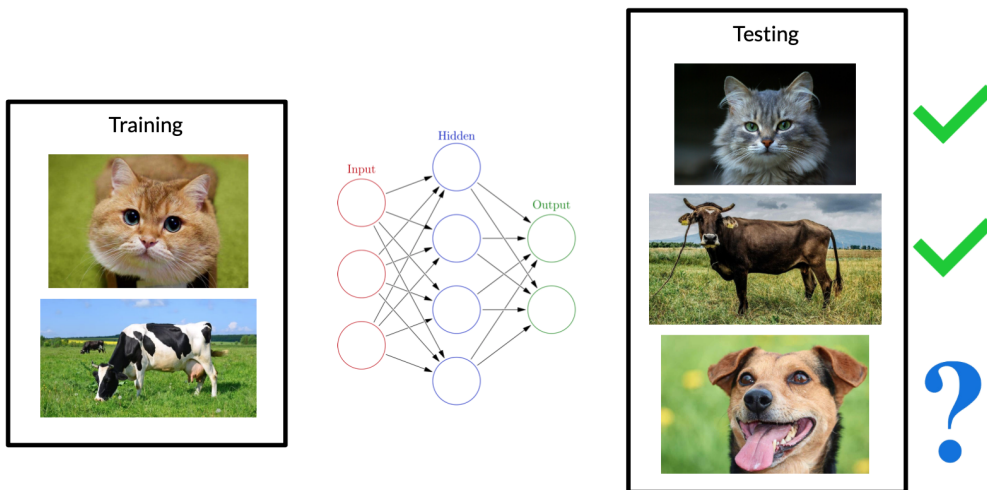


Figure 3: Example of an Out of Distribution scenario in the wild: a network has been trained to classify images of cats and cows. What will happen when an image of a dog is presented at test time? Common classification networks will classify the 'dog' image either as 'cow' or 'cat'. Our goal is to enable the network to understand that the 'dog' image is something that has never been seen before.

In this work, we propose

- **Detecting Out-of-Distribution (OOD) via Spectral Structure**, which is a hyperparameter-free framework that can efficiently detect OOD samples, without having access to them during training or validation. Our network relies on an enforced structure in the latent space which aims at maximizing the angular distance between known (in distribution) and unknown (out of distribution) samples.

Structure of the thesis

The rest of the dissertation is organised as follows:

- Chapter 1 is devoted to the trajectory prediction problem. we show our proposed solution along with conducted experiments, the results achieved, and their description and comparisons against the state-of-the-art
- Chapter 2 introduce the the simulation of crowded scenario. We present our point of view on the applications of simulation for crowded scenarios with advantages and drawbacks. We then present our purposed approach for modeling the high-level behavior and the appearance of the crowded scenarios.
- Chapter 3 shows a possible use case of our simulation environment in a crowd surveillance scenario.
- Chapter 4 discusses anomalies detection in crowded scenarios. We show how the challenge can be viewed as an out of distribution problem and we describe our approach for detecting unknown samples.
- Chapter 5 draws the conclusions of this research work and discusses possible future directions.

1 Trajectory Prediction in Crowded Scenarios

1.1 Introduction

Crowd motion analysis is a key feature in surveillance and monitoring applications, providing useful hints about potential threats to safety and security in urban and public spaces. It is well known that crowds are generally difficult to model, due to the diversity of the agents composing the crowd. Each individual is unique, being driven not only by the destination but also by personality traits and attitude. In fact, when choosing their paths, pedestrians are influenced by multiple social and environmental factors, which include, for example, the distance from neighbors, the local crowd density, the rules regulating specific environments (e.g. crossroads, streetlights), as well as social rules regulating public indoor or outdoor spaces (e.g., standing in line at a ticket counter), different relationships with pedestrians they are walking with (e.g., couples, friends).

Trajectory prediction in crowded scenarios is a key component of a modern surveillance system [160, 93, 4, 73]. In crowded scenes, forecasting both short and long term scenarios allows for better planning in many contexts, such as security, retail and crowd management. Existing approaches predict each individual trajectory, without explicitly modeling both the environment and interactions with other pedestrians. Interactions with other pedestrians are treated as similar, without differentiation between socially related people (e.g. couples, families) and people who do not know each other.

The literature has demonstrated that some general principles can be defined, to model the human motion within a social context. In particular, the Social Force Model (SFM) [64], the Reciprocal Velocity Obstacle [148], and the more recent Social-LSTM [3], exploit the presence and velocity of subjects in the neighborhood to improve the trajectory prediction of each pedestrian. These approaches, however, neither exploit the existing social relationships (e.g. couples walking together) between pedestrians, nor the interactions with obstacles in the environment. We show how social relationships can be exploited to improve the prediction. Moreover, we extend our model with the study of obstacles influences and we demonstrate how combining obstacles and social relationship features can further improve the prediction task.

Differently from crowd behavior recognition, the prediction task has some distinguishing characteristics, which are generally addressed by observing the motion histories of the subjects moving in the scene. In some specific applications (i.e., early warning, abnormal event detection, collision avoidance), prediction plays a more relevant role as compared to activity recognition, since potentially dangerous behaviors should be warned in advance.

Deep Neural Networks have allowed considerable improvements in most Computer Vision areas, including image classification [126], image segmentation [61], image captioning [6], and image generation [169]. They have also made it through the domain of pedestrian trajectories prediction demonstrating outstanding performances [3, 150, 86], in particular when using Recurrent Neural Networks (RNN) and Long-Short Term Memory (LSTM) cells. While traditional methods can merely make one-step forecasting (e.g., Kalman filter, particle filters, Markov chains), deep learning has shown the capability of enabling longer-term prediction.

When navigating in a crowded space, pedestrians tend to follow non-linear trajectories to avoid collisions with other pedestrians and obstacles. This non-linearity makes it difficult to predict their trajectories.

In fact, although each agent is attracted by its destination, it is worth noting that the scene evolution over time plays an important role in driving human behaviors, through macroscopic features like, for example, people exhibiting coherent motion patterns when moving in groups. Such coarse-level information may help to characterize the populated areas, providing useful details in terms of the direction of motion and other flow characteristics, motivating the need to also include the group activities in the motion model.

It is known in the literature that people moving in the crowd tend to follow a series of implicit social rules [129]. For instance, individuals tend to speed up or slow down their paces in order to avoid collisions; people prefer to preserve personal space, thus keeping a certain distance from their neighbors.

Although existing methods have achieved very good results, they still suffer from three main limitations:

- **Interactions between socially-related pedestrians.** Human-to-human interactions play a key role in determining what will be the future states of a given pedestrian. These interactions must be taken into account when predicting future paths. Pioneering work proposes several pooling modules for LSTM to capture global interactions [58], i.e., interactions with all other people in the scenario, or capture local interactions [3], i.e., interactions with only nearby pedestrians. However, these approaches do not exploit the different behavioral cues between socially-related and socially-unrelated pedestrians; as an example, people walking together in a group display a different behavior with respect to people who do not know each other [24, 111].
- **Interactions between obstacles and pedestrians.** We notice that pedestrians navigating in the environment are highly influenced by the presence/absence of obstacles in the scene. Existing models, like the SFM [64], provide repulsive forces with respect to obstacles to model the avoidance behavior. To this aim, we exploit obstacle annotation to refine and improve the results of the prediction task.
- **Attentive exploitation of past trajectories.** State-of-the-art models [58, 133] utilize the information of history paths by applying an LSTM-based encoder to obtain the hidden states of all the observed pedestrians, then pass them to an LSTM-based decoder to compute the prediction for all the observed

trajectories. Nevertheless, the analysis does not consider the relevance of each portion of the past trajectories, like the presence of sharp turns.

- **Overall evaluation metrics for predicted paths.** Most every trajectory prediction framework followed the convention of adopting the Average Displacement Error (ADE) in [87], used as the quantitative evaluation metric. The assessment by a point-to-point displacement error may not be comprehensive; other elements like trajectory similarity or collision rate are also worth being considered.

1.2 Contribution

In our opinion, to further enhance the results of the prediction of future paths and behavior of people, we must focus on modeling not only the relationships between vehicles and pedestrians but also the relations between pedestrians themselves. For this purpose we propose two novel approaches:

- a **Group LSTM** [24] which explicitly model pedestrian relationship. We detect wherever they belong to the same social group or not based on their motion coherency. The prediction is performed in a one-to-many manner, so each individual trajectory is predicted with respect to the current relationship of the pedestrian of interest with others. Moreover we also explicitly model the relationship between the pedestrian and fixed obstacles in the environment (**Obstacle LSTM**), which allows us to further improve the prediction task.
- a **Group GAN** which uses a similar module to exploit relationships between pedestrians moving coherently. Moreover, we employ an *attention* module that allows focusing on the most important part of each observed trajectories (e.g. a sudden turn or a stop) to improve the prediction task.
- **Dynamic Time Warping (DTW)** [18] is introduced in the trajectory forecasting model to evaluate the similarity of two predictions. Together with the collision rate and the common ADE/FDE measurements, we believe this leads to a more comprehensive assessment of different trajectory prediction models.

1.3 Problem Statement

Predicting pedestrian trajectories implies the observation of the past motion history and interactions of pedestrians in the scene. The position p_i^t of pedestrian i at time t can be expressed in terms of his coordinates $p_i^t = (x_t^i, y_t^i)$. The trajectory P^i of pedestrian i can be defined by the time sequence of past, present, and future positions:

$$P^i = \{p_0^i, p_1^i, \dots, p_{t_{obs}-1}^i, p_{t_{obs}}^i, p_{t_{obs}+1}^i, \dots, p_{t_{pred}}^i\} \quad (1.1)$$

Let N denote the number of pedestrians in the scene, then the total trajectory set can be presented as:

$$S^N = \{P^1, P^2, \dots, P^N\} \quad (1.2)$$

In our forecasting problem, we observe trajectories of pedestrians S_{obs}^N from time 0 to t_{obs} and we predict future trajectories S_{pred}^N for a certain time interval, namely from t_{obs+1} to t_{pred} .

1.4 Related Work

Detailed literature on the recent works in crowd analysis, especially regarding the topics of crowd dynamics modeling, social activity forecasting, and group segmentation, can be found in some recent surveys [90, 55, 78]. In the next paragraphs, we will concentrate on behavior forecasting, group analysis and interactions between fixed obstacles.

1.4.1 Human behavior prediction

Forecasting social activities has lately gained significant attention, especially in the domain of crowd analysis. As far as the interaction modeling is concerned, Helbing et al. [64] propose the well known SFM, which relies on Newtonian forces to model the interactions between pedestrians, and guide each agent towards his goal. Other models, such as the continuum crowds model [146], are capable to reproduce human interactions using priors. The crowd is modeled as a fluid, and agents are influenced by their goal, position, preferred speed, and a discomfort factor. In [4], the Social Affinity Maps (SAM) features and the Origin and Destination (OD) priors are proposed to forecast pedestrians destinations using multi-view surveillance cameras. In robotics, the Reciprocal Velocity Obstacle (RVO) [147] is widely used to model the collision avoidance between agents.

The models mentioned above require though the estimation or computation of parameters, in order to accurately predict the interaction among agents. The literature has tackled this issue relying on common optimization methods [154], like the Gradient-based Newton method [135] and Genetic Algorithms [117]. Crowd modeling has also been developed using discrete choice models [7] and Gaussian functions [141]. Groups in crowds, either stationary or moving, have also been a subject of study, usually as an extension of already developed models [111]. In [12, 104], contextual information is taken into account as well, to model the static configuration and the dynamic evolution of the scene.

All these methods are able to describe crowd behaviors assuming the availability of a strong prior associated with the model. They are based on energy potentials, relative distances, and hand-crafted rules.

1.4.2 RNN-based Trajectory Forecasting Network

From machine translation [37, 34] to speech recognition [35, 57], Recurrent Neural Networks (RNN), including Long Short Term Memory (LSTM) [66] and Gated Recurrent Unit (GRU) [36] have achieved good results in sequence prediction tasks. Trajectories can be viewed as 2D time-series with no seasonality, which facilitates the adoption of RNN-based approaches. Recently lots of RNN-based trajectory prediction networks have been proposed thanks to their ability to effectively learn

long-term dependency in sequences [3, 58, 133, 85, 134].

Jain et al. [71] adopt a structural RNN that combines spatio-temporal graphs and recurrent neural networks to model motion and interactions in the scene. Fernando et al. [149] apply both the soft attention and the hard-wired attention on the Social-LSTM, significantly promoting the trajectory prediction performance. Varshneya et al. [149] present a soft attention mechanism to forecast an individual’s paths, exploiting the spatially-aware deep attention model. Vemula et al. [150] propose a novel social attention model that can capture the relative importance of each person when navigating in the scene. Alahi et al. [3] propose the Social-LSTM to model the interactions among people in a neighborhood by adding a new social pooling layer; in [86], Lee et al. present a deep stochastic IOC RNN encoder-decoder framework to predict the future paths of multiple interacting agents in dynamic scenes. Hug et al. [68] present an experiment-based study to evaluate the effectiveness of some RNN models in the context of socially-aware trajectory prediction. Lee et al. [85] introduce an RNN Encoder-Decoder framework, which uses Variational AutoEncoder (VAE) for trajectory prediction. Ballan et al. [11] consider both the dynamics of moving agents and the scene semantics to predict scene-specific motion patterns. In [58], generative adversarial neural networks (GANs) are employed to predict socially-plausible trajectories in a crowded environment.

Recent approaches have focused on improving the modeling of social interaction via the application of adversarial networks and attention mechanisms. Gupta et al. [58] applied generative setting on the LSTM network to model many-to-many interaction. Sadeghian et al. [133] generated socially-plausible trajectories by adopting a soft attention mechanism [155] on social and physical constraints.

1.4.3 Group analysis in crowds

Social interactions among pedestrians play a crucial role in determining their future trajectory. By clustering trajectories with similar motion trends, pedestrians can be segmented into groups. In [167], traditional k-means clustering is exploited to learn different motion modalities in the scene. In [83], support vector clustering is adopted to define groups among pedestrians. In [168], coherent filtering is used to detect coherent motion patterns in a crowded environment [156].

As far as the representation of collective activities is concerned, Ge et al. [52] work on the automatic detection of small individual groups traveling together. Ryoo et al. [132] introduce a probabilistic representation of group activities, for the purpose of recognizing different types of high-level group behaviors. Yi et al. [159] investigate the interactions between stationary crowd groups and pedestrians to analyze pedestrian behaviors, including walking path prediction, destination prediction, personality classification, and abnormal event detection. In [24], the authors detect and exploit pedestrians moving coherently to improve the prediction task. Shao et al. [138] propose a series of scene-independent descriptors to quantitatively describe group properties, such as collectiveness, stability, uniformity, and conflict. Bagautdinov et al. [9] present a unified end-to-end framework for multi-person action localization and collective activity recognition using deep recurrent networks. Moussad et al. [111] model the reciprocal interactions between socially-related and socially-unrelated pedestrians.

1.4.4 Obstacle avoidance

In the literature obstacles are generally bounded to moving agents through forces [64] [115], or as boundaries to a fluid [146].

Data-driven approaches have recently been employed to capture and model interactions among people in a crowd and obstacles. The overall idea consists of trying to learn and reproduce either local or global features related to the crowd collective behavior. Vector fields are applied in multiple works to learn the velocity and navigation features of real scenes. Patil et al. [114] use vector fields, either learned or manually sketched, to guide the crowd flow in the environment. Goal-dependent velocity fields are also used to guide the simulation at a global level [22]. Hu et al.[67] propose a solution to deal with the global and local crowd flow, as a feature to be transferred from real to simulated scenes, learning crowd motion patterns on the basis of the instantaneous motion field.

1.4.5 A baseline approach: the Social LSTM

Modeling human behaviors in crowded scenarios has been addressed in the literature mostly using empirically-defined functions [64, 147, 146].

Parametric functions are generally good in reproducing the global motion properties of the crowd, but they tend to fail when capturing the personality that leads each human to react differently when facing the presence of other subjects or obstacles.

In the domain of path prediction, LSTM networks [66] have shown good capabilities in predicting the behavior of pedestrians. The Social-LSTM model [3] is able to capture the status of the neighbourhood of each agent (namely, the number of agents in the surroundings and the corresponding positions) to refine the trajectory prediction. The state of the neighbourhood of each pedestrian is represented by a *social* hidden-state tensor, as proposed by [3]. The social pooling layer allows pedestrians to share their hidden states, thus enabling each network to predict the next position of the agent by reasoning about its hidden state and the neighbourhood state.

The j -th pedestrian, referred to as ped^j at time t in the scene, is represented by the hidden-state h_t^j of a LSTM network. Let the hidden-state dimension be D , and the neighbourhood size be N_0 .

The neighbourhood of the agent ped^i is described by the social pooling layer defined as the tensor H_t^i , with dimensions $N_0 \times N_0 \times D$:

$$H_t^i(m, n, :) = \sum_{j \in N_i} 1_{mn}[x_t^j - x_t^i, y_t^j - y_t^i] h_{t-1}^j \quad (1.3)$$

where h_{t-1}^j represents the hidden-state of the LSTM for ped^j ($\forall j \neq i$) at $t - 1$, N_i represents the set of neighbours of pedestrian ped^i , and $1_{mn}[x, y]$ is an indicator of presence at location (m, n) defined as:

$$1_{mn}[x, y] = \begin{cases} 0 & \text{if } [x, y] \notin \text{cell } mn \\ 1 & \text{if } [x, y] \in \text{cell } mn \end{cases} \quad (1.4)$$

A graphical representation of the pooling operation is shown in Fig. 1.3. Once computed, the social hidden-state tensor is embedded into a vector a_t^i . The output

coordinates are embedded in vector e_t^i .

The resulting recurrence is then defined by the following equations:

$$e_t^i = \Phi(x_t^i, y_t^i; W_r) \quad (1.5)$$

$$a_t^i = \Phi(H_t^i; W_e) \quad (1.6)$$

$$h_t^i = LSTM(h_{t-1}^i, e_t^i, a_t^i; W_l) \quad (1.7)$$

where Φ is a ReLU (Rectified Linear Unit) embedding function, W_r and W_e represent the embedding weights, and W_l represents the LSTM weights.

The next position (x_{t+1}^i, y_{t+1}^i) in the prediction depends on the hidden-state at the previous time-step h_t^i . Inspired by [56], and as performed in [3], the following parameters are predicted, which characterize a bi-variate Gaussian distribution: the mean $\mu_{t+1}^i = (\mu_x, \mu_y)_{t+1}^i$, the standard deviation $\sigma_{t+1}^i = (\sigma_x, \sigma_y)_{t+1}^i$ and the correlation coefficient ρ_{t+1}^i . The original model uses a $5 \times D$ weight matrix W_p to estimate the parameters. Thus, the coordinates at the next time-step $t + 1$ are computed as:

$$(x_{t+1}^i, y_{t+1}^i) \sim N(\mu_t^i, \sigma_t^i, \rho_t^i) \quad (1.8)$$

In order to estimate the parameters of the LSTM model, the negative log-likelihood loss L^i for agent ped^i is minimized for the current time t :

$$[\mu_t^i, \sigma_t^i, \rho_t^i] = W_p h_{t-1}^i \quad (1.9)$$

$$L^i(W_e, W_l, W_p) = - \sum_{t=T_{cur}+1}^{T_{step}} \log(\mathbb{P}(x_t^i, y_t^i | \mu_t^i, \sigma_t^i, \rho_t^i)) \quad (1.10)$$

The model is trained minimizing the log-likelihood loss for all the trajectories belonging to the dataset.

Pedestrians moving in a crowded scene adapt their motion according to the motion and position of other pedestrians and the position of obstacles in their neighbourhood. Thus, the prediction of the trajectory of an individual necessarily requires taking into consideration also the environment where the agent moves. In [3], the state of the neighbourhood is modeled considering the interaction of the pedestrian with the neighbours. Compared to the original formulation of the Social-LSTM (briefly recapped in Section 1.4.5), we propose the use of a more comprehensive model, called Group-LSTM, which includes the social relationships between pedestrians in the crowd, thus improving the prediction performance. Furthermore, we propose a system module, called Obstacle-LSTM, whose goal is to extend the description of the current state of the agent with the information related to fixed obstacles in the environment.

1.5 Group-LSTM

As mentioned in the previous paragraphs, the motion of pedestrians in crowded scenes is highly influenced by the behavior of other people in the surroundings and their mutual relationships. Stationary groups, groups of pedestrians walking together, people coming from opposite directions, will exert different effects on the action that one pedestrian takes. Thus, we propose a framework, which is able to consider whether the subject of interest is walking coherently with the pedestrians in his surroundings or not. By exploiting the coherent filtering approach [168], we first detect people moving coherently in a crowd, and then adopt the Social-LSTM to predict future trajectories. In this way, we are able to improve the prediction performance, accounting for the interactions between socially-related and unrelated pedestrians in the scene.

1.5.1 Pedestrian trajectory clustering

Coherent motion describes the collective movements of particles in a crowd. Coherent filtering [168] introduces a prior, meant to describe the coherent neighbour invariance, namely the local spatio-temporal relation between particles moving coherently. The algorithm is based on two steps. First, it detects the coherent motion of pedestrians in the scene. Then, points moving coherently are associated to the same cluster. Point clusters will continue to evolve, and new clusters will emerge over time. Eventually, each pedestrian i is assigned to a cluster s^i . The output of the coherent filtering consists of sets s^i ($i = 1, 2, \dots, n$) of people moving in a coherent manner.

The original implementation of the coherent filtering relies on the KLT tracker [144], aiming at detecting candidate points for tracking and generating trajectories, which are then used as the input of the algorithm. However, the KLT tracker may detect multiple key points for each pedestrian, thus there is no clear correspondence between the number of key points and the number of pedestrians. Our objective is to cluster pedestrians into groups, where each individual in a group is represented using a single point, as shown in Fig. 1.1. For this purpose, and without loss of generality, we apply the coherent filtering algorithm directly on the ground truth of pedestrians' trajectories.

1.5.2 Group trajectory prediction

In the Social-LSTM model, each pedestrian is modeled using a LSTM network as displayed in Fig. 1.2. Each pedestrian is then linked with the other people in his neighbourhood via a so-called social pooling layer. The social pooling layer allows pedestrians to share their hidden states, thus enabling each network to predict the future positions of an individual based on his own hidden state and the hidden states in the neighbourhood.

The i^{th} pedestrian at time t in the scene is represented by the hidden state h_t^i in an LSTM network. We set the hidden-state dimension to D and the neighbourhood size to N_0 , respectively. The neighbourhood of the i^{th} agent ped^i is described using a tensor H_t^i as in Eq. 1, with dimensions equal to $N_0 \times N_0 \times D$:

$$H_t^i(m, n, :) = \sum_{j \in N_i} 1_{mn}[x_t^j - x_t^i, y_t^j - y_t^i] G_{ij}[s^i \neq s^j] h_{t-1}^j \quad (1.11)$$



Figure 1.1: Each pedestrian is represented by a single keypoint. Pedestrians walking in the same direction are clustered into one group s^i . In this example, two sets of pedestrians going in opposite directions are identified.

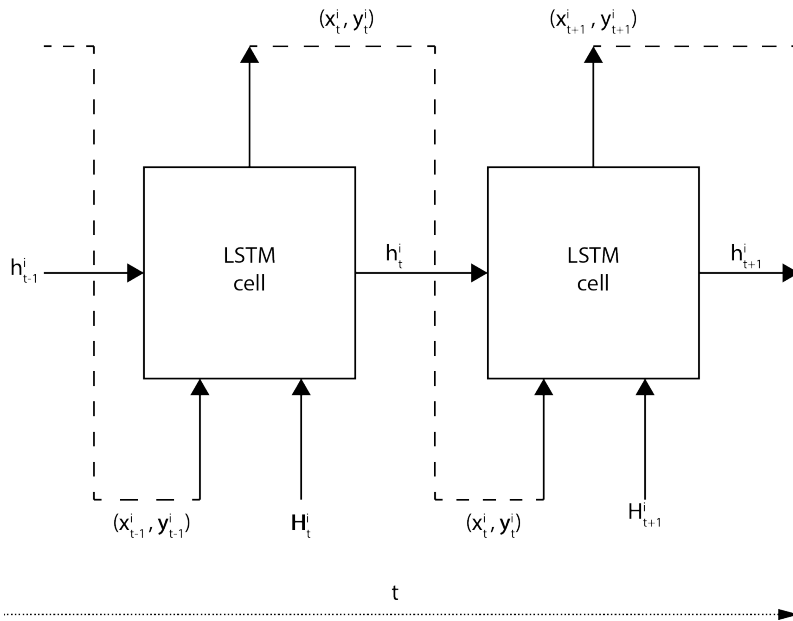


Figure 1.2: The figure represents the chain structure of the LSTM network between two consecutive time steps, t and $t + 1$. At each time step, the inputs of the LSTM cell are the previous position (x_{t-1}^i, y_{t-1}^i) and the Social pooling tensor H_t^i . The output of the LSTM cell is the current position (x_t^i, y_t^i) .

where $1_{mn}[x, y]$ is an indicator function to select pedestrians in the neighbourhood defined as in Eq. 1.4.

If two pedestrians i and j belong to the same coherent set s^i , they will not be considered when computing the social pooling layer for each of them. The function $G_{ij}[i \in s^i, j \in s^i]$ is an indicator function defined as in Eq. 1.12:

$$G_{ij}[s^i \neq s^j] = \begin{cases} 0 & \text{if } i \in s^i, j \in s^i \\ 1 & \text{if } i \in s^i, j \notin s^i \end{cases} \quad (1.12)$$

Doing so, the social pooling layer of each pedestrian contains information only about pedestrians that are not moving coherently with him. The intuition behind this operation is that the network usually learns the repulsive forces between pedestrians avoiding collisions. If pedestrians are socially-related, they will tend to stay closer to each other, thus the repulsive forces are much smaller (almost neglected) and they have a smaller influence on the future trajectory with respect to pedestrians not moving coherently. If socially-related pedestrians were considered in the social pooling, this would cause social groups to spread around instead of staying closer together as shown in Sec. 3.4.

Once computed, the social hidden-state tensor is embedded into a vector a_t^i . The output coordinates are embedded in the vector e_t^i . Following the recurrence defined in Sec. 1.4.5, we can predict our trajectories gradually.

1.6 Obstacle-LSTM

Fixed obstacles in an environment have a non-negligible impact in shaping and determining the flow of a crowd. Depending on the nature of the obstacles, pedestrians actuate different strategies to prevent collisions, taking subjective decisions. We want to embed this piece of information in the proposed model, in order to consider the presence of an obstacle in the agent’s neighbourhood. Since automatic obstacle detection is out of the scope of this paper, in the present work we manually annotate fixed obstacles in the scene, being a one-time operation. We exploit this additional annotation to improve the prediction performance by taking into account the interactions between pedestrians and obstacles in the scene.

1.6.1 Obstacle Sampling

To embed the scene layout into the network, we label physical obstacles in the scene in two main classes:

- *obstacle*, which has to be avoided, and that the pedestrian can not traverse (e.g., trees or light poles, walls);
- *semi-obstacle*, which is preferably avoided by a walking pedestrian (e.g., the flower bed around a tree, a meadow or a snowy part of the street).

Once these classes have been annotated, two different obstacle maps (one with obstacles only, and one with obstacles and semi-obstacles) are generated for each investigated scenario taken from [117] and [87]. Similarly to the process of trajectory annotation, obstacles are annotated on the image plane and their coordinates are

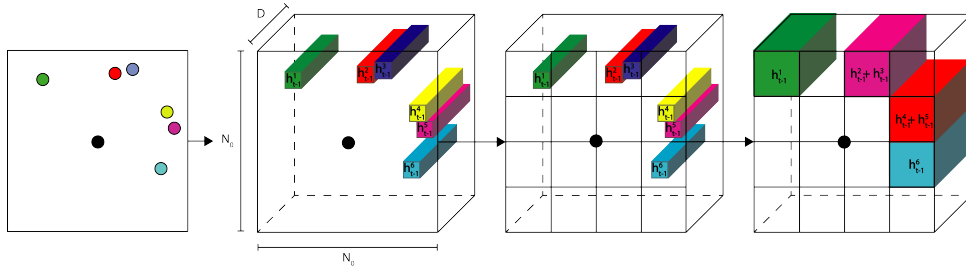


Figure 1.3: Representation of the Social hidden-state tensor H_t^i . The black dot represents the pedestrian of interest P^i . Other pedestrians P^j ($\forall j \neq i$) are shown in different color codes. The state of the neighbourhood of P^i is described by $N_0 \times N_0$ cells, which pooling together spatially-close neighbours preserves the spatial information.

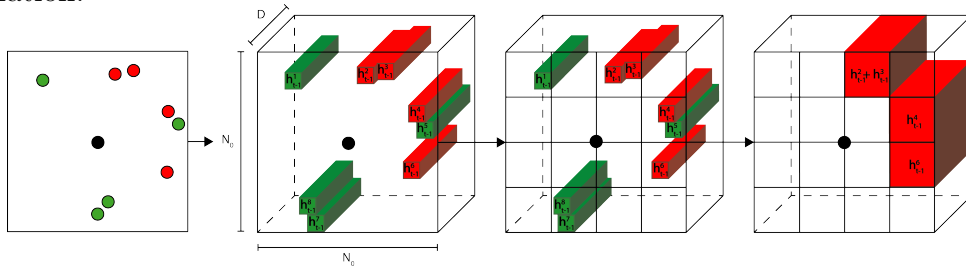


Figure 1.4: Other pedestrians P^j ($\forall j \neq i$) are shown in different color codes, namely green for pedestrians belonging to the same set, and red for pedestrians belonging to a different set. The neighbourhood of P^i is described by $N_0 \times N_0$ cells, which preserves the spatial information by pooling spatially adjacent neighbours. Pedestrians belonging to the same set are not used for the final computation of the pooling layer H_t^i .

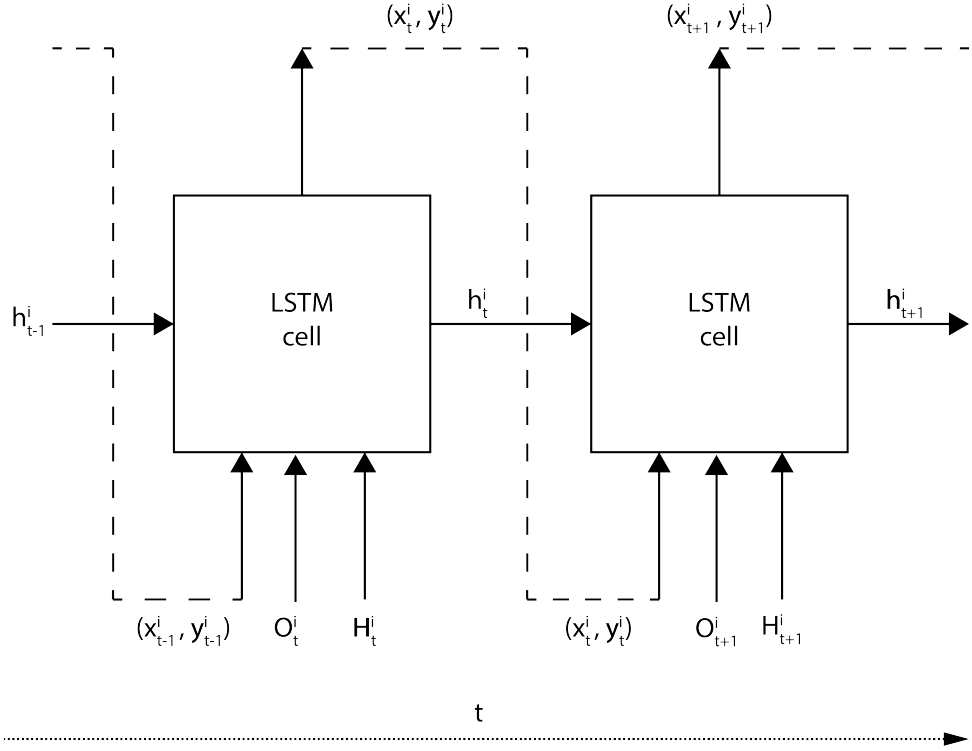


Figure 1.5: The figure represents the chain structure of the LSTM network between two consecutive time steps, t and $t + 1$. At each time step, the inputs of the LSTM cell are the previous position (x_{t-1}^i, y_{t-1}^i) , the Obstacle tensor O_t^i and the Social pooling tensor H_t^i . The output of the LSTM cell is the current position (x_t^i, y_t^i) .

then projected on the ground plane in meters using the available homography for each dataset.

Figure 1.6 reports the obstacle and semi-obstacle map of the 5 scenes of interest. From now, if not explicitly mentioned, we will use the term *obstacles* including both obstacles and semi-obstacles. After obtaining the obstacle maps, the corresponding coordinates have been annotated by equally sampling points on each external boundary area.

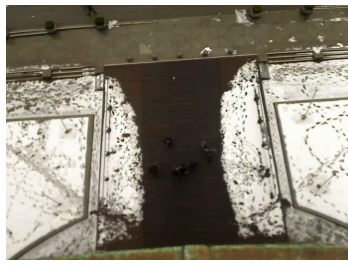
1.6.2 Obstacle Tensor

The obstacle pooling layer allows pedestrians to understand the state of obstacles in their neighbourhood, thus improving their knowledge about the surrounding environment.

The configuration of obstacles is represented by the Obstacle Tensor. We set the neighbourhood size to N_0 . The obstacles in the neighbourhood of the i^{th} agent ped^i are described using a tensor O_t^i , with dimensions $N_0 \times N_0 \times D$.

We define two versions of the obstacle tensor:

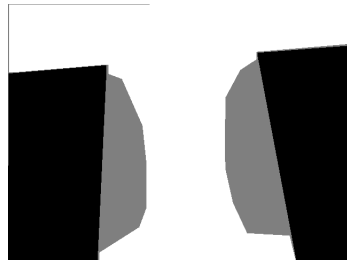
- the Obstacle Presence (OP) tensor, where a constant value indicates the presence or absence of an obstacle in a given cell;
- the Obstacle Distance (OD) tensor, where the distance between the obstacle



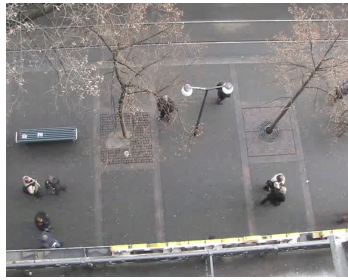
(a) Eth-Univ Scene



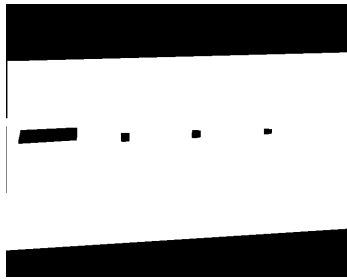
(b) Eth-Univ Obstacle



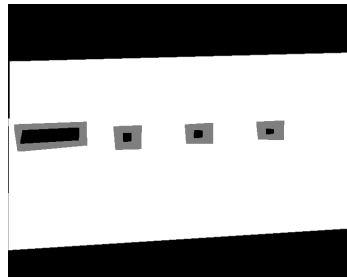
(c) Eth-Univ Semi-Obstacle



(d) Eth-Hotel Scene



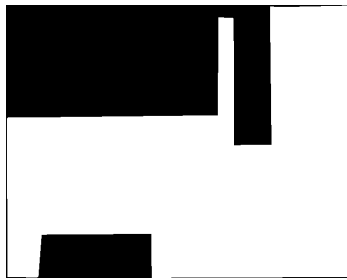
(e) Eth-Hotel Obstacle



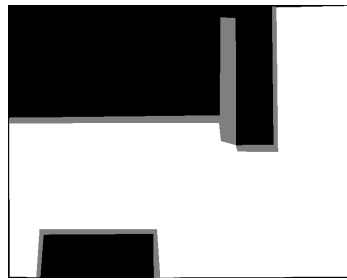
(f) Eth-Hotel Semi-Obstacle



(g) Zara01 Scene



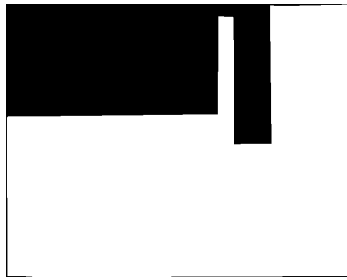
(h) Zara01 Obstacle



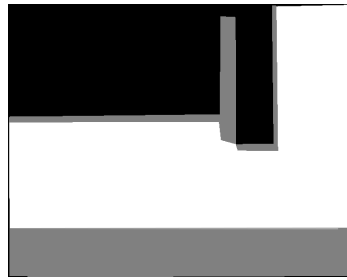
(i) Zara01 Semi-Obstacle



(j) Zara02 Scene



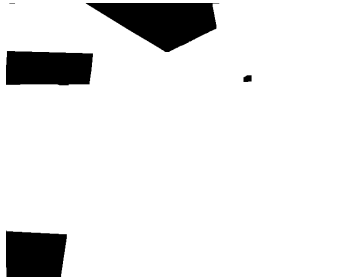
(k) Zara02 Obstacle



(l) Zara02 Semi-Obstacle



(m) Ucy-Univ Scene



(n) Ucy-Univ Obstacle



(o) Ucy-Univ Semi-Obstacle

Figure 1.6: Obstacle(black) and Semi-Obstacle(gray) classes.

and the pedestrian of interest indicates the presence or absence of an obstacle in a given cell.

The OP tensor for pedestrian i at time t is represented as:

$$OP_t^i(m, n) = \sum_{j \in N_o} 1_{mn}[x_t^i - x^j, y_t^i - y^j] \quad (1.13)$$

where 1_{mn} is the presence function at location (m, n) of the pedestrian i w.r.t. each obstacle j in the scene obstacle set N_o . The OP tensor allows the model to learn whether a cell (m, n) in the neighbourhood of the pedestrian i is occupied or not.

Instead of representing just the occupancy, the OD Tensor takes into account the Euclidean Norm $D(i, j)$, between the observed pedestrian i and the obstacle j , such that:

$$D(i, j) = \sqrt{(x_t^i - x^j)^2 + (y_t^i - y^j)^2}. \quad (1.14)$$

The OD_t^i tensor for pedestrian i at time t is represented as:

$$OD_t^i(m, n) = \sum_{j \in N_o} 1_{mn}[x_t^i - x^j, y_t^i - y^j]D(i, j) \quad (1.15)$$

where 1_{mn} is the presence function in position (m, n) of the pedestrian i w.r.t. each obstacle j in the scene obstacle set N_o and $D(i, j)$ is the Euclidean Norm between i and j .

1.6.3 Pedestrian trajectory prediction with obstacles

Like in [3], the Obstacle Tensor and the social hidden-state tensor are embedded into vectors b_t^i and a_t^i , respectively. The output coordinates are embedded in vector e_t^i .

The resulting recurrence is then defined by the following equations:

$$e_t^i = \Phi(x_t^i, y_t^i; W_r) \quad (1.16)$$

$$a_t^i = \Phi(H_t^i; W_e) \quad (1.17)$$

$$b_t^i = \Phi(O_t^i; W_o) \quad (1.18)$$

$$h_t^i = LSTM(h_{t-1}^i, e_t^i, a_t^i, b_t^i; W_l) \quad (1.19)$$

Thus, in the Obstacle-LSTM, the input tensor to the LSTM cell is composed by three embedded tensors with the same size (as shown in Fig. 1.5): the Social Tensor, the actual pedestrian position, and the Obstacle Tensor. At deploying time, we can have 4 different Obstacle-LSTM configuration, depending on whether we are using OP or OD, coupled with either obstacles or semi-obstacles maps.

1.7 Group-GAN

1.7.1 Overall model

Pedestrians walking in a crowd possess the innate ability to interact with others. Their chosen paths depend on past trajectories, and also take into account the motion of neighboring people. Past trajectories directly reveal future motion trend of the pedestrians. Depending on the relevance, certain portions of trajectory can be more informative than others. Moreover, the social relationship between pedestrians can be of great interest, as pedestrians who are socially-related tend to stay closer to each other and move coherently. Our Group-GAN aims at improving path prediction exploiting the motion history as well as the social relationship.

As shown in Fig. 1.7, Group-GAN consists of two key modules: the Generator G and the Discriminator D . The Generator G is based on an encoder-decoder framework, which contains the group pooling module and the attention mechanism. It takes as input the positions of pedestrians in the scene between time 0 and t_{obs} . Each pedestrian is modeled with an LSTM cell, which represents the hidden state. Hidden states pass through the group pooling and then the attention module. The group pooling module models the selective neighborhood interaction between socially-related and unrelated pedestrians. The attention module focuses on the most informative segments of past trajectories and improves the modeling of neighborhood interaction.

Given the trajectories generated by the encoder, the LSTM decoder of Generator G computes the hidden state of each pedestrian and it is able to generate socially-plausible future trajectories. The Discriminator D also uses an encoder to distinguish whether the trajectory is plausible or not.

1.7.2 LSTM-based Generative Adversarial Networks

Generative Adversarial Networks (GANs) proposed by Goodfellow et al. [54] offer a distinct and successful approach that focuses on a game-theoretic formulation for training a synthesis model. GANs are composed of a generator and a discriminator that are trained iteratively with competing goals. In particular, our generator is trained to generate a set of future trajectory predictions. The predicted set S_{pred}^N consists of N trajectories $P^i = \{p_{t_{obs}+1}^i \dots p_{t_{pred}}^i\}$. The discriminator is trained to minimize the distance between the set of trajectories generated and the ground truth.

Generator. The location of each pedestrian p_t^i is embedded by a linear layer to get a fixed-length representation e_t^i . The LSTM encoder takes the embedding vector e_t^i as input and attains the hidden states at time t by recurrence:

$$y_t^e, h_t^{ei} = LSTM^{en}(h_{t-1}^{ei}, e_t^i) \quad (1.20)$$

where y_t^e is the output of encoder, which will be utilized in the attention module. In order to capture human-to-human interaction in the neighborhood, the hidden states until time t_{obs} are pooled together in tensor T^i (of each person) in our group pooling module (PM). We define the context vector c_t^i for each pedestrian as:

$$c_t^i = MLP(T^i, h_t^{ei}) \quad (1.21)$$

where $MLP(\cdot)$ is a Multi-layer Perception with ReLU activation. Note that the context vector c_t^i in our model is only provided once to the decoder, which results in increased speed when compared against S-LSTM [3] and makes the deployment of our global attention mechanism feasible. The hidden states for the decoder are computed by concatenating context vector c_t^i and white noise vector z sampled from a multivariate normal distribution, in line with [58]. This initialization process can be denoted as:

$$h_t^{di} = [c_t^i, z] \quad (1.22)$$

Then the prediction path is obtained recurrently by the decoder as:

$$\begin{aligned} a_t &= \text{attn}(e_t^i, h_t^{di}, y_t^e) \\ T^i &= PM(h_{t-1}^{d1}, \dots, h_t^{di}) \\ h_t^{di} &= LSTM^{de}(MLP(T^i, h_t^{di}), a_t) \\ \hat{y}_t^i &= MLP(h_t^{di}) \end{aligned} \quad (1.23)$$

where e_t^i is the embedding representation of the relative position of pedestrian i at time t . Another Multi-Layer Perception is also used to obtain the predicted location \hat{y}_t^i .

Discriminator. Inspired by [58], the Discriminator D in our model is comprised of a LSTM encoder, the hidden pooling module proposed in [58], and the MLP classifier. According to Qi et al. [119], the hidden pooling module based on MLP and followed by a symmetric function, is able to capture the global social interaction context, i.e. the interaction between the pedestrian of interest and all the others in the scene. Different with group pooling in the Generator G , we adopt hidden pooling module [58] here, in order to improve the discrimination capability of D . Given the set S_{obs}^N of observed trajectories, we feed D with the ground truth set of N trajectories S_{gt}^N and a set of N predicted trajectories S_{pred}^N as in Eq. 1.24:

$$L_t^i = LSTM^{en}([S_{obs}^N, S_N], h_t^i) = \begin{cases} 1 & \text{if } S_N = S_{gt}^N \\ 0 & \text{if } S_N = S_{pred}^N \end{cases} \quad (1.24)$$

where L_t^i is the label of input trajectories, namely 1 (True) for a socially-acceptable trajectory and 0 (False) for a non-acceptable trajectory.

Losses. The adversarial loss is a standard training procedure of a GAN, in a two-player min-max game and it is expressed as follows:

$$\begin{aligned} \mathcal{L}_{GAN}(L_t^i, \hat{L}_t^i) &= \min_G \max_D E_{S_{gt}^N} [L_t^i \log(\hat{L}_t^i)] + \\ &E_{S_{pred}^N} [(1 - L_t^i) \log(1 - \hat{L}_t^i)] \end{aligned} \quad (1.25)$$

in line with the prior work [58, 133, 134], we also use L2 loss to train the GAN, as shown in Eq. 1.26. By using L2 loss, the GAN can intuitively learn to predict the future path for each agent. Furthermore, variety loss is also applied to encourage diverse generation. For each scene we generate k possible predictions by randomly sampling white noise z from multivariate normal distribution, and choosing the *best* generation in L2 sense, as our prediction.

$$\mathcal{L}_{L2}(S_{gt}^N, S_{pred}^N) = \min_k \|S_{gt}^N - S_{pred}^N\|^2 \quad (1.26)$$

According to the controlled experiments of trajectory prediction in [80], only combined adversarial and L2 losses are able to render realistic trajectories in a GAN framework. Hence, the loss function in our Group-GAN consists of the original GAN loss \mathcal{L}_{GAN} and the L2 loss \mathcal{L}_{L2} , which can be defined as:

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda \mathcal{L}_{L2} \quad (1.27)$$

where λ is a regularization weight of the L2 loss.

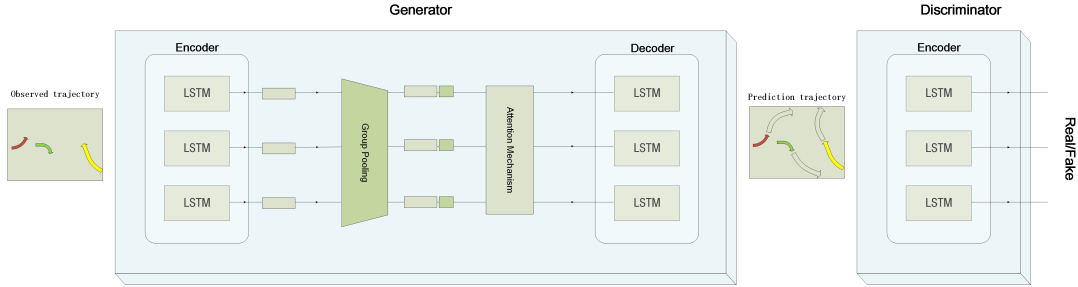


Figure 1.7: The architecture of the proposed Group-GAN. The Generator G takes past trajectories and encodes them into hidden states. The hidden states are pooled by a group pooling module with the information of selected neighborhood interaction. The attention mechanism helps the decoder focusing on relevant segments of the trajectories for future path generation. The Discriminator D is fed with both prediction path and ground truth.

1.7.3 Group Pooling

The motion of pedestrians in crowded scenes is highly affected by the interactions with other people in the neighborhood, which are influenced by their mutual relationship. Similar to the work in [24], we propose a pooling module for the generator G , which takes into account only the interaction of pedestrians who are not moving coherently in the neighborhood. The idea behind this design choice is that pedestrians walking in the same direction (thus coherently) share some interests such as the same goal, or willingness to talk and interact. We can exploit this behavioral cue to improve the performance in terms of path prediction. To this aim, we first use coherent filtering [168] to detect people walking coherently in crowds, and then adopt the social pooling method proposed by Alahi et al. [3] to model the interaction between socially-unrelated pedestrians only.

Group Clustering. Coherent motion reveals the collective movements of particles in a crowd. The coherent filtering is able to infer the coherent neighbor invariance, which measures the local spatio-temporal relation between pedestrians moving coherently. The coherent filtering is used to first detect the coherent motion of pedestrians in the scene. Then, clusters of points moving coherently are created. For our purposes, each point represents a single pedestrian, using, without loss of generality, the ground truth coordinates provided. Clusters can evolve, be deleted

and new clusters can emerge over time, with the goal of assigning each pedestrian i to a cluster s_i . The output of the coherent filtering consists of the sets $s_i (i = 1, 2, \dots, n)$ of people moving coherently. A pedestrian standing still or walking on their own is considered as belonging to his own set.

Pooling Module. We extend the social pooling module applied in S-LSTM [3] and S-GAN [58]. In the generator G of our Group-GAN, the pooling module allows pedestrians to share their hidden states, thus enabling the network to model interaction of people in the neighborhood.

The hidden state h_t^i in the pooling module represents the i_{th} pedestrian at frame t in the scene. The hidden-state dimension is set to D and the neighborhood size to N_0 , respectively. A tensor H_t^i represents the neighborhood of agent i and it is described as in Eq. 1.28, with dimensions of $N_0 \times N_0 \times D$:

$$H_t^i(m, n, :) = \sum_{j \in N} l_{mn}[x_t^j - x_t^i, y_t^j - y_t^i] l_{ij}[s_i \neq s_j] h_{t-1}^j \quad (1.28)$$

where $l_{mn}[x, y]$ is an indicator function to select pedestrians in the neighborhood defined as:

$$l_{mn}[x, y] = \begin{cases} 0 & \text{if } [x, y] \in \text{cell } mn \\ 1 & \text{if } [x, y] \notin \text{cell } mn \end{cases} \quad (1.29)$$

Two pedestrians i and j belonging to the same coherent set s_i will not be considered when computing the pooling module for each of them; this is modeled through the indicator function l_{ij} in Eq. 1.30:

$$l_{ij}[s_i \neq s_j] = \begin{cases} 0 & \text{if } i \in s_i, j \in s_i \\ 1 & \text{if } i \in s_i, j \notin s_i \end{cases} \quad (1.30)$$

The hidden state of pedestrian i will represent information about pedestrians, who are not moving coherently with i . Then the attention mechanism will take them as input of the hidden states.

1.7.4 Attention Mechanism

When a pedestrian changes direction, this observed piece of trajectory conveys more information than other parts. The network should be aware of such subtle variation and *focus* on that specific part of the input sequence. Moreover, as an additional element, the attention mechanism can help improve neighborhood interaction modelling. When hidden states arrive at the attention mechanism, they have already passed through the group pooling module and contain interaction information.

Inspired by the so-called *global attention mechanism* proposed by Luong et al. [103], we design our global attention mechanism and adopt it in the encoder-decoder framework of our Generator G . The concept of global attention is to consider all the hidden states of the encoder when deriving the context vector C_t , namely, in our case, involving all past trajectories. The context vector C_t is passed from encoder to decoder and carries the information needed for path prediction. The+ attention

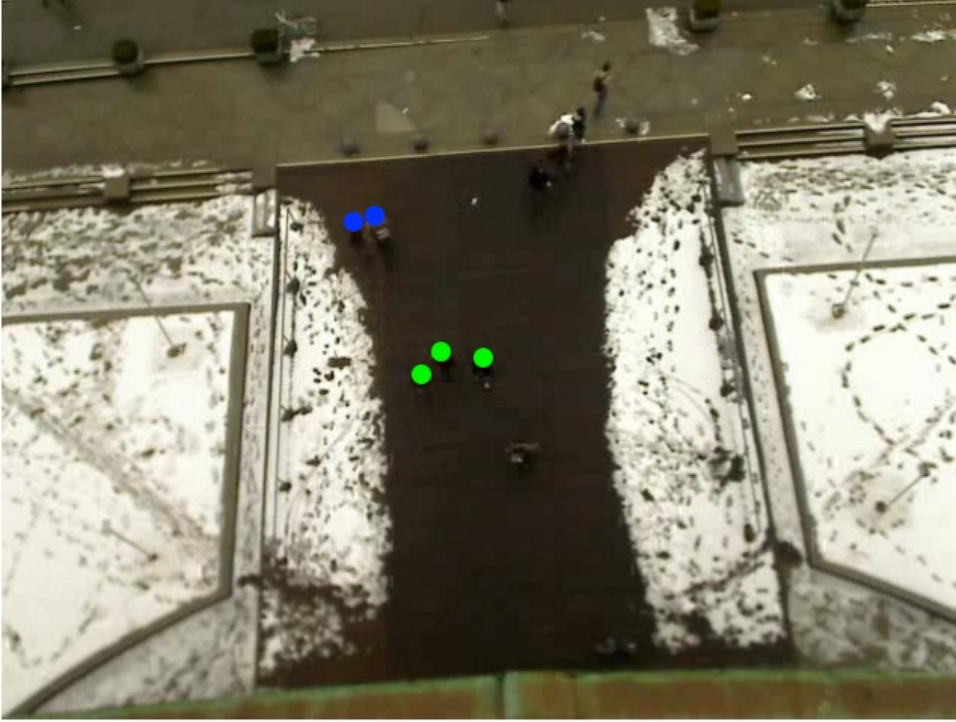


Figure 1.8: Each pedestrian is represented by a single keypoint. Pedestrians walking coherently are clustered into one group s_i . In this frame, two sets of pedestrians going in opposite directions are identified.

vector a_t is calculated as the product of the encoder output y_e and attention weights W_{attn} as indicated in Eq. 1.31:

$$a_t = y_e * W_{attn}, \quad (1.31)$$

where $*$ is the matrix product. In RNN-based encoder-decoder frameworks, the encoder mainly fulfills the task of encoding the input into the hidden states h_e^t and encoder outputs y_e are often not used. However, the encoder output y_e contains raw scores of the input, which can be utilized for labeling or attention as in [82, 98]. We use the encoder outputs to help the decoder have specific attention on the hidden states, based on the history of past trajectories. The attention weights is derived by the decoder input e_i and the current hidden states h_{di}^t as:

$$W_{attn} = \text{softmax}(\text{align}(e^i, h_t^{di})). \quad (1.32)$$

We use a softmax layer to get the attention weight after alignment. The decoder will take the attention vector a_t and hidden states as input, then predict future trajectories. Fig. 1.9 illustrates the attention module between encoder and decoder. The hidden states, encoder output, and decoder input, are all used for the decoder attention.

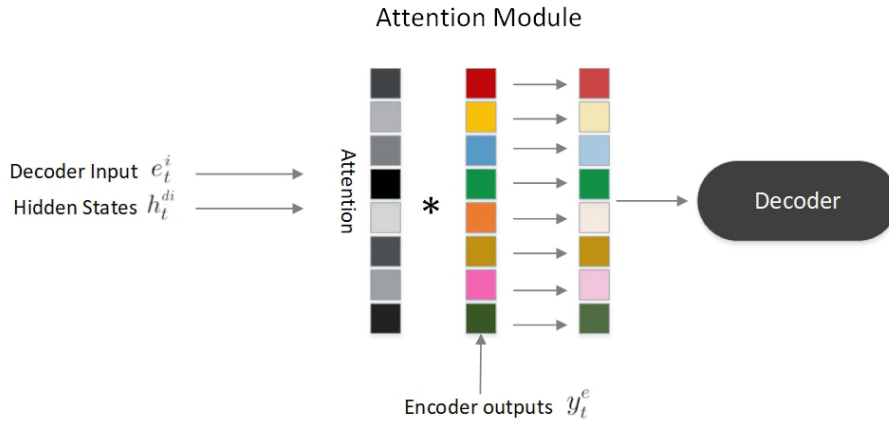


Figure 1.9: The attentive decoder in the Generator G of Group-GAN. Both decoder input and hidden states are squeezed into attention weights. The output is computed multiplying the encoder outputs with the weights to build the final attention vector.

1.8 Implementation details

For both training and validation, in line with the relevant works in the literature, we observe and predict trajectories using a time interval of 0.4 seconds. We observe trajectories for 8 time steps and predict for the next 12 time steps, meaning that we observe trajectories for $t_{obs} = 3.2$ seconds and predict for the next $t_{pred} = 4.8$ seconds. In the training phase, only trajectories that remain in the scene for at least 8 seconds are considered.

1.8.1 Group and Obstacle LSTM

For the implementation, we relied on the PyTorch framework, and the RMSprop is used to optimize the objective function. Our network is trained using the following hyper-parameters: 200 epochs, 0.0015 learning rate, 0.95 decay rate. The final model has 3 embedding input layers one for each input: the position p_t at time t , the Social Tensor and the Obstacle Tensor. In line with the state-of-the-art, our method is also evaluated on the UCY [87] and ETH [117] benchmark datasets. The UCY datasets consists of 3 videos of two different scenes with 786 people. The ETH dataset consists of two videos of different scenes containing 750 pedestrians in total. The training and test procedure is performed using a leave-one-out approach, namely, training on 4 sets of trajectories and testing on the remaining one.

In the first place, we need to configure the coherent filtering to cluster pedestrians. To this aim, we use $K = 10$, $d = 1$ and $\lambda = 0.2$ according to the original implementation.

For our LSTM network, we adopt the following configuration. The embedding dimension for the spatial coordinates is set to 64. The spatial pooling size, which corresponds to an area of $4 \times 4 m^2$, is set to 32. The pooling operation is performed using a sum pooling window of size 8×8 with no overlaps. The hidden state dimension is 128. The learning rate is set to 0.003, and RMS-prop [42] is used as the optimizer. The model is trained on a single GPU using a PyTorch¹ implementation.

¹<http://pytorch.org>

1.8.2 Group GAN

We trained iteratively the generator and discriminator using Adam optimizer [76] with batch size 64, 200 epochs and learning rate 0.001. The hyperparameters generation times k is set to 20 and L2 loss regularization weight λ is set to 1. The embedding dimension of input coordinates is 16. For Generator G , the hidden states dimension of both encoder and decoder is set to 32, while the hidden states of the encoder in D is set as a 48 dimensional vector intuitively. This is because only observation trajectories S_{obs}^N are provided as input to the Generator G whilst the Discriminator D takes as input the entire sequences $[S_{obs}^N, S_{pred}^N]$, therefore the Discriminator should have higher embedding dimensions.

1.9 Results

1.9.1 Quantitative Results

Metrics

Similarly to other works in the literature [117, 87], we evaluate our approaches using the following two metrics:

- *Average Displacement Error (ADE)*, namely the average displacement error (in meters) between each point of the predicted path with respect to the ground truth path.
- *Final Displacement Error (FDE)*, namely the distance (in meters) between the final point of the predicted trajectory and the final point of the ground truth trajectory.

Prior works [3, 58, 133] analyze trajectory matching using the ADE/FDE approach. However, ADE/FDE is a point-to-point comparison, which lacks of a global assessment on the trajectory similarity. For our Group-GAN, we adopt **Dynamic Time Warping (DTW)** [18] to calculate the distance between predicted trajectories $S_{pred}^N(t : t + \tau)$ and ground truth $S_{obs}^N(t : t + \tau)$. It can be formally defined as the minimization of the cumulative distance between two time series:

$$\begin{aligned} DTW(S_{pred}^N(t : t + \tau), S_{gt}^N(t : t + \tau)) &= \\ &= \sum_{i=t+1}^{t+\tau} \sum_{j=t+1}^{t+\tau} \min(\|S_{pred}^N(i) - S_{gt}^N(j)\|) \end{aligned} \quad (1.33)$$

where $\|\cdot\|$ is the Euclidean norm distance. As shown in Fig. 1.10, the DTW approach compares two trajectories at the positions where their relative distance is minimum. DTW has the benefit that the similarity of two trajectories can be assessed even if each trajectory is of different length or contains accelerations.

Baselines

In line with the existing works, we compare our results against some existing baselines: (i) a linear regressor that estimates linear parameters by minimizing the least square

error (Linear); (ii) a simple LSTM network (LSTM); (iii) S-LSTM, a model which combines the LSTM network with a social pooling layer [3]; (iv) *S-GAN* & *S-GAN-P*, social LSTM-based GAN that uses social pooling and hidden pooling, respectively [58]; (v) *Sophie*, a GAN-based framework leveraging on both social and physical context information [133].

Group and Obstacle LSTM

The proposed Group-LSTM (G-LSTM) performs on average better or equal than both Social-LSTM and Social-GAN, especially on the UCY dataset. This is due to the characteristics of crowd flows in the scene, which usually consist of easily identifiable groups walking in opposite directions. For the ETH dataset, the motion patterns are instead more varied and chaotic.

The achieved results show that the prediction performance can be improved when considering pedestrians that are not moving coherently. In fact, the change of motion and the evolution of trajectories are mainly influenced by pedestrians moving in different directions with respect to the pedestrian of interest. People walking together, instead, loosely influence each other, as they behave as a group.

The performance of our Obstacle-LSTM varies depending on whether we are using a combination of obstacles or semi-obstacles with either the Presence Tensor or Distance Tensor. The Obstacle-LSTM performs better than the other methods on the ETH dataset, thanks to the strong influence determined by obstacles in both scenes. In the *Zara* scene, instead, moving obstacles (e.g., car) have not been annotated and this causes a decrease of the network performance in the prediction task.

When combined together, Obstacle and Group-LSTM outperform other methods. The best results are obtained using a combination of the Group-LSTM with the Distance Tensor.

Table 1.1: Legend: G-LSTM (Group LSTM), OP-LSTM (Obstacle Presence LSTM), OD-LSTM (Obstacle Distance LSTM), SOP-LSTM (Semi-obstacle Presence LSTM), SOD-LSTM(Semi-obstacle Distance LSTM)). Quantitative results using the Group-LSTM and the Obstacle LSTM and the mentioned baseline approaches on the UCY and ETH datasets, respectively.

Metric	Dataset	G-LSTM	OP-LSTM	OD-LSTM	SOP-LSTM	SOD-LSTM
ADE	ETH [117]	0.48	0.40	0.40	0.46	0.44
	HOTEL [117]	0.47	0.44	0.43	0.46	0.40
	ZARA1 [87]	0.23	0.42	0.38	0.37	0.36
	ZARA2 [87]	0.34	0.40	0.40	0.34	0.37
	UCY [87]	0.56	0.39	0.40	0.38	0.37
	AVERAGE	0.42	0.41	0.40	0.40	0.39
FDE	ETH [117]	1.12	0.87	0.85	1.19	1.20
	HOTEL [117]	0.89	1.10	0.83	0.86	0.88
	ZARA1 [87]	0.91	1.01	0.81	0.86	0.81
	ZARA2 [87]	1.49	1.68	1.57	1.35	1.57
	UCY [87]	1.48	1.89	1.67	1.74	1.67
	AVERAGE	1.18	1.31	1.14	1.20	1.23

Table 1.2: Legend: G/OP-LSTM (Group Obstacle Presence LSTM), G/OD-LSTM (Group Obstacle Distance LSTM), G/SOP-LSTM (Group Semi-obstacle Presence LSTM), G/SOD-LSTM(Group Semi-obstacle Distance LSTM)). Quantitative results using our Group-LSTM combined with the Obstacle-LSTM and the mentioned baseline approaches on the UCY and ETH datasets, respectively. Two error metrics, namely, the Our model outperforms other approaches, especially in terms of average error.

Metric	Dataset	G/SOP-LSTM	G/SOD-LSTM	G/OP-LSTM	G/OD-LSTM
ADE	ETH [117]	0.36	0.40	0.38	0.36
	HOTEL [117]	0.34	0.31	0.33	0.33
	ZARA1 [87]	0.18	0.17	0.20	0.21
	ZARA2 [87]	0.26	0.20	0.26	0.22
	UCY [87]	0.73	0.81	0.75	0.64
	AVERAGE	0.37	0.38	0.38	0.35
FDE	ETH [117]	0.78	0.92	0.79	0.75
	HOTEL [117]	1.08	0.83	1.06	0.93
	ZARA1 [87]	0.80	0.69	0.90	0.92
	ZARA2 [87]	1.15	0.91	1.12	0.95
	UCY [87]	1.75	2.04	1.95	1.96
	AVERAGE	1.11	1.08	1.16	1.10

Group-GAN

Table 1.3 shows the evaluation of our Group-GAN model using ADE and FDE, comparing with the other baselines. For all datasets, we present the results in the ablative setting of our model where G refers to the model with only group pooling mechanism, A indicates the model with attention mechanism, and $G + A$ includes both group-aware pooling and attention mechanism.

As can be seen, the group pooling module G and attention mechanism A individually lead to raised performance, and combined solution $G + A$ gives a better result. It shows that the two modules work on different aspects of trajectory prediction, namely interaction modeling for G and motion history exploitation for A . The proposed combined model $G + A$ outperforms other state-of-the-art methods in ETH, HOTEL, ZARA2 sequences, as well as in terms of average results.

Table 1.3: Comparison of our model with other baselines trained by observing 8 time steps and predicting subsequent 12 time steps. Each time step corresponds to a 0.4 seconds shift. ADE/FDE are reported in meters. Our model achieves better results in ETH, HOTEL, ZARA2 datasets as well as in terms of average scores.

		Other baselines						Group-GAN		
	Datasets	Lin	LSTM	S-LSTM [3]	S-GAN [58]	S-GAN-P [58]	Sophie [133]	G	A	G+A
ADE	ETH	1.33	1.09	1.09	0.81	0.87	0.70	0.72	0.67	0.67
	HOTEL	0.39	0.86	0.79	0.72	0.67	0.76	0.43	0.47	0.33
	UCY	0.82	0.61	0.67	0.60	0.76	0.54	0.63	0.62	0.61
	ZARA1	0.62	0.41	0.47	0.34	0.35	0.30	0.35	0.35	0.36
	ZARA2	0.77	0.52	0.56	0.42	0.42	0.38	0.32	0.37	0.31
	AVG	0.79	0.70	0.72	0.59	0.61	0.54	0.49	0.50	0.46
FDE	ETH	2.94	2.41	2.35	1.52	1.62	1.43	1.28	1.27	1.21
	HOTEL	0.72	1.91	1.76	1.61	1.37	1.67	0.87	0.95	0.67
	UCY	1.59	1.31	1.40	1.26	1.52	1.24	1.30	1.31	1.28
	ZARA1	1.21	0.88	1.00	0.69	0.68	0.64	0.70	0.72	0.73
	ZARA2	1.48	1.11	1.17	0.84	0.84	0.78	0.65	0.76	0.64
	AVG	1.59	1.52	1.54	1.18	1.21	1.15	0.96	1.00	0.90

Table 1.4 presents the mean DTW between predicted trajectories and ground truth of each model. Based on the motion history and interaction modeling, a trajectory prediction framework should be able to give not only precise prediction, but also trajectories similar in shape when compared to the ground truth. Our solution consistently outperforms other models, indicating that our model generates trajectories that are more similar to the ground truth.

Pedestrian Collision Rate. As people have the instinct to avoid collision with others in real scenarios, the collision rate can somehow reflect the effectiveness of interaction modeling. We assess our approach also by computing the collision rate, namely when the distance of two pedestrians is smaller than $0.1m$; the metric is computed at each frame as presented in Table 1.5. Thanks to group-aware pooling and attention mechanism, our model is able to offer better collision avoidance for pedestrians. ETH dataset displays a relatively sparse corridor where pedestrians mainly walk oppositely to each end, and the motion direction is therefore more predictable. Accordingly, no collision is observed for all models. For the sequence HOTEL, people tend to walk straight to their destination and almost no interaction

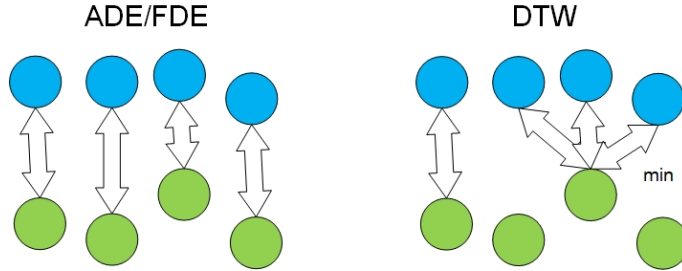


Figure 1.10: Using ADE/FDE vs DTW in comparing two trajectories. ADE/FDE takes two locations at the same timestamp while DTW compares two timestamps with minimum distance.

Table 1.4: Comparison of mean DTW of each trajectory for each model. For each model, the mean DTW is calculated between the prediction S_{pred}^N and ground truth S_{gt}^N . The lower the value, the higher the similarity of the generated trajectories.

mean DTW	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
LIN	2.08	0.76	5.80	1.60	1.93	2.43
S-LSTM [3]	1.72	1.33	2.78	0.72	0.94	1.50
S-GAN [58]	1.53	1.08	2.71	0.63	0.84	1.36
Group-GAN	1.43	0.73	2.82	0.67	0.78	1.29

Table 1.5: Comparison of the average collision rate % of the prediction trajectories S_{pred}^N per frame for each model.

%	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
LIN	0	0.06	4.57	0.15	1.09	1.17
S-LSTM [3]	0	0.18	7.21	0.11	0.52	1.60
S-GAN [58]	0	0.22	7.40	0.11	0.46	1.64
Group-GAN	0	0.13	4.45	0.14	0.40	1.02

Table 1.6: Legend: Lin. (Linear [3]), S-LSTM (social LSTM [3], S-GAN (social GAN [58])). The Average Displacement Error (ADE) and the Final Displacement Error (FDE) are reported (in meters) for an observation interval $t_{obs} = 3.2$ seconds and a prediction of subsequent $t_{pred} = 4.8$ seconds.

Metric	Dataset	Lin. [3]	S-LSTM [3]	S-GAN [58]
ADE	ETH [117]	1.33	1.09	0.81
	HOTEL [117]	0.39	0.67	0.72
	ZARA1 [87]	0.62	0.41	0.34
	ZARA2 [87]	0.77	0.52	0.42
	UCY [87]	0.82	0.61	0.60
	AVERAGE	0.79	0.70	0.58
FDE	ETH [117]	2.94	2.41	1.52
	HOTEL [117]	0.72	1.91	1.61
	ZARA1 [87]	1.21	1.11	0.84
	ZARA2 [87]	1.48	1.31	1.26
	UCY [87]	1.59	0.88	0.69
	AVERAGE	1.59	1.52	1.18

can be captured, thus interpreting, why the linear model outperforms others baselines in terms of collision rate. The UNIV sequence, which is part of the UCY dataset, covers a variety of complex human behavior with high crowd density. If solely social pooling is applied as in S-LSTM [3] and S-GAN [58], the pedestrian of interest will tend to "push" others away from his neighborhood, which is likely to generate collisions.

In summary, the presented results indicate our model is able to generate socially-acceptable trajectories, which are in many cases more realistic than other state-of-art methods. This is also confirmed when adopting different metrics.

1.9.2 Qualitative Results

Group and Obstacle LSTM

In Section 1.9.1 we have shown that considering only pedestrians not moving coherently can improve the prediction precision. In this section we will further evaluate the consistency of the predicted trajectories.

As a general rule, the LSTM-based approaches for trajectory prediction follow a data-driven strategy. Furthermore, the future planning of pedestrians in a crowd are highly influenced by their goals, their surroundings, and their past motion history. Pooling the correct data in the social layer can promote the prediction performance in a significant way.

In order to guarantee a reliable prediction, we not only need to account for spatio-temporal relationships, but also need to preserve the social nature of behaviors. According to the studies in interpersonal distances [60, 39], socially-related people tend to stay closer in their personal space and walk together in crowded environments as compared to pacing with unknown pedestrians. Pooling only unrelated pedestrians

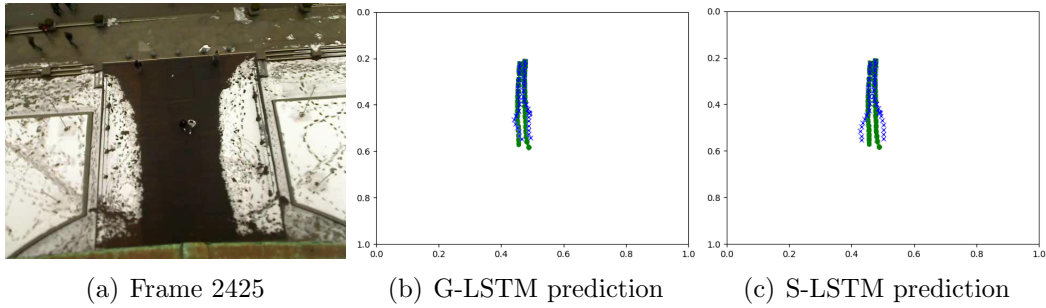


Figure 1.11: ETH dataset Frame 2425: the prediction is improved when pooling in the social tensor of each pedestrian only pedestrians not belonging to his group. The green dots represent the ground truth trajectories; the blue crosses represent the predicted paths. Using the grouping module allows the predictor to keep pedestrians belonging to the same group closer than in the case of S-LSTM.



Figure 1.12: Sequences taken from the UCY dataset. An interaction example between two groups is shown. An in-depth analysis is further presented in Figure 1.13.

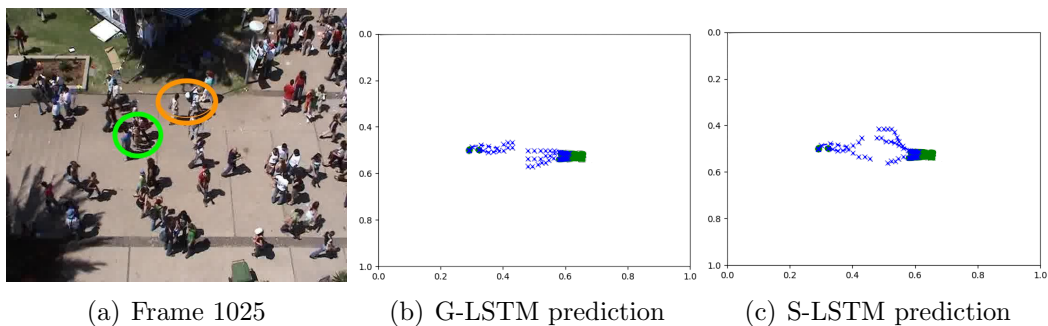


Figure 1.13: UCY univ Frame 1025. We display how the prediction is improved for two groups walking in opposite directions. The green dots represent the ground truth trajectories, while the blue crosses represent the predicted paths. The G-LSTM predicts socially-related people to stay closer together than in the S-LSTM approach.

will focus more on macroscopic inter-group interactions rather than intra-group dynamics, thus allowing the LSTM network to improve the trajectory prediction performance. Collision avoidance influences the future motion of pedestrians in a similar manner if two pedestrians are walking together as in a group.

In Figure 1.11, Figure 1.13 and Figure 1.12, we show some examples of predicted trajectories, which highlight how the Group-LSTM is able to predict pedestrian trajectories with better precision, demonstrating how the prediction is improved when we pool in the social tensor of each pedestrian only pedestrians not belonging to the group.

In Figure 1.11, we show how the prediction of two pedestrians walking together in the crowd improves, when they are not pooled in each other’s pooling layer. When the two pedestrians are pooled together, the network applies the typical repulsion force to avoid collisions. However, since they belong to the same group, each pedestrian *allows* the other to remain closer.

In Figure 1.12, we display the sequences of two groups walking toward each other. In Figure 1.13, we show how the prediction for the two groups is improved with respect to the Social-LSTM, demonstrating the ability to forecast how pedestrians belonging to the same group stay together when moving in the environment.

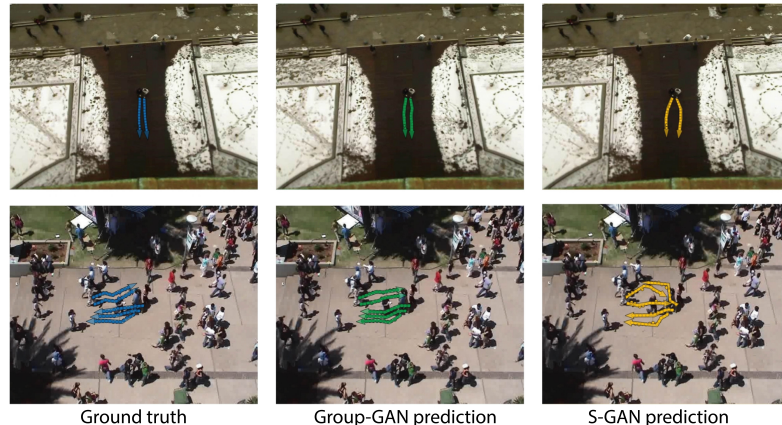


Figure 1.14: Two groups are walking in opposite direction at frame 2425 of ETH dataset (top row) and one group is moving coherently in the same direction at frame 986 of UCY dataset (bottom row). The pedestrians stay in the personal space of each other. Our model preserves the coherent group motion, therefore displaying a behavior closer to ground truth.

Group GAN

In this section we present some qualitative results comparing our Group-GAN with the existing literature. We have seen how the coherence of motion among pedestrians in a group is an important property; preserving this property by pooling only unrelated surrounding pedestrians can promote the prediction performance significantly. As the existing GAN framework in trajectory forecasting [58] use diverse sample generation and chooses the one with minimum ADE, the prediction

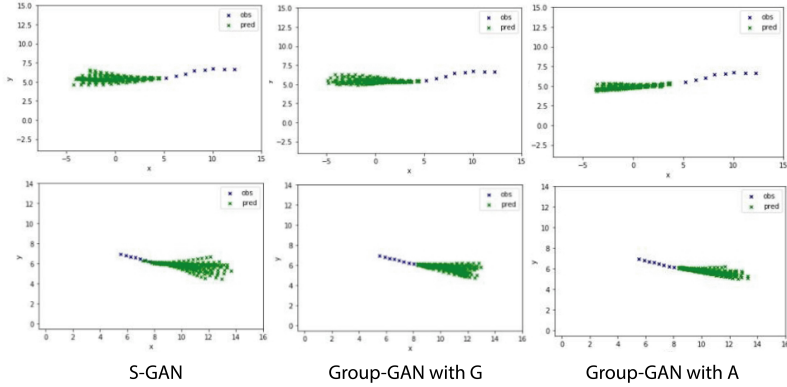


Figure 1.15: 20 generated trajectories for a pedestrian in ETH dataset (top row) and ZARA2 dataset (bottom row). The blue crosses represent the ground truth and the green crosses represent the predicted paths. Thanks to the use of Group-Aware pooling and attention mechanism, generation convergence is improved, getting closer to the ground truth.

quality is somehow concealed by generation times. The dispersion degree or, in other words, generation convergence, is able to reveal this aspect only in part.

Coherent motion. As mentioned above, and to guarantee a reliable prediction, not only spatio-temporal relationships, but also the social nature of behaviors need to be taken into consideration. According to interpersonal space study [60], socially-related people tend to stay closer in their personal space and walk together in crowded environments, as compared to pacing with unknown pedestrians. Pooling only unrelated pedestrians will focus more on macroscopic inter-group interactions rather than intra-group dynamics, which allows the model to improve trajectory prediction performance.

Fig. 1.14 presents two cases, in which our model improves trajectory forecasting when compared against S-GAN. Two groups are walking in opposite directions in ETH sequence (bottom row). Our model preserves the coherent group motion, thus predicting better future states. One group of two pedestrians is walking together in the same direction as shown in UCY sequence (top row). They keep staying in their personal space and walk coherently in our model, while a repulsive behavior can be observed in S-GAN.

Prediction Convergence.

As discussed in Sec. 1.7.2, we adopt variety loss, then we generate k samples once and we choose the one with minimum ADE. The dispersion degree indicates the prediction quality. Therefore it is crucial for the model to make the trajectory prediction converge as close to ground truth as possible. Fig. 1.15 provides two examples of 20 generated trajectories for one pedestrian in ETH and ZARA2 sequences respectively. We apply both G and $G + A$ settings of Group-GAN to validate the convergence. The generation convergence improves, being closer to the ground truth.

In Section 1.9.1 we have shown that considering only pedestrians not moving coherently can improve the prediction precision. In this section we will further evaluate the consistency of the predicted trajectories.

As a general rule, the LSTM-based approaches for trajectory prediction follow a

data-driven strategy. Furthermore, the future planning of pedestrians in a crowd are highly influenced by their goals, their surroundings, and their past motion history. Pooling the correct data in the social layer can promote the prediction performance in a significant way.

In order to guarantee a reliable prediction, we not only need to account for spatio-temporal relationships, but also need to preserve the social nature of behaviors. According to the studies in interpersonal distances [60, 39], socially-related people tend to stay closer in their personal space and walk together in crowded environments as compared to pacing with unknown pedestrians. Pooling only unrelated pedestrians will focus more on macroscopic inter-group interactions rather than intra-group dynamics, thus allowing the LSTM network to improve the trajectory prediction performance. Collision avoidance influences the future motion of pedestrians in a similar manner if two pedestrians are walking together as in a group.

1.10 Conclusions and future work

In this chapter, we tackled the problem of pedestrian trajectory prediction in crowded scenes. We propose a novel approach, which combines group detection and obstacle annotation to refine the prediction. With the Group-LSTM, coherent filtering is used to identify pedestrians walking together in a crowd, while the LSTM network is used to predict the future trajectories by exploiting inter and intra group dynamics. The Obstacle-LSTM displays how obstacles and non-trespassing areas can strongly influence future trajectories of pedestrian. Moreover we present a novel LSTM-based Attentive Group-Aware GAN framework for trajectory forecasting. Both coherent group clustering and attention on the hidden states are exploited, yielding to social plausible trajectory prediction. Experimental results show that the proposed LSTM framework, which embeds group and obstacle information, outperforms the Social-LSTM in the prediction task on two public benchmarks (the UCY and the ETH datasets). In future work, we expect to improve the model, making it possible to predict longer time ranges, a limitation currently imposed by the available annotated datasets.

2 Crowd Simulation

2.1 Introduction

Although a considerable amount of research has been carried out in the domain of behavior prediction and crowd motion analysis, there is still a lack of a unified framework for validation. This is due to the fragmentation and heterogeneity of datasets used for testing and benchmarking, which often suffer from the scarcity of training and testing data. Furthermore, the quality of videos, size of datasets, duration of sequences, content, the density of the crowd, quality of the annotation, are only a few variables that make it difficult to critically evaluate different methods. To address this problem one of the possibilities is to rely on simulators, which, although they may not report scenes as natural as the real videos, provide a considerable number of advantages. On the other hand, a simulation framework allows the quick generation of large sets of crowd configurations.

The integration of simulators in the processing pipeline has initially been ruled by purely agent-based models [46] thus conducting the analysis of their ground-truth position. Only recently we have observed a trend that leverages the idea of generating virtual videos instead of virtual agents' behavior [33]. The substantial difference between the two consists of carrying out the analysis with standard computer vision techniques, using for test the videos generated by the simulator. This allows taking into account potentially all the challenges of a real video, including the presence of occlusions, obstacles, changes in the illumination conditions, the similarity of objects in the appearance model, etc. The use of simulators in computer vision is not new and has been subject of research for object tracking and camera control [120, 121]. However, the quality of the rendering and the simulation tools are still not providing an acceptable *visual fidelity* [153], i.e. a reliable representation of a realistic scenario. *Visual fidelity* refers in our case to both the pleasantness of the video sequence when presented to a human observer, and, mostly, the reliability when processed by a computer vision algorithm. There is a high demand for more realistic and autonomous crowds, also driven in part by the movie industry and the virtual reality domain [143].

2.1.1 Properties of a good crowd simulator

A real video sequence of a crowd and its virtual counterpart should return the comparable quantitative and qualitative results when processed by an automatic analysis algorithm. Strong similarities should then be achieved not only on the visual appearance side but also on the human motion models. Such features refer to *model validation*. Three different layers of the modeling task can be identified:

- **Appearance modeling.** The simulated crowd should be rendered in a realistic way, and with different levels of details depending on their distance from the camera.

- **Realistic, smooth and adaptable motion.** Agents in the simulation should be able to fluently move and change the motion behavior, like going from walking to running or to stopping, depending on the situation.
- **Realistic high-level behavior (dynamics).** This is the most difficult challenge to achieve. The simulation should be able to consistently reproduce the behavior of the crowd in a variety of situations and environments, like in scenes populated with different densities or in emergency situations.

2.1.2 The advantages of a crowd simulator

The main advantages of crowd simulation frameworks are the implicit knowledge of the ground truth, crowd behavior customization, environment modeling, and virtual camera layout.

In fact, the task of manually annotating test sequences is time-consuming and, in densely populated environments, due on the one hand to the high number of objects and the rapid evolution of the scene, and on the other hand, to the uncertainty in determining the ground truth coordinates of these objects because of persistent occlusions, it is challenging to annotate the video because a person can be completely occluded and pedestrians far away from the camera may be indistinguishable from each other. One of the main benefits is, therefore, the implicit knowledge of the *ground truth*.

Simulators can generate an arbitrary number of video sequences, where any sort of behavior can be repeated by changing the configuration of the environment, altering the density of people in the scene, their appearance models, the lighting setup. In addition, simulators can help to reproduce anomalies (e.g. dangers and threats) that are rare in the existing datasets [77]. Simulating crowds also allows us to completely customize the scenes in terms of both the crowd’s behavior and features. One of the focal points of research in crowd behavior analysis is the detection of anomalies [105], such as spotting a pedestrian moving in the opposite direction with respect to the crowd main flow or identifying an abandoned item in a crowded environment.

Recreating the needed sequences to test and validate algorithms for anomalies detection is an important feature of crowd simulators. Other areas of interest in the crowd behavior analysis field are the behavior of the crowd in specific cases, like in emergency situations.

Simulation can also be exploited to improve the robustness of available analysis algorithms replicating the same events with different settings. In particular, how the density of the crowd can affect its reaction to a specific event.

Moreover, a real scene can be replicated in the simulation environment in order to record it from a different viewpoint with respect to the original one. The same local event can also be reproduced in different crowd density levels to understand how the crowd reacts in different conditions in the same environment.

Moreover, environment modeling can also be used to simulate yet-to-be-built buildings to make sure they can safely function in crowded situations.

Developing a *virtual vision* paradigm to simulate a CCTV camera circuit and perform an action like surveillance and tracking in a simulated environment has been subjected to researches [122]. The simulator provides intrinsic and extrinsic parameters of all the virtual cameras used to record a synthetic test sequence in

the virtual environment, thus allowing to replicate the setup in the real world after the model has been tested in the simulated environment. Many kinds of cameras, like PTZ (Pan, Tilt, and Zoom) cameras and wide field of view cameras, have been recreated and deployed in the synthetic environment.

Another advantage of the simulator is the possibility to triangulate the information deriving from different cameras since the ground truth not only indicates the position in pixel of each pedestrian from each viewpoint, but also the 3D position and the identity of the person.

2.1.3 Simulation-driven analysis

As discussed in section 2.1.2, one of the main advantages of a simulator is the implicit knowledge of the ground truth, which is the information of how many and which agents are present in the test sequence at a given time. In really crowded scenes, where one agent is represented by just few pixel in an image, if he is not completely occluded, the simulator precision outperforms even the manual counting. Precisely annotated synthetic test sequences can be used to validate crowd analysis algorithms [97], like people counting or density estimation. In the case of tracking multiple people in crowded environments, the ground truth data requires the labeling of people in the environment across time and, if needed, across multiple points of view. This task is really time-expensive if done manually, can be automatically performed by simulators. In [70], they validate a tracking and classification algorithm for groups of people on synthetic data. In [142, 122], simulation is used to test and validate surveillance systems, where cameras are placed in the environment to perform specific surveillance tasks.

Despite all the progress in both social interaction models and computer graphics, using simulators to validate and improve crowd analysis algorithms still presents some open issues. Social interaction models are mathematical models developed to replicate the average behavior of a human in crowds but are not capable of reproducing unpredictable behaviors. Moreover, collision avoidance models have been developed approximating agents as either circles or dots. Although this can be acceptable in low-density scenarios, it is not feasible for high-density ones like in [140], where people are subject to actual collisions and does not have enough space to walk normally. Moreover, the quality of the output video of some simulator still has to be optimized, since multiple input parameters can be modified and ad-hoc created scenes can differ from real scenarios both in terms of video quality and actual scene. For instance, a tracking algorithm can perform really well when all the agents are dressed in different colors and can fail when agents are too similar in real scenarios.

2.1.4 From real to synthetic crowds: data-driven crowd simulation

Crowd analysis algorithms can be used to extract data from real scenes. This data can be used as the input of a simulator, thus improving the realism of simulated scenes. Data-driven crowd simulation protocol has been applied in the development of multiple simulators.

Optical-flow based methods to learn the main motion pattern in a given environment are often used [67, 41]. The extracted information is usually used to generate

velocity fields that carry the information of the usual crowd behavior in a specific environment. Other methods such as [15, 112], use information derived from tracking pedestrians in both crowded and non-crowded environments to improve the realism of the simulation. The main issue in this field is the lack of a recognized evaluation metric to assess the realism of a crowd simulation result. If the simulation is related to an existing, real-life scenario, it can be compared using tools discussed in the previous section. Transfer learning and validation of new simulating scenarios remains an open issue.

2.2 Data-driven crowd simulation

In this section, we propose a framework for data-driven crowd simulation starting from a small set of trajectories [22]. To model **the high-level behavior**, our method extracts pedestrian trajectories from real videos, clusters all trajectories of pedestrians who intend to reach the same goal and computes the velocity field associated with each exit region in the scene to guide virtual agents toward their destinations, namely, the goal-dependent path selection. While at the microscopic level, the simulation is performed using, on the one hand, the Social Force Model to handle the collision-avoidance among agents and on the other hand the computed velocity fields to model the macroscopic behavior. The experimental results demonstrate that the velocity field can be exploited to effectively reproduce crowd behaviors.

2.2.1 Introduction

The development of crowd simulators using agent-based techniques (such as the SFM [64] and the RVO [148]) has been the research focus [112, 167, 114, 166] in the past years. However, for the data-driven crowd simulation, some important issues have not been solved completely. First of all, the behavior parameters (for instance the parameters in the SFM and RVO model) should be learned and fine-tuned carefully to match the data extracted from the real case scenario. How to learn the model parameter from a realistic scenario is usually not a straightforward problem since the behavior of the crowd in a scene is strongly influenced by the density of the agents.

The RVO and the SFM have been developed to model collision avoidance behaviors among agents at the local level. At the local level, both of them have shown the capability of displaying the desired collective behavior in the crowd, such as lines of people moving in the same direction within the crowd. However, at the global level, the above two models are not able to simulate complex behaviors, such as path-planning and goal-selection. These behaviors are always present in crowded environments such as train stations, airports, and shopping malls. To model these scenarios, different algorithms have been proposed. The most common approaches are based on the continuum model [146] and on velocity fields [114, 167, 112].

Zhong et al. [167] propose a data-driven crowd simulation framework, which uses a dual-layer agent-based model to navigate in the scene. The lower layer deals with modeling the local collision avoidance behaviors based on the SFM. The upper layer deals with the global behavior of agents, which involves a deeper understanding of the scene to perform goal-selection and path-planning. Moreover, we also improve the way that the presence of stationary groups affects the crowd. The upper layer in the reference method uses trajectories extracted from realistic scenes to compute

the velocity fields, which guides agents at the global level. Their algorithm is evaluated on the New York Grand Central Terminal dataset, which consists of 40,000 trajectories around. Our proposed approach can make the algorithm suitable for small-scale datasets as well (containing about 1,000 trajectories), by integrating more sophisticated priors of the environment and behavioral models in the processing pipeline.

2.2.2 Related work

Recently, many approaches to automatize the processing of real-world crowd data have been developed. Data-driven crowd simulation techniques have been proposed to simulate crowd behaviors as close/natural as possible with respect to the realistic ones. Common protocols to evaluate the similarities are the density-based metric [88] and entropy-based metric [59]. In [154], the authors propose a method to optimize the parameters for different motion models. Pellegrini et al. [117] and Scovanner et al. [135] also use optimization methods (i.e. the Gradient-based Newton method, and the Genetic Algorithm) to obtain proper parameters for motion model learning. Musse et al. [112] propose a method to compute the velocity field using trajectories extracted in low-density scenarios. Patil et al. [114] propose a method for guiding crowds using the velocity and navigation fields either extracted from realistic scenarios or sketched manually. In [92], Li et al. use a copy and paste technique to populate the simulation environment, pasting together crowd features and trajectories extracted from real-world data. In a similar way, Zhao et al. [166] use examples extracted from real videos and reproduce crowd states over time using a neural network. Rodriguez et al. [130] propose a method to learn motion priors to help crowd analysis by transfer learning across scenes. In the computer vision field, many works have been proposed on learning global and local crowd motion features from real scenarios. Lin et al. [95, 96] use an algebraic approach to learn crowd flows from videos using the geometric flow estimation instead of the motion flow. Ali et al. [5] use a Lagrangian particle system approach to segment crowd motion and detect motion instabilities in dense environments. Mehran et al. [107] use a streak-line representation to improve the accuracy in understanding spatiotemporal changing behaviors of the crowd. Although the aforementioned works focus on extracting and understanding collective motion patterns from the scenes, they have not integrated common agent-based models into the simulation procedure. The most used agent-based models are the Social Force Model (SFM) proposed by Helbing et al. [64] and RVO (Reciprocal Velocity Obstacle) proposed by van Berg et al. [148]. In many recent works, the above two models are used to handle the so-called collision avoidance behaviors, while approaches using the velocity fields [114, 167, 112] or continuum approaches [146] are used to model the global behaviors such as path planning. In [114], the velocity and guidance fields are coupled with RVO and SFM, alternatively, to model the desired crowd motion patterns. In the work proposed by Zhong et al. [167], trajectories are extracted to compute velocity fields. The velocity fields are then used to perform the simulation at the macroscopic level, while the SFM is used for collision avoidance at the microscopic level. Their method has some limitations due to the high number of trajectories used for training, and the difficulties in describing crowd motion in the presence of social rules, like behaviors

at crossings and roundabouts.

2.2.3 Methods

In this work, we adopt the Stanford Drone dataset [129], which consists of annotated scenes with different agent types (i.e., pedestrians and bikers). We choose this dataset due to the positions of recording cameras in the top-view, which allows for an accurate mapping of the environment. The top-view position also allows projecting the velocity field from the camera plane to the ground plane without introducing too much uncertainty. Trajectories are clustered with respect to their final destinations, and the clustering results are then used to compute the velocity fields.

Velocity field computation

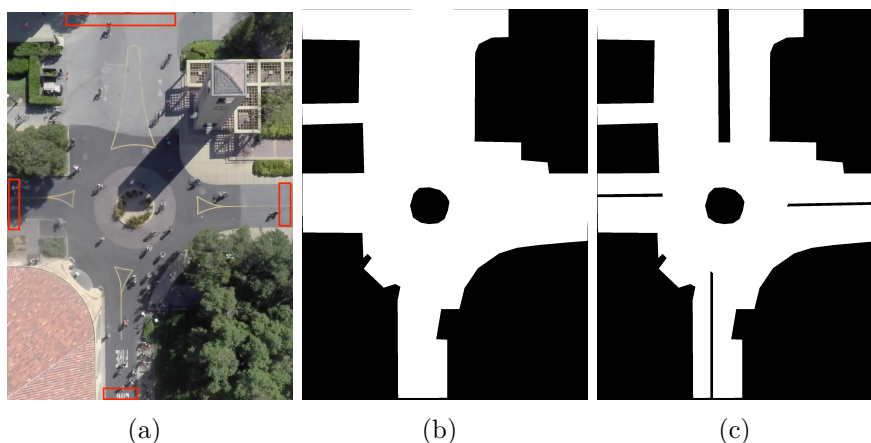


Figure 2.1: Examples: a real scenario from the top view, with the entry/exit region highlighted in the red boxes (a), the mask to describe the obstacles in the environment (b), and the mask to describe both real obstacles and virtual obstacles imposed by social/external rules (c).

The trajectory clustering algorithm adopted in our approach is a modified version of the one presented in [167]. The original algorithm is designed to work with thousands of trajectories that are not available in our dataset. In the initial step, we do not take obstacles into consideration in the scenario. This situation is handled in the refinement stage.

The velocity field is defined as a vector field, where each vector determines the motion status (i.e. velocity and orientation) of a pedestrian in his neighborhood. Each exit region (ER) in the scene is associated with a specific velocity field. Pedestrians in the same local region can move in different directions, depending on their goals. In order to compute the velocity fields for each ER, the whole space is divided into $H * W$ discrete grids evenly, where each grid has the size of $\Delta * \Delta$. The i -th velocity

field corresponding to the $i - th$ exit region is defined as in Eq. 2.1:

$$V_i = \begin{bmatrix} v_{i,1,1} & v_{i,1,2} & \cdots & v_{i,1,W} \\ \vdots & \vdots & \vdots & \vdots \\ v_{i,H,1} & v_{i,H,2} & \cdots & v_{i,H,W} \end{bmatrix} \quad (2.1)$$

where $v_{i,h,w}$ represents the orientation vector that guides pedestrians when they are located in cell (h, w) . The value of each cell can be learned from realistic scenarios. The center of each cell (h, w) is defined as $C_{h,w} = (x_{h,w}, y_{h,w})$. In the initialization step, every cell in the velocity field V_i is initialized with a vector that points towards the corresponding ER_i , as demonstrated in Eq. 2.2, where Λ represents a normalization function:

$$v_{i,h,w} = \Lambda(ER_i - C_{h,w}) \quad (2.2)$$

At this point, we apply a modified version of the K-means algorithm described in [167], to segment and cluster different trajectories in order to obtain the velocity field with respect to each ER. In the beginning, the velocity fields are sparse since only grids that trajectories have passed through are initialized. For other grids without obstacles inside, we still need to assign them some specific values. In order to distinguish areas with obstacles from non-obstacle areas, we adopt a binary mask of M as shown in Fig. 2.1. The mask is discretized in the same way as the velocity field, which is initialized as in Eq. 2.3:

$$m_{h,w} = \begin{cases} 0, & \text{if the cell does not contain any obstacle} \\ 1, & \text{if the cell contains an obstacle} \end{cases} \quad (2.3)$$

For the grids that do not contain any obstacle, we set the empty cells of the velocity field as a least-effort field, instead of initializing every vector pointing directly towards the destination. This allows for smoother navigation in the environment since people are implicitly aware of fixed obstacles in the least-effort hypothesis. To compute the least-effort path, a heat-map is proposed to determine the path-distance between the destination and every cell. The computed value is stored in the cell for every possible goal. The path-distance is the minimum number of non-obstacle cells to be traversed to reach the goal. An example of the heat-map is shown in Fig. 2.2. The cells of the velocity field are then initialized in the descent direction of the heat-map gradient.

The clustering results are a little bit noisy due to the small number of trajectories. The results are then refined using the following procedure. First, the neighbors of a given cell are selected with the kernel (as shown in Eq. 2.4 centered on a non-obstacle cell $v_{i,h,w}$ which has the orientation $\angle v_{i,h,w}$.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.4)$$

Then, the average orientation $\angle v_{avg}$ of the neighbors is computed. The orientation $\angle v_{i,h,w}$ is finally updated using Eq. 2.5.

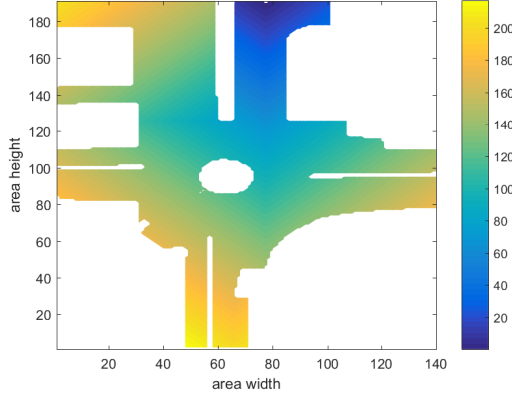


Figure 2.2: The heat-map shows the distance of each point from the top exit. The darker the color, the closer the point is to the goal. One heat-map is computed for each possible goal.

$$\angle v_{i,h,w} = \begin{cases} \angle v_{i,h,w}, & \text{if } \angle v_{i,h,w} \in \{\angle v_{avg} \pm 90^\circ\} \\ \angle v_{avg}, & \text{if } \angle v_{i,h,w} \notin \{\angle v_{avg} \pm 90^\circ\} \end{cases} \quad (2.5)$$

Next, the computed velocity field is used to model macroscopic behaviors, such as path planning.

Social Force Model

The Social Force model exploits a Newtonian approach to avoid collisions. The Social Force Model (SFM) [64] is frequently used to model the collision avoidance behavior in the crowd. The model describes pedestrian motion dynamics taking into account personal goals and environmental constraints.

The SFM equation of the total force imposed on each agent at a given time consists of three main parts: the $f^{att}(a_i)$ is the attractive force that drives a pedestrian towards his destination, the repulsive force $F^{ped}(a_i)$ reflects the psychological tendency to maintain a social distance between individuals, and the environmental force $F^{obj}(a_i)$ allows an agent to wander in the surroundings avoiding colliding with obstacles (walls, trees, etc.). Let A indicate the set of agents, where $a_j \in A$ is a specific agent. Let O indicate the set of obstacles, where $o_k \in O$ is a specific obstacle. So, the social force imposed on a pedestrian a_i at a given time can be modeled as in Eq. 2.6:

$$F_a(a_i) = f^{att}(a_i) + \sum_{a_j \in A, j \neq i} f_j^{ped}(a_i) + \sum_{o_k \in O} f_{o_k}^{obj}(a_i) \quad (2.6)$$

The attractive force $f^{att}(a_i)$ depends on the preferred velocity $v_{a_i}^p$ of each agent and is defined as in Eq. 2.7, where m_i is the mass of the agent, τ represents the reaction time, and v_{a_i} is the current velocity of the agent.

$$f^{att}(a_i) = m_i \frac{1}{\tau} (v_{a_i}^p - v_{a_i}) \quad (2.7)$$

The preferred speed of each agent $v_{a_i}^p$ depends on its motion parameters, while the direction depends on the velocity field. The preferred velocity $v_{a_i}^p$ can be expressed as in Eq. 2.8, where P is a constant, which depends on the personal characteristics and tendencies of each agent, $|v^p|$ is the absolute value of the speed at which people usually move in the environment, and \hat{v} is the orientation stored in the nearest cell of the velocity field corresponding to the agent’s destination.

$$v_{a_i}^p = P|v^p|\hat{v} \quad (2.8)$$

Moreover, the basic social force model can be updated by introducing the behavior of multiple people walking coherently as suggested in [111]. If an agent belongs to a socially bounded group, a social group force factor f_i^{group} is added to Eq. 2.6, as shown in Eq. 2.9, where f_i^{vis} allows the agent to keep all the members within his group and enhance their communication, f_i^{att} keeps agents within the same group moving in a tight way, and f_i^{rep} is a repulsion force that avoids pedestrian colliding with each other.

$$f_i^{group} = f_i^{vis} + f_i^{att} + f_i^{rep} \quad (2.9)$$

2.2.4 Experimental results

We evaluate our crowd simulation framework using the *deathCircle* video in the dataset presented in [129]. The scene (shown in Fig. 2.1(a)), consists of 4 entry and 4 exit regions, which is recorded from the top-view. The ground truth attached to this video consists of 870 trajectories within a short period.

The Velocity Field Computation

The velocity fields are computed using all the trajectories extracted from the *deathCircle* video. The maximum iteration of the clustering algorithm is set to 10. The image of $1400 * 1900$ pixels is sampled in cells of $10 * 10$ pixels. Therefore, the velocity field consists of a grid of $140 * 190$ cells. Fig. 2.3 shows the computation of the velocity field at different stages.

From Fig. 2.3 (d) and (h), it can be observed that the velocity field navigates agents towards their destinations as we expect. Comparing to the baseline method (Fig. 2.3 (b) and (f)), the new velocity field can provide a more sophisticated path-planning schedule, which allows smoother navigation in the environment.

Crowd Simulation

The validation of the crowd simulation is performed by comparing the density distribution between the realistic scenarios and the corresponding simulated videos. The density distribution of a realistic crowd is estimated using 870 trajectories. Considering the computational burden, the distribution of the simulated video is computed from around 200 sampled trajectories. The discrete points from where the distribution value is computed are chosen every 100 pixels in the image space, thus resulting in a grid of $(I, J) = 14 * 19$ points. The density of the distribution is computed using the equation suggested by Helbing et al. [63], as shown in Eq. 2.10, where n represents the total number of points of all trajectories; $d(i, j, k)$ is the

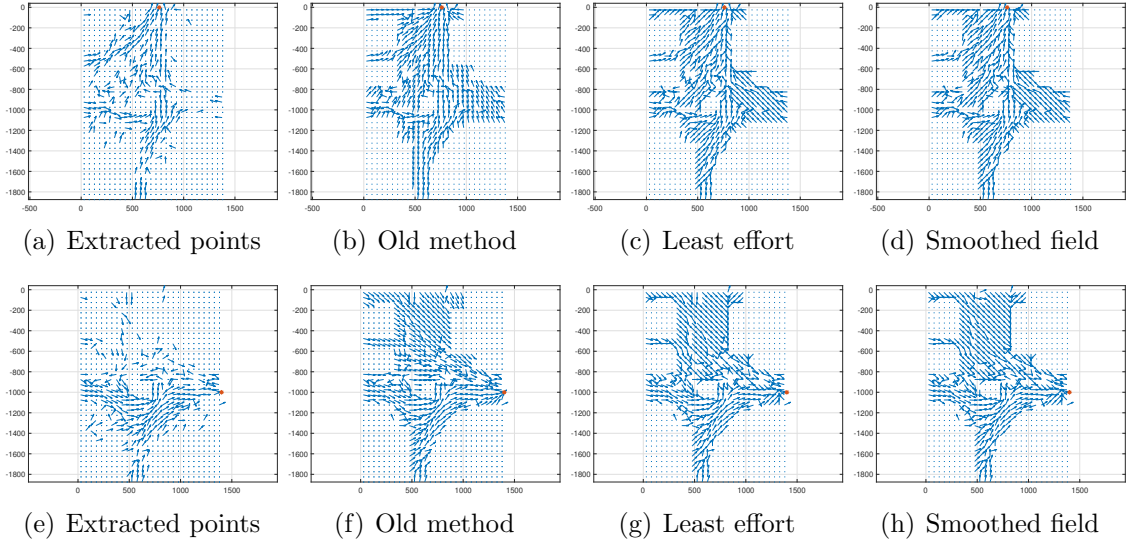


Figure 2.3: Different stages of the velocity field computation for the top exit (a,b,c,d) and the exit on the left side of the image (e,f,g,h). Exit points are represented by the orange stars. First, the velocity field is initialized only in the points where trajectories are extracted (a,e). The implementation in [167] fulfill the empty cells with vector pointing directly toward the exit (b,f). We initialize the empty cells using a least effort hypothesis(c,g) and in the refinement stage the field is smoothed (d,h).

euclidean distance between the k -th point along a trajectory and the (i, j) point of the grid; and R is a scaling factor:

$$\rho_{i,j} = \frac{1}{2\pi R^2} \sum_{k=1}^n \exp\left(-\frac{d(i,j,k)^2}{R^2}\right) \quad (2.10)$$

The results are shown in Fig. 2.4, where 2.4(b) represents the baseline method [167]; 2.4(c) represents the normal mask method; and 2.4(d) indicates the social mask method when people moving to the left exit of the scene. The smaller number of simulated trajectories results in smaller values of the densities. The normal mask method outperforms the baseline method in restoring the spatial density distribution, where the shape of the distribution and the locations of the peaks are consistent with the realistic distribution. In 2.4(d), our method shows more promising results when using a social mask.

2.2.5 Future work

The proposed framework can be further improved by taking into account different natures of agents since pedestrians and bikers should be treated in a different way. Moreover, using a different mask for each goal would also improve the distribution densities. The future work would focus on transferring learning from the typical velocity fields of common scenarios (like crossroads or roundabouts) to new environments. The main limitation of this work is that it still relies on empirically defined parameters to describe the high level behavior of the crowd.

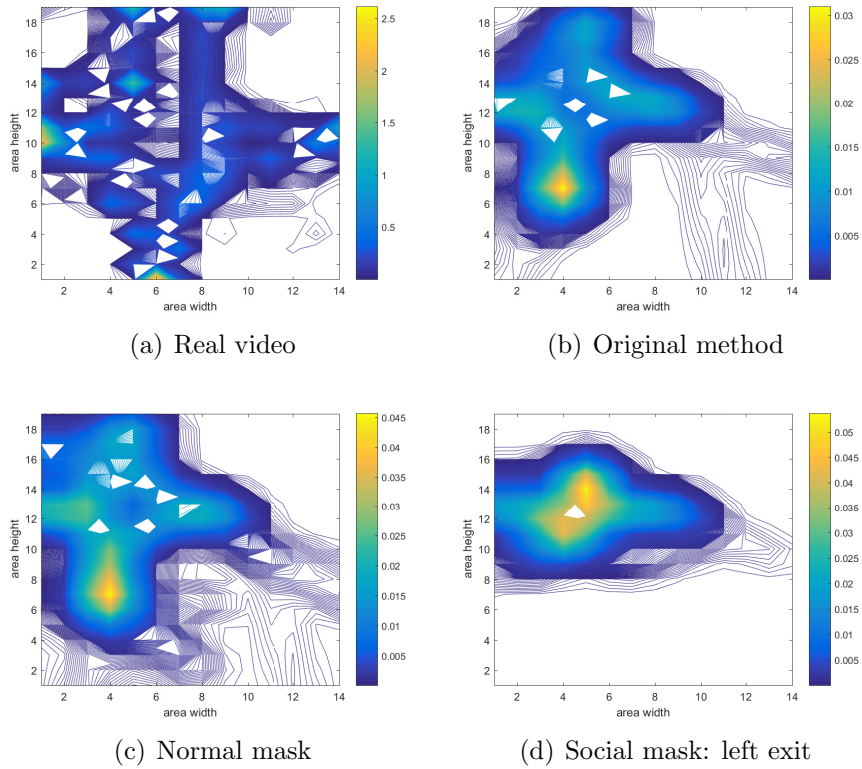


Figure 2.4: Crowd density distribution of the different simulations. In (a), the distribution of a crowd in a realistic video is represented. It can be noted the *normal mask* method (c) is able to better reproduce the densities distribution compared to the original method (b). Moreover, the *social mask* method shows promising results in modeling the density distribution of a single exit within the whole crowd (d).

2.3 Virtual Crowds: a simulation framework to model and observe crowded scenes

Social modeling of pedestrian dynamics is a key element to understand the behavior of crowded scenes. Existing crowd models like the Social Force Model and the Reciprocal Velocity Obstacle, traditionally rely on empirically-defined functions to characterize the **dynamics of a crowd**. On the other hand, frameworks based on deep learning, like the Social LSTM and the Social GAN, have proven their ability to predict pedestrians' trajectories without requiring a predefined mathematical model. In this work [23], we propose a new paradigm for crowd simulation based on a pool of LSTM networks. Each pedestrian is able to move independently and interact with the surrounding environment, given a starting point and a destination goal.

2.3.1 Introduction

The use of simulators in computer vision is not new per se, in particular for object tracking and camera control [122], and, in computer vision, we have recently observed an increasing interest in the generation of virtual videos instead of virtual agents behaviors. The substantial difference between the two consists of carrying out the analysis with standard computer vision techniques, using the videos generated by the simulator for testing purposes. This allows taking into account potentially all the challenges of a real video, including the presence of occlusions, obstacles, changes in the illumination conditions, the similarity of objects in the appearance model, etc.

In pedestrian dynamics, researchers have focused in the past on the study of trajectories and social interactions between agents. The Social Force Model (SFM) [64], the Reciprocal Velocity Obstacle model [147], and the continuum crowds model [146] are among the best existing empirical models for crowd simulation derived from observation of the real world.

Recently, deep learning has been applied to the prediction and forecasting of agents in a video. Recurrent Neural Networks (RNNs), and in particular Long-Short Term Memory (LSTM) networks, have successfully been adopted to predict the scene evolution over time [3]. This kind of network has shown the ability to learn the relation between spatially distributed data and its evolution. LSTMs has also proven to be capable of generating video sequences complying with predefined patterns [56]. Starting from the work in [3], we extend the use of LSTM networks from path prediction to a simulation of crowded video sequences. The proposed method is implemented via a recurrent deep neural network based on LSTM cells. Each agent in the simulation is driven by its own LSTM network, which is aware of the hidden state of neighboring agents. We train the LSTM on our synthetic dataset, showing how the network is effectively able to learn and replicate the simulated motion model. This demonstrates that gathering person-specific datasets of real subjects would open to the opportunity to learn person-specific behaviors, allowing for the inclusion of personal inclinations in the virtual crowd model for a more realistic simulation.

To prove the feasibility of our approach in simulating end-to-end trajectories, we validate our model using a synthetic dataset where agents move according to the SFM [64], demonstrating how the proposed framework can effectively learn the typical features of the SFM and reproduce them on different agents. The compliance

of the predicted paths against the SFM is measured using the spatial distribution metric introduced in [63].

2.3.2 Related work

The so-called Social Force Model relies on Newtonian forces to model the interaction between pedestrians and to guide each agent towards its goal. Similarly, in robotics, the Reciprocal Velocity Obstacle (RVO) [147] is widely used to model the collision avoidance behavior between agents.

In the work by Treuille et al. [146], the crowd is modeled as a fluid, where agents are influenced by their goal, position, their preferred speed, and a discomfort factor. However, the models mentioned above require the estimation or computation of parameters, in order to replicate the interaction among agents. The literature has tackled this issue relying on common optimization methods, like the Gradient-based Newton method [135] and Genetic Algorithms [117]. All these methods are able to describe crowd behaviors assuming the availability of a strong prior to the model. While they are good at describing the movements of the whole crowd, they often fail in correctly representing the agent’s personality, which is a key feature to enrich the model and make the simulation closer to the behavior of real subjects. Data-driven approaches have recently been employed to capture and model interactions among people in a crowd. Patil et al. [114] use vector fields, either learned or manually sketched, to guide the crowd flow in the environment. Goal-dependent velocity fields have also been used to guide the simulation at a global level [22]. Lerner et al. [87] guide the agents using learned trajectories, choosing the one that best matches the situation that the agent is facing. Empirically-defined models can only tweak the behavior of different agents by varying parameters such as personal distances and preferred velocities; data-driven simulations allow instead learning and simulating common social rules, such as the behavior at a roundabout, but local descriptors for each agent personality cannot be handled directly. Cognitive models have been employed to allow agents performing high-levels tasks such as path planning [53]. Various approaches have been employed to vary the pedestrian behavior in the same situation, such as the OCEAN model [46] and the general adaptation syndrome [17]. These models focus on the local shaping of the motion, based on personality traits, rather than long-term tasks.

As far as path prediction and activity forecasting is concerned, the literature reports a good amount of relevant works. The Social Force Model has shown good results in predicting pedestrian trajectories in the environment. Pellegrini et al. have proposed the Linear Trajectory Avoidance model [117], which predicts the short term path of a pedestrian. Interactive Gaussian processes have shown the ability to effectively improve the capability of a robot to predict pedestrian trajectories and navigate a crowded environment [145]. More recently, Recurrent Neural Networks, in particular, LSTMs, have proven good capabilities in describing spatially distributed data, also providing a temporal link in sequences. LSTMs have been applied to crowd trajectory prediction by Alahi et. al [3]. However, as mentioned above, one important issue in crowd simulation is the lack of a recognized evaluation metric to assess the quality of the simulation. If the simulation refers to an existing, real-life scenario, it can be evaluated using common tools, by measuring global features, such

as density and distribution of the elements in the scene. Density-based [88] and entropy-based metrics [59] are often used to compare real and synthetic data.

In [63], Helbing et al. present a metric based on the density distribution between two sets of trajectories. As detailed later, we will adopt this metric to evaluate the performances of our simulator in a quantitative manner.

2.3.3 The proposed model

Empirically-defined parametric functions are generally good in reproducing the global motion properties of the crowd, but they tend to fail when capturing the complexity that leads each human to react differently when facing the presence of other subjects or obstacles.

In the domain of path prediction, LSTM networks [66] have shown good capabilities in predicting the behavior of pedestrians, if properly trained. In particular, the Social LSTM model [3] is able to capture the status of the neighborhood around each agent (number and position of other agents), which is then used to refine the trajectory prediction. In our work, we propose a model able to perform a simulation of a crowded scene by learning the motion properties of the agents from a set of training video sequences. Each agent in the simulation is provided with an autonomous network, based on LSTM cells. Each pedestrian tries to reach its target destination given a starting point, a goal and the status of his neighborhood.

Depending on the set of trajectories used to train the specific network, each LSTM network can react in a different way to the same scenario. For example, if an LSTM network is trained with a set of trajectories belonging to a specific person, it will then be able to learn that model, thus replicating the subject reactions (personality traits) to specific situations occurring in the simulation. The possibility of reproducing different personality traits makes it possible to simulate semantically rich realistic situations, which do not rely on purely deterministic models. To the best of our knowledge, such paradigm has not been explored in the state-of-the-art and no dataset in this form has been made available to the research community.

Virtual Crowds

Extending the LSTM paradigm to crowd simulation, each agent’s movements are controlled by an LSTM network, able to guide it through the environment.

Agents (pedestrians in our scenario) moving in the crowd are influenced by their goal, their personality, and the state of their neighborhood. All the necessary details are provided in the next paragraphs.

Goal modeling

At the beginning of the simulation each agent/pedestrian ped_i , whose position is defined as (x_t^i, y_t^i) , is initialized at a starting position (x_0^i, y_0^i) . The goal of the i -th pedestrian ped_i is defined as

$$g^i = (x_g^i, y_g^i) \tag{2.11}$$

where x_g^i and y_g^i are coordinates defined in meters in the simulator.

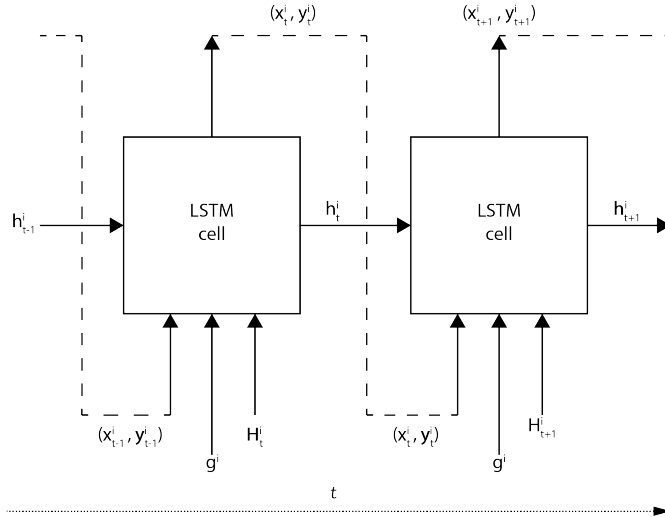


Figure 2.5: The figure represents the chain structure of the LSTM network between two consecutive time steps, t and $t + 1$. At each time step, the inputs of the LSTM cell representing a pedestrian are the previous position (x_{t-1}^i, y_{t-1}^i) , the goal g^i and the Social pooling tensor H_t^i . The output of the LSTM cell is the current position (x_t^i, y_t^i) .

The chain structure of a cell of the neural network is shown in Fig. 2.5. At each time step, the inputs of the LSTM cell representing a pedestrian are (i) the previous position (x_{t-1}^i, y_{t-1}^i) , (ii) the goal g^i , and (iii) the Social pooling tensor H_t^i . The output of the LSTM cell is the agent position (x_t^i, y_t^i) .

When the Euclidean distance between the current position (x_t^i, y_t^i) and the goal g^i is less than one meter, the simulation of that pedestrian is terminated.

Neighborhood modeling

The state of the neighborhood of each pedestrian is represented by a "Social" hidden-state tensor, as proposed by [3]. The Social pooling layer allows pedestrians to share their hidden states, thus enabling each network to predict the next position, reasoning about its hidden state and the neighboring state.

The pedestrian ped^j at time t in the scene is represented by the hidden-state h_t^j of a LSTM network. We choose the hidden-state dimension D and the neighborhood size N_0 . The neighborhood of the agent ped^i is handled by a "Social" pooling layer. This layer has the aim of pooling information received from the LSTM cells of its neighbors while preserving also their spatial mapping. This spatial mapping is preserved through a grid pooling as explained in Equation 2.13. The pooled information is used to build a tensor, called "Social" hidden-state tensor H_t^i , with dimensions $N_0 \times N_0 \times D$:

$$H_t^i(m, n, :) = \sum_{j \in N_i} 1_{mn} [x_t^j - x_t^i, y_t^j - y_t^i] h_{t-1}^j \quad (2.12)$$

where h_{t-1}^j represents the hidden-state of the LSTM of ped^j ($\forall j \neq i$) at $t - 1$,

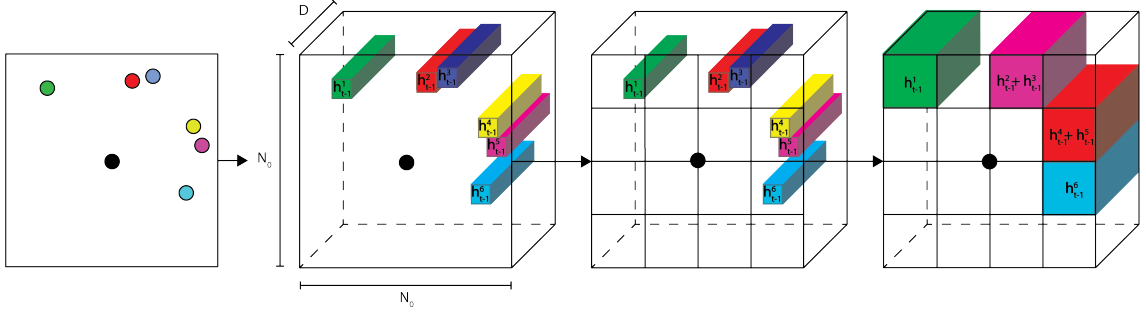


Figure 2.6: Representation of the Social pooling layer. The black dot represents the pedestrian of interest ped_i . Other pedestrians ped_j ($\forall j \neq i$) are shown in different color codes. The state of the neighborhood of ped_i is described by $N_0 \times N_0$ cells, which pooling together spatially-close neighbors preserves the spatial information. This will later be used to construct the hidden-state tensor H_t^i .

N^i represents the set of neighbors of pedestrian ped^i and $1_{mn}[x, y]$ is an indicator function defined as:

$$1_{mn}[x, y] = \begin{cases} 0 & \text{if } [x, y] \notin \text{cell } mn \\ 1 & \text{if } [x, y] \in \text{cell } mn \end{cases} \quad (2.13)$$

A graphical representation of the pooling operation is shown in Fig. 2.6.

Once computed, the Social hidden-state tensor is embedded into a vector a_t^i . The output coordinates are embedded in the vector e_t^i . The resulting recurrence is then defined by the following equations:

$$r_t^i = \Phi(x_t^i, y_t^i; W_r) \quad (2.14)$$

$$e_t^i = \Phi(a_t^i, H_t^i; g^i, W_e) \quad (2.15)$$

$$h_t^i = LSTM(h_{t-1}^i, e_t^i; W_l) \quad (2.16)$$

where Φ is an embedding function with ReLU activation; W_r and W_e represent the embedding weights, and W_l represents the LSTM weights.

The next position (x_{t+1}^i, y_{t+1}^i) in the simulation depends on the hidden-state at the previous time-step h_t^i . Inspired by [56], and as performed in [3], we predict the following parameters, which characterize a bivariate Gaussian distribution: the mean $\mu_{t+1}^i = (\mu_x, \mu_y)_{t+1}^i$, the standard deviation $\sigma_{t+1}^i = (\sigma_x, \sigma_y)_{t+1}^i$ and the correlation coefficient ρ_{t+1}^i . We use a $5 \times D$ weight matrix W_p to estimate the parameters. Thus, the coordinates at the next time-step $t + 1$ are computed as:

$$(x_{t+1}^i, y_{t+1}^i) \sim N(\mu_t^i, \sigma_t^i, \rho_t^i) \quad (2.17)$$

In order to estimate the parameters of the LSTM model, the negative log-Likelihood loss L^i for agent ped^i is minimized for the current time instant t :

$$[\mu_t^i, \sigma_t^i, \rho_t^i] = W_p h_{t-1}^i \quad (2.18)$$

$$L^i(W_e, W_l, W_p) = - \sum_{t=T_{cur}+1}^{T_{step}} \log \left(\mathbb{P}(x_t^i, y_t^i \mid \mu_t^i, \sigma_t^i, \rho_t^i) \right) \quad (2.19)$$

Our model is trained minimizing the log-Likelihood loss for all the trajectories belonging to the dataset.

Personality

As far as the personality and decision making is concerned, the motion of the pedestrian in the crowd has been modeled in existing simulators by empirically varying parameters, such as pedestrian velocities and personal distances. Although this could actually increase the diversity of the agents’ motion, it is still not sufficient for a realistic modeling of the personality traits, which in general lead to non-deterministic and non-linear motion models. Providing a ”personal” set of trajectories to each agent in the scene would allow for much more complex and diverse simulation.

Let us consider two pedestrians in the real world, where one exhibits an *aggressive* behavior, while the other exhibits a shy behavior [17]. Let us assume that we are able to collect a significant set of trajectories for each of the two pedestrians. During the simulation, we associate ped_i as the aggressive subject, and ped_j as the shy one. In this way, if we introduce both of them in the same simulated scenario, such that the social pooling tensor is $H_t^i(m, n, :) = H_t^j(m, n, :)$, and goals are $g^i = g^j$, the two pedestrians will make different decisions, depending on the specific features.

2.3.4 Implementation details

To achieve our results, we have used the following configuration. The embedding dimension for the spatial coordinates is set to 64. The spatial pooling size, which corresponds to an area of $4 \times 4 m^2$, is set to 32. The pooling operation is performed using a sum pooling window of size 8×8 with no overlaps. The hidden state dimension is 128. The learning rate parameter is set to 0.003 and RMS-prop [42] is used as the optimizer. The model is trained on a single GPU using a PyTorch¹ implementation.

At the input of the LSTM, each trajectory consists of a set of coordinates (X_{real}, Y_{real}) in meters, which needs to be normalized. Each pair (x_{real}, y_{real}) in the set is normalized between $[-1, 1]$ using the following conversion:

$$(x_{norm}, y_{norm}) = \left(2 * \frac{x_{real} - \min(X_{real})}{Norm} - 1, 2 * \frac{y_{real} - \min(Y_{real})}{Norm} - 1 \right) \quad (2.20)$$

where $Norm$ represents the maximum range (in meters) of the biggest scene. This normalization is in line with the experiments conducted in previous works in this area [66].

¹<http://pytorch.org>

2.3.5 Results

Dataset

To test our approach, we created a dataset composed of 5 fully annotated scenes, which visually match other known real datasets, namely the ETH [117] and UCY dataset [87], used in the experiments in [3]. An example is shown in Fig. 2.7. The choice of using real datasets as reference is to confront the agents' motion to scenes known to the community, rather than reproducing all the very fine details in terms of appearance, which is in this work not relevant.

The average density of pedestrians per frame is 10. The five different scenes include 204, 177, 194, 210, and 174 trajectories, respectively. The videos have been recorded at a rate of 15fps. The preferred velocity in the Social Forces model is set to $4.7km/h$, which corresponds to the average speed of a walking pedestrian.

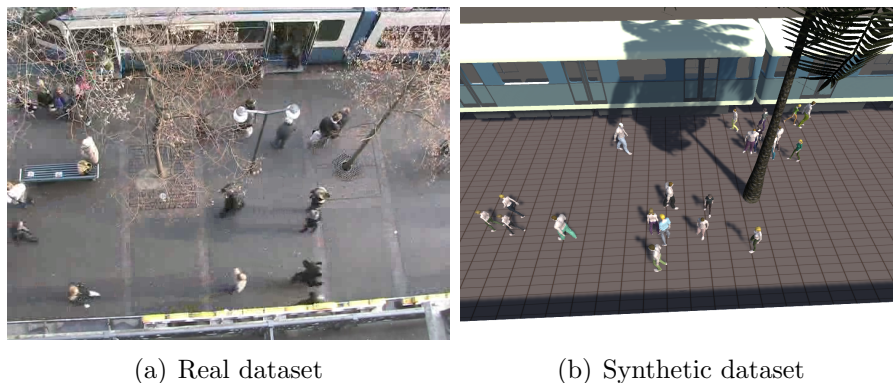


Figure 2.7: A comparison of two frames of the real and synthetic datasets. The scene of interest is the so-called Hotel scene of the ETH dataset [117].

Experiments

To demonstrate the learning capabilities of the proposed framework, we rely on the simulated sequences based on SFM. Each LSTM network is trained for 100 epochs. Both training and test are conducted according to the leave-one-out strategy. This corresponds to training on 4 sets of trajectories and simulates a scene, which is similar to the fifth one. The prediction is updated at steps of 0.4 seconds. In other words, the agent will re-consider its next displacement according to its surroundings after 6 frames.

For each agent, the simulation starts in the time instant it appears in the scene. Each pedestrian is assigned a goal, namely the last known position in the scene. Our model computes the trajectories for each pedestrian and returns the set of positions for all of them at each time step, for a complete and exhaustive representation of the scene content.

	n	i	j
ETH Univ	356	4	25
ETH Hotel	383	4	18
UCY Zara1	455	25	14
UCY Zara2	654	25	15
UCY Univ	3006	30	18

Table 2.1: For each video sequence, the table reports the number of points available in the simulated video, and the number of grid cells in the horizontal (i) and vertical (j) dimensions.

Validation

To evaluate the simulation of the model with the dataset trajectories, we compare the density distribution $\rho_{i,j}$. The density of the distribution is computed using the equation suggested by Helbing et al. [63], as shown in Eq. 2.21, where n represents the total number of points of all trajectories; $d(i, j, k)$ is the Euclidean distance between the k -th point along a trajectory and the (i, j) point of the grid; and R is a scaling factor:

$$\rho_{i,j} = \frac{1}{2\pi R^2} \sum_{k=1}^n \exp\left(-\frac{d(i, j, k)^2}{R^2}\right) \quad (2.21)$$

The accumulation points (i, j) are displaced in a grid that covers the whole image with a distance of 1 meter between them. The real dimensions of each scene, grid size (i, j) , and the number of total points n are shown in Table 2.1. The scaling factor is equal to 10. The plot of the density distribution of a set of trajectories for each dataset is shown in Table 2.2.

2.3.6 Future work

In future works, we plan to incorporate an Ego-Vision framework such that each pedestrian navigates the environment according to the information gathered from a first-person perspective. This would also allow us to better extract personality traits since the first person view contains many more fine-grained details about the person wearing the camera.

2.4 Virtual camera modeling for multi-view simulation of surveillance scenes

A recent trend in research is to leverage on advanced simulation frameworks for the implementation and validation of video surveillance and ambient intelligence algorithms. However, in order to guarantee a seamless transferability between the virtual and real worlds, the simulator is required to represent the real-world target scenario in the best way possible. This includes, on the one hand, the **appearance modeling** of the scene and the motion of objects, and, on the other hand, it should be accurate with respect to the sensing equipment that will be used in the acquisition

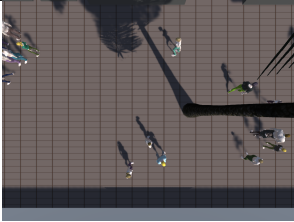
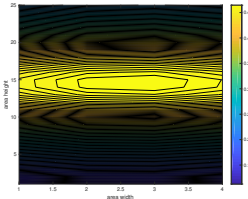
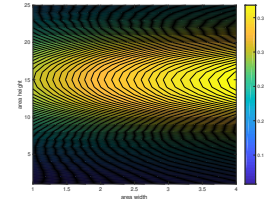
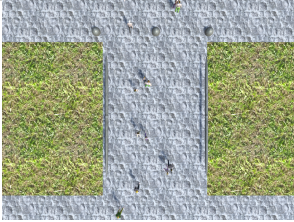
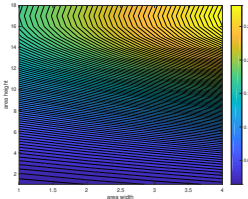
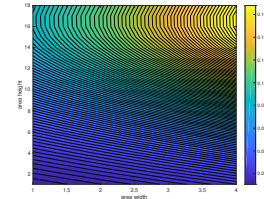
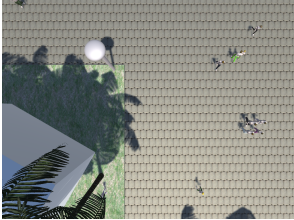
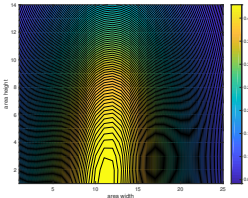
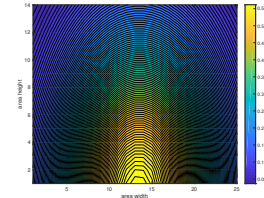

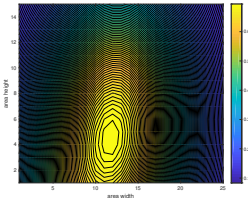
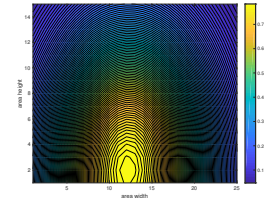

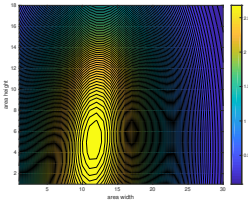
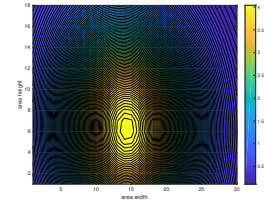
Name	Scene	Original Density	Simulated Density
ETH Hotel			
ETH Univ			
UCY Univ			
UCY Zara1			
UCY Zara2			

Table 2.2: Simulation of the five different video scenes that mimic real datasets, as used in our experiments. The left column reports an overview of the scene, while the central and right columns report an example of the original video density and a sample of the one simulated using our framework. The value of ρ is color-coded: warmer colors represent higher densities, while colder colors represent lower values. Numerical values are reported on the right column at the right of each plot.

phase. Originally published in [20], this work focuses on the latter problem related to camera modeling and control, discussing how noise and distortions can be handled, and implementing an engine for camera motion control in terms of pan, tilt, and zoom, with particular attention to the video surveillance scenario.

2.4.1 Introduction

Video surveillance has been a matter of study for the past three decades, and researchers have investigated many different facets of the subject, ranging from simple motion detection and segmentation algorithm [8, 19], to object tracking [2], person re-identification [109], and analysis of complex crowded scenes [73]. Although most video surveillance networks rely on the use of ordinary static cameras, there is an increasing trend in updating the existing infrastructures with smart cooperative networks of cameras. In such a scenario, cameras are required to share relevant information over the network, in order to improve the tracking of objects and provide the best possible coverage [79]. To guarantee flexibility and dynamic reconfiguration of the camera network, a viable option is to adopt PTZ (Pan-Tilt and Zoom) cameras, which allow tracking and focus on specific objects of interest in the scene thanks to the possibility of dynamically repositioning the sensor [74].

However, researchers in this domain keep facing two common problems: (i) the lack of labeled data, especially for those rare events, i.e., anomalies, that should be detected by the monitoring infrastructure, and (ii) the incapability of reproducing the same type of event when dealing with reconfigurable camera networks. One possible solution to tackle such limitations is to deploy virtual environments and simulation frameworks. Virtualization has been subject of research in the camera networks community [123, 142] and crowd analysis [73].

Most papers in the state of the art adopting simulation, have focused on the deployment and assessment of different network configurations to guarantee good coverage of the scene thus improving the chances of detecting critical events. However, there is no sufficient literature that demonstrated the transferability of the lessons learned from the simulated environment into the real world. In fact, it is to be noted that when a sequence is recorded using a virtual framework, we must also capture the peculiarities of the specific sensor used for the acquisition. Besides the actual modeling of the simulated environment, also the camera modeling has to be representative of the real equipment, being able to model multiple features, as for example the noise sources and distortions of real cameras and lenses.

In the computer vision field, advanced graphical simulation frameworks are being exploited to perform data augmentation [127]. To our knowledge, the rendering and visual appearance of the scenes has been studied and developed rather thoroughly, while the peculiarities of the virtual recording system have not been deeply investigated. Camera models implemented in modern computer graphics engines aim at producing contents that enhances user quality of experience [28], rather than producing realistic (noisy) images.

To correctly model a camera network in a virtual environment we need to deal with the camera intrinsic and extrinsic parameters. We also need to model different kinds of noise sources and distortions, which are typical of real cameras.

Extending the model to include PTZ cameras, also requires to model the camera

motion, which significantly impacts the types of algorithms that can be used for the analysis, in terms of object detection and tracking, since common background subtraction techniques would be impaired by the apparent motion of the background.

In the next sections, we present a framework for modeling set of parameters and distortions related to lenses and camera sensors. We then focus on the specific case of a PTZ camera, showing how to deal with the challenges posed by the camera motion. Eventually, we present a use case scenario for the developed camera model, deployed in a simulated camera network.

2.4.2 Camera model description

As mentioned above, modeling an acquisition system requires taking into account two main components, namely the lens and the image sensor. In the real world, these elements are sources of noise and distortion, making the acquisition process significantly different from the theoretical ideal model. In this section we propose a virtual camera model able to deal with the noise and distortions existing in real devices, and also handling the needs in terms of camera motion and control.

Lenses

In optics, a lens is a refractive device, which either focuses or disperses light beams. In order to capture an image, an ideal lens focuses all the captured light on a single point and is characterized by a certain numbers of parameters, as:

- focal length
- field of view (FOV)
- depth of field
- aperture

In our simulation framework we focus on modeling the following distortions:

- chromatic aberration
- radial distortion

Focal length and Field of View

The focal length of a lens is a measure of how the incoming light is either diverted or converged. The bigger the focal length, the higher will be the magnification of an object. The relation between the focal length and the magnification of the object is ruled by the *thin lens* equation:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \quad (2.22)$$

where f is the focal length, u is the distance between the lens and the object and v is the distance between the focal length and the image plane.

The angular field of view (AFOV) is defined as the maximum angular size of an object of interest that can be captured by the camera. The object under inspection is supposed to be at an infinite distance from the lens:

$$AFOV(^{\circ}) = 2 * \tan^{-1}\left(\frac{h}{2f}\right) \quad (2.23)$$

where h is the horizontal sensor size and f is the focal length in millimeters. Common simulation tools natively offer the FOV as a parameter to be set.

The described model is embedded in our simulation framework to re-project the acquired environment onto the image plane.

Aperture

the aperture of a camera regulates the amount of light reaching the image sensor. The aperture size is usually regulated by a device called the diaphragm, which increases or decreases the aperture size at a factor of two aperture area per stop. The f -number (N) of a camera lens corresponds to the ratio between the focal length f and the diameter D of the aperture:

$$N = \frac{f}{D} \quad (2.24)$$

When modeling the aperture in synthetic images, we need to take into account that a smaller value of N causes a wider aperture size (we are allowing more lights to reach the sensor). A higher value of f causes the camera aperture to become narrower, thus allowing less light to reach the sensor.

Depth of Field

The depth of field is defined as the distance between the nearest and the further object, located in the zone of acceptable sharpness in a photo. The depth of field is determined by three main factors: the focal length, the distance of the object from the camera, and the aperture.

The f -number controls how wide the depth of field will be around the subject that the camera is capturing. The lower the value, the shallower the total depth of field being captured; the higher the value, the wider the total depth of field.

To model the depth of field, we use the full derivation of formulas presented in [124].

Chromatic aberration

Chromatic aberration is the effect caused by the inability of the lens to focus all the different colors in the same point, as shown in Fig. 2.8.

In order to simulate the chromatic aberration effect, we then need to slightly separate the color spectrum at the edges and corners of the image.

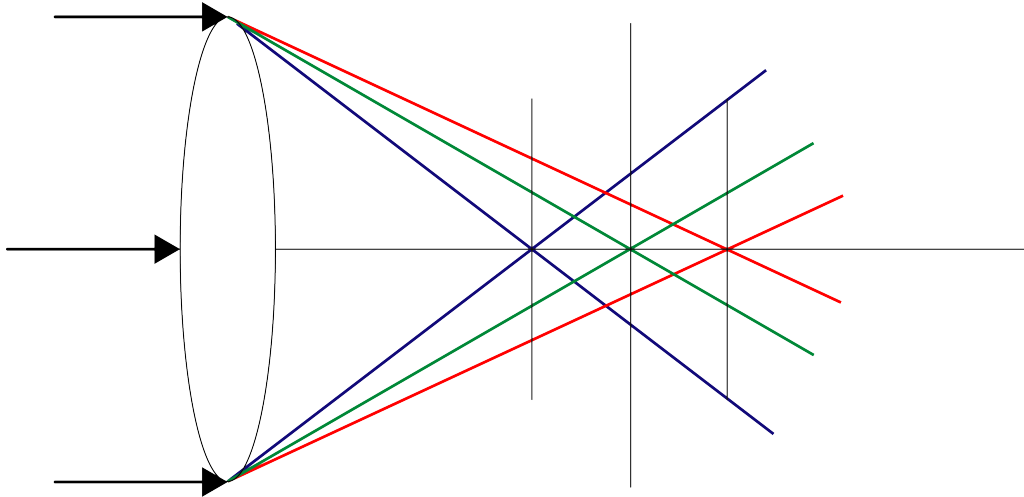


Figure 2.8: Chromatic aberration depends on the lens inability to focus the entire color range in the same spot.

Lens radial distortion

Common lens distortions present or can be approximated as having symmetries along the radial axis. They are usually classified in three different classes: barrel distortion, pincushion distortion, and a combination of the previous two, the so-called mustache distortion.

Barrel distortion, as shown in Fig. 2.9.a, is characterized by the decrease of the object magnification as the distance from the optical axis increases. This distortion is sometimes intentionally used to obtain the so-called fish-eye effect.

Pincushion distortion, as shown in Fig. 2.9.b, is characterized by the increase of the object magnification as the distance from the optical axis decreases.

Mustache distortion, as shown in Fig. 2.9.c, presents a mixture of the two previous distortions. A barrel-like distortion is present toward the center of the image and it becomes a pincushion-like distortion as the distance from the radial axis increases.

The mathematical formulation to correct those distortions is the Brown-Conrady model [40, 30], which has been used in our implementation to test and correct the simulation model we have implemented.

Camera sensor

The sensor is the element in a camera, which transforms the incoming light rays into an electrical signal. The signal is represented in our case by an image. Since we are trying to reproduce a digital image, we have to understand how to reproduce the characteristics and distortions of a digital sensor. In a camera, the lens and the sensor influence parameters such as the field of view. If the sensor is too small to capture all the light conveyed by the lens, the effective field of view is determined by the sensor. If the captured light does not fill the whole sensor area, the effective field of view is determined by the lens. The image sensor format of a digital camera determines the angle of view of a particular lens when used with a particular sensor.

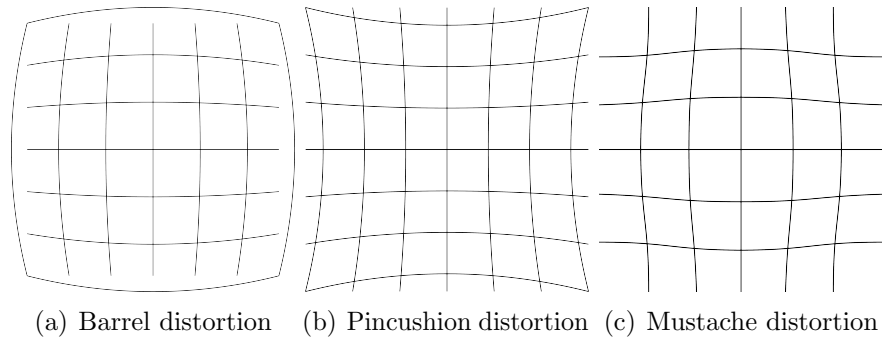


Figure 2.9: Common radial distortions patterns in real lenses.

The fundamental elements that characterize a sensor are:

- sensor size (format)
- resolution
- dynamic range
- camera sensor noise

Sensor size

The sensor size (or sensor format) indicates the shape and the size of the sensor capturing the light. It determines how much light will be used to produce the final image. It determines the final size and format of image that can be captured with a camera device. The size of sensor ultimately determines how much light it uses to create an image. Increasing the sensor size causes the depth of field to decrease, aperture being fixed.

Resolution

Resolution of a camera is the ability to distinguish details in the image. It is usually limited by the lens diffraction and by the sensor resolution.

Optical resolution describes the ability of an imaging system to resolve detail in the object that is being imaged. Resolution is usually measured in pixels.

In the simulation framework we are able to manually set the resolution and resize the image as needed.

Dynamic Range

The dynamic range is defined as

$$DR = \log \frac{N_{max}}{N_f} \quad (2.25)$$

where N_{max} represents the maximum signal level that the sensor can output, and N_f represents the noise floor at minimum amplification. The noise floor is calculated

as the root mean square of the noise level in a black image. The dynamic range measures the capability of a sensor to capture the brightest and darkest spot in an image and the number of levels in between.

Modeling the dynamic range of a sensor in a simulated environment is achieved starting from the color distribution of an object in a scene and applying some contrast stretch techniques, as also commonly used in photography. Contrast stretching is a technique, which aims at improving/modifying the quality of an image by stretching or compressing the intensity value of the different colors, such that it fits the desired interval of values. In our methodology we are able to set the lower and the upper limit of the stretched histogram, in order to fit the color values of our virtual camera to the one of a real device. Contrast stretch adaptation speed also allows reproducing common camera effects in videos when there is a sudden change in lighting.

Camera sensor noise

Ideally, the camera sensor should produce exactly one electron for each photon striking one of his pixels. In practice, the process, which allows the camera sensor to convert light into a proper image is affected by noise. In captured images, noise can be seen as a granular color variation on surfaces, which look uniform at a distance.

In [69], noise is segmented into spatio-temporal categories to be measured. The final noisy image N_{cap} is defined as

$$N_{cap} = (I * PRNU + SN_{ph}(I) + FPN + SN_{dark} + N_{read}) * N_D * N_{filt} + N_Q \quad (2.26)$$

where I is the sensor irradiance, PRNU is the photo response non-uniformity, SN_{ph} is the photon shot noise, FPN is the offset fixed-pattern noise, SN_{dark} is the dark-current shot noise, N_{read} is the readout noise, N_D is the demosaicing noise, N_{filt} is the post image capture effect, and N_Q is the quantization noise.

The distribution of all sources at a given CCD-sampling frequency is measured as an additive Gaussian distribution.

For the simulation, we are interested in reproducing the noise intensity and overall distribution rather than exactly calibrating the model to reproduce a specific camera brand or type. This is achieved by adding a white Gaussian noise, which can be modified in terms of mean value and standard deviation to fit the requirements at hand. It allows the simulation of typical scenarios, such as the noise in low light conditions and bloom borders.

2.4.3 Pan-Tilt-Zoom Camera

In visual surveillance, the use of PTZ (Pan-Tilt and Zoom) camera has been thoroughly investigated [123, 142, 8]. PTZ cameras provide an effective way to increase the coverage of a certain area thanks to their ability to move, either by progressively scanning the environment or zooming in to specific locations in presence of events of interest. In a cooperative camera network, PTZ cameras have to be able to dynamically detect and track the objects of interest [8, 74, 19], guaranteeing a smooth handover across cameras.

Therefore, when modeling such cameras, we must deal with the motion parametriza-

tion of each camera in the network, thus acting on both the intrinsic and extrinsic camera parameters.

PTZ motion model

In this section we describe the model used to replicate the movement of a PTZ camera in a virtual environment. Different types of cameras are available on the market, like mechanical PTZ and virtual PTZ. The term virtual implies that no physical sensor movement occurs; the captured images, instead, are obtained by cropping from the full resolution picture obtained by a high-resolution image. Generally, to model the motion of a PTZ camera we need to replicate its three extrinsic parameters (pan, tilt and zoom), assuming that the camera is anchored to a fixed location.

Also, pan and tilt are subject to constraints, and the step size for the variation must be defined to control the velocity response of the camera at each time step. Dynamically changing the Field of View of the virtual PTZ allows for zoom modeling, along with providing a maximum and minimum range for the FOV to vary.

PTZ tracking algorithms

The development of a real-time object tracking algorithm to be applied on a PTZ camera must tackle a variety of problems, such as camera movement, complex object motion, presence of other moving objects in the video scene, and real-time processing requirements.

Algorithms satisfying these constraints can be divided into two main classes: the ones relying on background segmentation [8, 74], and the ones that allow to track only specific objects classes [19].

Algorithms exploiting the background segmentation have to cope with the constant camera movement, requiring re-initialization every time a camera displacement is triggered.

Other algorithms can track even in the presence of camera motion, relying on common vision [118], and machine learning [2] tools. However, the main drawback is the difficulty in dealing with the real-time constraints and the related computational costs, which is addressed in literature by either tracking only a category of objects [2] or by developing customized hardware [19].

In our test application, we exploit a simple background segmentation application to detect the object, which is then tracked using a state of the art appearance-based tracking, namely the cam-shift algorithm [29], as shown in Fig. 2.10.

2.4.4 Results

To validate the simulation of the focal length, we use the Camera Calibration Toolbox [27]. From the conducted experiments we noticed that the simulated focal length differs from the ground truth in millimeters of an average error of 4%.

We provide samples of computer-generated images which have been recorded using our camera model. In Fig. 2.11 we show an example of the variation of the focal length on a sample virtual image, without moving the camera. As can be seen, the field of view angle decreases as the focal length increases.

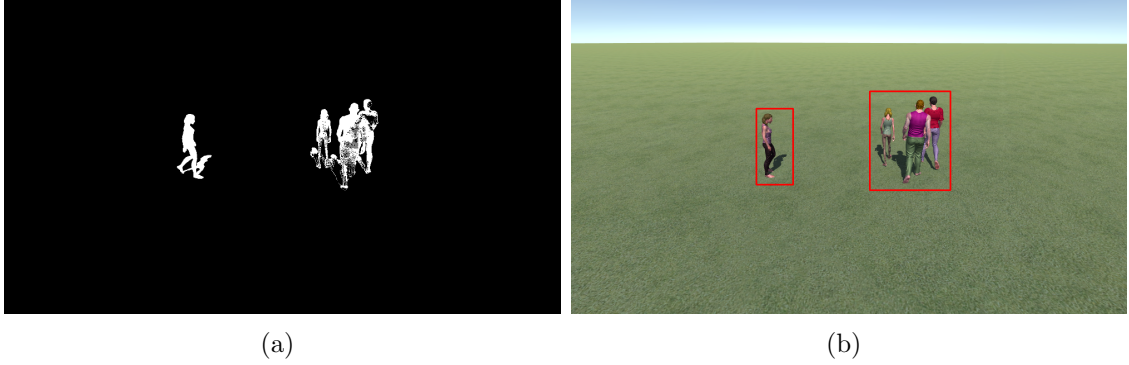


Figure 2.10: On the original image we perform the background subtraction (a) and blob detection (b). The histogram computed on the blob is then provided as input to the cam-shift algorithm, which is able to track the target even in presence of camera motion.

In Fig. 2.12, we show a noise pattern that increases as a multiplication factor. It is also possible to vary the noise pattern depending on the channel of interest. As can be seen, the noise pattern looks comparable with the noise generated by camera sensors in the presence of low illumination, introducing artifacts and color aberrations.

In Fig. 2.13, we show different examples of distortions. To validate the distortions applied, we applied rectification to restore the undistorted images according to [165].

In Fig. 2.14, we show the effect of different values of contrast stretching in the images.

Similarly to all the other elements that characterize a camera, the developed model also takes into account the depth of field, which is a crucial parameter that can alter the perception of objects in a visual scene, in terms of sharpness and level of detail. A sample view to show the capabilities of handling the depth of field is shown in Fig. 2.15.

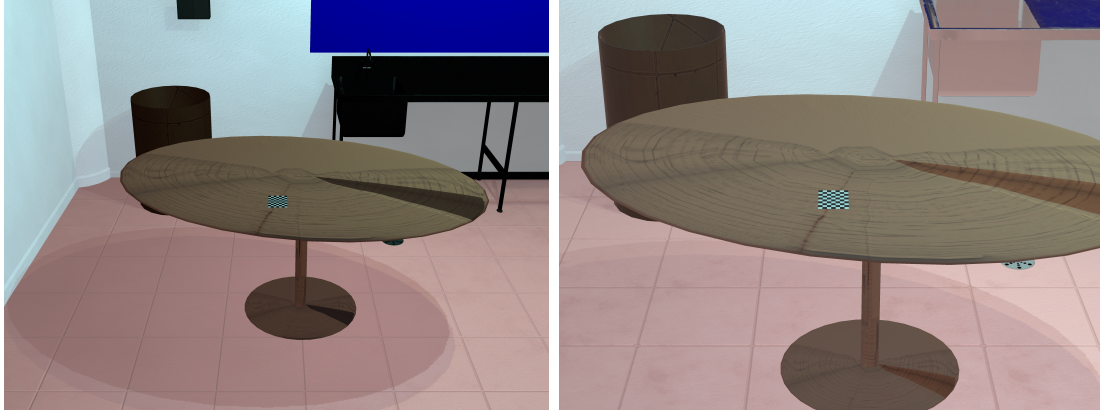
2.4.5 Future work

We presented a framework for modeling camera modeling in a simulation framework, highlighting the need of properly handling the issues of noise and distortions. Future work will focus on the development of a full framework to allow the testing and evaluation of smart camera networks algorithm, allowing researchers to test their own solutions, for tracking, layout optimization, and cooperation among cameras.

2.5 Conclusions

In this chapter, we proposed two frameworks for modeling the crowd dynamics and an approach to the appearance modeling of rendered scenes.

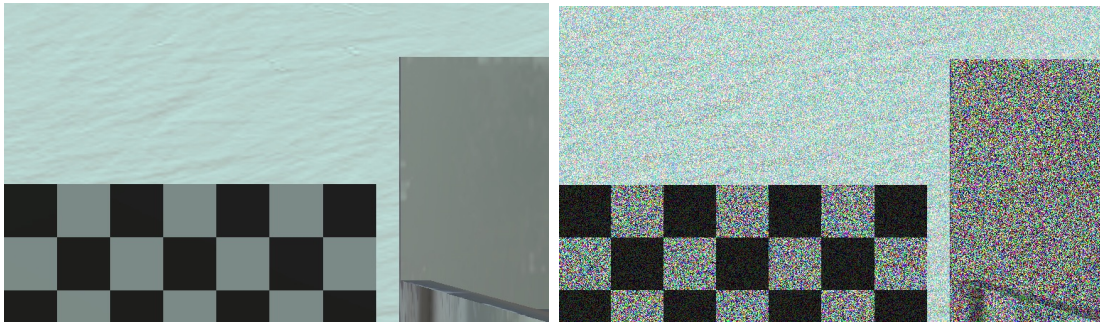
First, we described our data-driven crowd simulation framework, where only a small number of trajectories are available in realistic scenarios. Based on the least-effort hypothesis and by manually selecting the walkable areas, the crowd dynamics extracted from the realistic scenarios are integrated to compute the so-called *velocity*



(a) FL: 25 mm, FOV: 48.1°

(b) FL: 40 mm, FOV: 31.2°

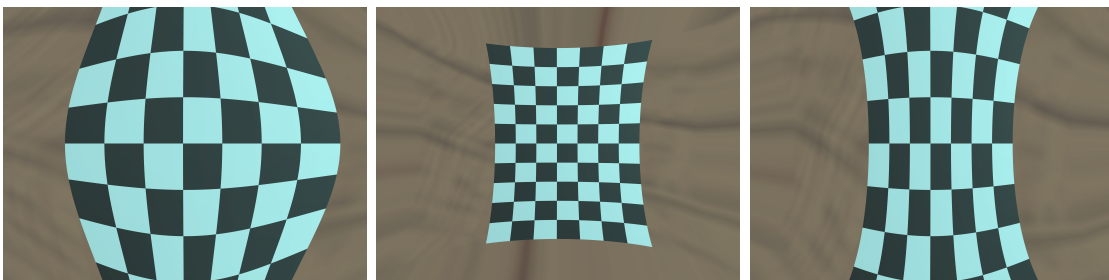
Figure 2.11: Example of images taken by a fixed camera (virtual) with an increasing focal length. FL corresponds to the focal length and FOV is the field of view angle.



(a)

(b)

Figure 2.12: Detail of a noiseless (a) and noisy (b) image affected by the camera sensor noise.



(a) Barrel distortion

(b) Pincushion distortion

(c) Mustache distortion

Figure 2.13: Example of radial distortions applied in the simulation framework.

fields, which can be used to navigate the macroscopic crowd behaviors.

Second, we presented a deep-learning based method to model crowd dynamics. The model assigns a neural network to each agent in the simulation. The pedestrians are connected by a social-pooling layer, which allows them to be aware of the status of

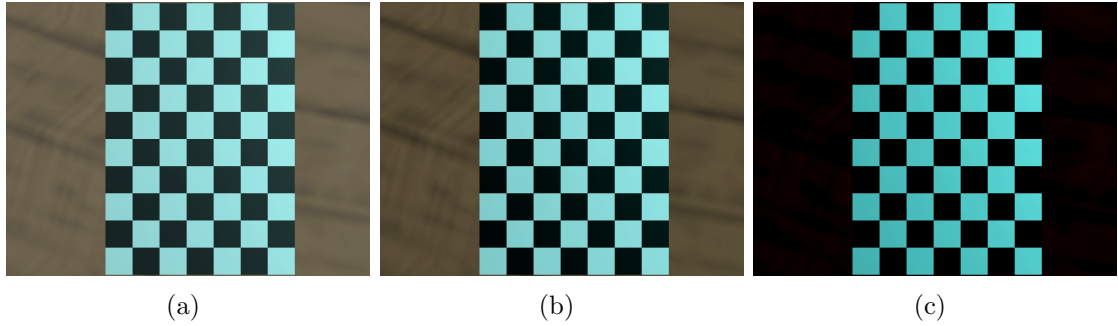


Figure 2.14: Modifying the minimum and maximum value of color stretch it is possible to change the appearance of the scene.

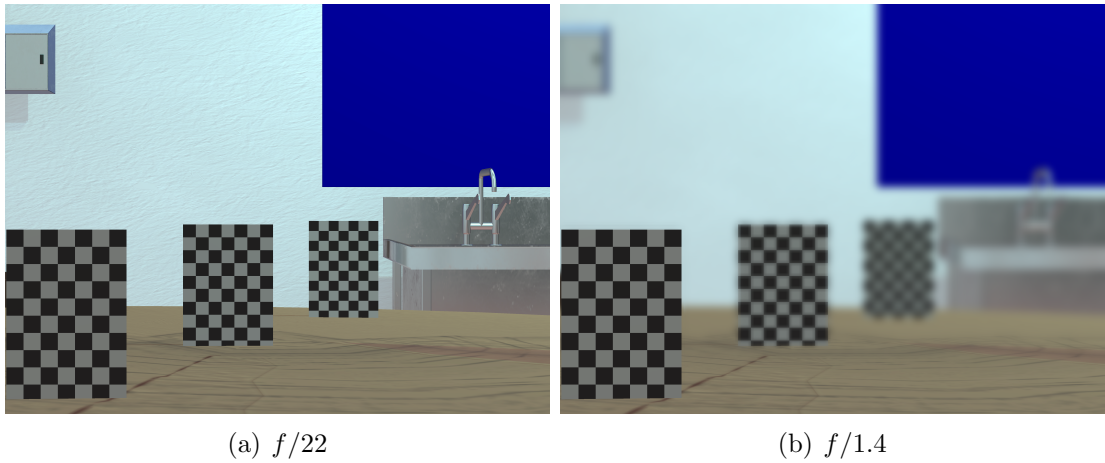


Figure 2.15: Examples of synthetic depth of field effect. The images are captured using a fixed camera and varying the aperture. The f -number is reported for each image. The setting consists of a fixed camera with 3 checkerboards at a distance of 1, 1.5, and 2 meters from the camera.

their neighborhood. Compared to empirically-defined functions used in the literature to model crowds, the proposed solution allows simulating a more complex and enriched variety of behaviors. Training the network with different datasets allows in fact to reproduce the behavior of agents with different personalities. As stated in the beginning, the proposed method does not have the goal of improving the state of the art but proposing a novel approach based on a simulator. Due to this reason, there is no comparison with any state of the art method for trajectories, since, to the best of our knowledge, there are no other simulators in literature which we can compare to.

Finally, we showed how smart camera networks and PTZ cameras research would benefit from the application of the virtual vision paradigm, relying on sophisticated 3D engines to replicated real challenges in the synthetic domain.

2.5.1 Open Issues

How similar is the virtual scene compared to the original one? How close should they be?

The development of simulators has been carried out by researchers in both computer vision [77, 91] and 3D graphics fields [116] with different purposes. In computer vision, the focus is to reproduce the main features of the crowd that are subject to analysis, such as trajectories, optical flow, density, the behavior changes in an emergency scenario, and emergent behavior. To achieve this purpose, it is not necessary to model and render pedestrians in a realistic fashion in the simulation environment, and pedestrians are often represented as points or cylinders, while the environment can be rendered using uniform surfaces. In computer graphics, the main focus is on reproducing the realistic visual appearance of the crowd. To reach this goal, pedestrians are modeled and animated in 3D in a realistic way, but the social interaction model is usually less sophisticated with respect to the Computer Vision field. Thus, if the interest is to carry on a research on crowd features, such as the crowd main flow or collision avoidance behavior, it is not strictly necessary that the simulator is able to reproduce the crowd appearance, since the output of the simulator is compared on a feature-level with the real results. For example, if one wants to estimate the crowd density in a video in which every pedestrian occupies just few pixels, there is no need to model that pedestrians in an accurate way in the simulated environment, but the most important thing is the modeling of characteristics such as occlusions and trajectories rather than the realistic appearance of each individual. On the other side, if the simulator is meant to output test sequences for video analysis, the visual appearance of the video is of a great importance. The synthetic video should be as similar as possible to real ones, even reproducing the typical distortion of a video recorded in a real environment, such as poor or variable light condition, camera's poor resolution/frame-rate and partial occlusions of the scene. The ideal output would be an indistinguishable synthetic video with respect to a real one, but a good output would be a video which, although it does not seem "real" to human observer, it shows similar features to be analyzed by crowd analysis algorithm, such as trackers, crowd segmentation algorithms, and crowd counting.

Validation of the simulation model.

The validation of the social interaction model has to be performed before using the output of the simulator as reliable test sequences. The task consists of checking how well the model is able to reproduce behaviors seen in real-world crowds, such as well-known emergent behaviors and pattern flows. In general, validation is done comparing the synthetic results to hand-processed video sequences. Low-level feature comparison, such as path by path or position, does not carry much information since it would only lead to a simulation perfectly fitting a real scene. A common instrument of validation is the comparison between the global path patterns of real and simulated scenes, which also carries information about the distribution of the agent in the scenes. The validation of local flow patterns has to be accurately monitored over time since there could be a high variation of common flow patterns related to particular events over time (e.g. a train station has high changes of distribution over time depending

on which platform a train is arriving/leaving). In general, analysis algorithms have to be run both on real and simulated videos. Depending on the specific crowd behavior, such as tracking individuals in the crowd, density estimation or pattern flow studies, the proper analysis algorithm has to be tested on the videos. If the results are similar, the simulator is validated. If the simulator is validated, it can be assumed that videos of different scenes of the same crowd behavior from multiple viewpoints are representative of how real videos would look like. This would allow the researcher to create a diverse and complete dataset for testing for each specific crowd behavior, with the possibility to record the simulation from every required viewpoint.

3 Crowd Simulation Application: Surveillance

Automatic crowd surveillance will play a fundamental role in the coming generation of video surveillance systems, in particular for improving public safety and security. However, traditional camera networks are mostly not able to closely survey the entire monitoring area due to limitations in coverage, resolution and analytics performance. A smart camera network, on the other hand, offers the ability to reconfigure the sensing infrastructure by incorporating active devices such as pan-tilt-zoom (PTZ) cameras and UAV-based cameras.

In camera network research, virtual vision for testing and deploying network has long been a subject of research [123, 142]. Recent improvements in computer graphics have expanded the possibilities of using game engines to tackle computer vision tasks [127]. Deploying a network of cameras in a virtual environment would provide researchers with a valid testbed for validation and benchmarking purposes. Tracking algorithms can benefit from the ground truth knowledge, which does not need to be manually annotated, and different tracking algorithms can be tested on the very same scene. Also, re-identification and tracking of subjects across cameras is another area in which the data recorded in the simulated environment can be effectively exploited. Besides effectively tracking an object, a network should be able to optimize the coverage of the environment. Optimization of camera deployment is the first step to guarantee the best camera displacement [79]. In the case of PTZ cameras, at running time cameras must be able to correctly perform hand-offs. While a camera is focused on tracking a target, the other cameras should be able to reconfigure, so as to guarantee the maximum coverage of the space of interest. We focus on the development of such validation frameworks, tackling the problem of camera modeling in terms of distortions, noise, and PTZ control, through the parametrization of such artifacts within the simulation environment.

In this chapter, originally published in [21], we propose a novel decentralized approach for dynamic network reconfiguration, where cameras locally control their PTZ parameters and position, to optimally cover the entire scene. For crowded scenes, cameras must deal with a trade-off among global coverage and target resolution to effectively perform crowd analysis. We evaluate our approach in a simulated environment surveyed with fixed, PTZ, and UAV-based cameras.

3.1 Introduction

Surveillance of crowded scenes is a key issue for public safety in indoor and outdoor environments. Various factors influence the development of a critical situation of crowds, hence a camera network must be able to capture local events as well as guarantee a global coverage of the whole area. Covering the entire monitoring

area while maintaining a sufficient resolution of the (moving) individuals might be challenging with fixed cameras. A costly camera infrastructure is necessary to provide a sufficient target resolution in every part of the monitoring area to perform common tasks such as personal identification. Consequently, video footage of potentially empty parts would also be captured with such a static camera network.

An alternative approach is to deploy reconfigurable cameras, which can dynamically adapt their field of view (FoV), resolution and position. In this case, the goal is to optimize coverage and target resolution depending on the current state of the crowded scene. Such camera networks aim to focus the attention on critical areas of the crowd but ensuring an acceptable level of attention also on less critical areas. In this chapter, we propose a novel network control approach to explore the trade-off between target resolution and coverage in heterogeneous networks consisting of fixed, PTZ, and UAV-based cameras. In our approach, we model the crowd scene and the camera network in a simulation environment, we estimate the state of the crowd by merging the contributions of the individual cameras' FOVs and we let cameras locally decide on their next PTZ or position parameters.

Our contribution can be summarized as (1) a policy to trade-off between global coverage and crowd coverage, (2) a new metric to evaluate the performances of the surveillance task, (3) a framework to track the crowd flow based on the coverage maps, and (4) a 3D simulator of crowd behaviors based on [64] and heterogeneous camera networks.¹

The remainder of this chapter is organized as follows: Section 3.2 briefly discusses related work. Section 3.3 describes the key components of the proposed approach along with the evaluation metric. Section 3.4 presents the results of our simulation study, and Section 3.5 provides some concluding remarks together with a discussion about potential future work.

3.2 Related Work

Automated video surveillance systems have been studied with the goal of reducing human intervention while operating a control room [110, 49, 136]. In such frameworks, cameras need to be aware of the network configuration sharing the necessary information to improve events capturing and global coverage of the scene [79, 89, 125]. Due to the dynamic nature of the events and the corresponding need for reconfiguring the camera network layout, research in the field has to deal with a limited amount of annotated data. This also makes each event unique and difficult to reproduce.

Relying on virtual environments and simulation tools can help to partially address these issues. Virtualization has been widely adopted in research, both in the community of camera networks [123, 142] and crowd analysis [73].

Pan, tilt, and zoom (PTZ) cameras have been deployed to survey crowded scenes [123, 142, 8]. PTZ cameras can be reconfigured to increase coverage of certain areas, either by progressively scanning the environment, or zooming in to specific locations in presence of events of interest. In a cooperative camera network, PTZ cameras can be effectively used to track targets of interest [8, 74, 19, 128].

Unmanned Aerial Vehicles (UAVs), or drones, have been adopted for different

¹Simulator available at <https://github.com/nick1392/HeterogenousCameraNetwork>

services and purposes, both in civil and military applications including environmental pollution monitoring, agriculture monitoring, and management of natural disaster rescue operations [131, 157, 75].

Yao et al. [158] identify the key features of a distributed network for crowd surveillance, i.e., to (1) locate and re-identify a person across the network, (2) track persons, (3) recognize and detect local and global crowd behavior, (4) cluster and recognize actions, and (5) detect abnormal behaviors. To achieve these goals, issues like how to fuse information coming from multiple cameras performing crowd behavior analysis tasks, how to learn crowd behavior patterns, and how to cover an area with particular focus on key events, are among a variety of challenges to be tackled.

3.3 Dynamic Camera Network Reconfiguration

Our approach is based on a set of fixed, PTZ, and UAV-based cameras with different characteristics and capabilities for the surveillance of crowded scenes. Multiple cameras provide diversity by observing and sensing an area of interest from different points of view, which further increases the reliability of the sensed data. Our framework for camera network reconfiguration is suitable for both static and dynamic scenarios.

In this section, we introduce the key components of our proposal. In particular, we first introduce the observation model for the environment, which describes the relationship between the observation and its confidence. We then describe, how each type of camera is modeled in the simulation environment, and formalize the reconfiguration objective. Next, we describe our reconfiguration policy that allows the network focus to be tuned in order to achieve a suitable trade-off between global coverage and crowd resolution.

3.3.1 Observation Model

The region of interest C , which has to be surveyed is divided in a uniform grid of $I \times J$ cells where the indexes $i \in \{1, 2, \dots, I - 1\}$ and $j \in \{1, 2, \dots, J - 1\}$ of each cell $c_{i,j} \in C$ represent the position of the cell in the grid. We assume a scenario evolving at discrete time steps $t = 0, 1, 2, \dots, t_{end}$. At each time step, the network is able to gather the observation over the scene to be monitored, process it, and share it with the other camera nodes in order to plan the next set of actions to be taken. For this purpose we define

- an observations vector $O_{i,j}$, which represents the number of pedestrians detected for each cell $c_{i,j} \in C$;
- a spatial confidence vector $S_{i,j}$, which describes the confidence of the measures for each cell $c_{i,j} \in C$. The value only depends on the relative geometric position between the observing camera and the observed cell;
- a time confidence vector $L_{i,j}^t$, which depends on the time passed since the cell has last been observed;
- an overall confidence vector $F_{i,j}^t$, which depends on the temporal and spatial confidences.

The observations vector is defined as

$$O_{i,j} = \{o_{1,1}, o_{1,2}, \dots, o_{i,j}, \dots, o_{I,J}\} \quad (3.1)$$

The value $o_{i,j}$ for each cell $c_{i,j}$ is given as

$$o_{i,j} = \begin{cases} \frac{ped}{ped_{max}} & \text{if } ped \leq ped_{max} \\ 1 & \text{if } ped > ped_{max} \end{cases} \quad (3.2)$$

where ped is the number of pedestrians detected within the cell by a given camera, and ped_{max} is the maximum number of pedestrian for a cell to be considered as crowded. Crowded cells should be monitored with a higher resolution.

The occlusion of targets is one of the main challenges in crowded scenarios. We assume that our camera network is able to robustly detect a pedestrian when its head is captured with a resolution of at least 24×24 pixels, in line with the smaller bound for common face detection algorithms [72].

For each cell a spatial confidence vector is defined as

$$S_{i,j} = \{s_{1,1}, s_{1,2}, \dots, s_{i,j}, \dots, s_{I,J}\} \quad (3.3)$$

where the value $0 < s_{i,j} \leq 1$ is bounded, and decreases as the distance between the observing camera and the cell of interest $c_{i,j}$ increases. The actual value of a cell depends on the type of observing camera and is described in Section 3.3.2.

Similarly, a time confidence vector is defined as

$$L_{i,j} = \{l_{1,1}^t, l_{1,2}^t, \dots, l_{i,j}^t, \dots, l_{I,J}^t\}. \quad (3.4)$$

Each value $l_{i,j}^t$ is defined as

$$l_{i,j}^t = \begin{cases} 1 - \frac{t - t_{i,j}^0}{T_{MAX}} & \text{if } t - t_{i,j}^0 \leq T_{MAX} \\ 0 & \text{if } t - t_{i,j}^0 > T_{MAX} \end{cases} \quad (3.5)$$

where $t_{i,j}^0$ is the most recent time instant, in which cell $c_{i,j}$ was observed, and T_{MAX} represents the time instant, after which the confidence drops to zero. The value $l_{i,j}^t$ decays over time if no new observation $o_{i,j}$ on cell $c_{i,j}$ become available.

Given the spatial and temporal confidence metrics, the overall confidence vector is defined as

$$F^t = \{f_{1,1}^t, f_{1,2}^t, \dots, f_{i,j}^t, \dots, f_{I,J}^t\} \quad (3.6)$$

with

$$f_{i,j}^t = s_{i,j} * l_{i,j}^t. \quad (3.7)$$

Thus, for each cell $c_{i,j}$ we have an observation $o_{i,j}$ with an overall confidence $f_{i,j}^t$. The confidence value varies between 0 and 1, where 1 represents the highest possible confidence. If a sufficient number of cameras is available for covering all cells concurrently, the overall confidence vector is given as $F^I = \{1, \dots, 1\}$.

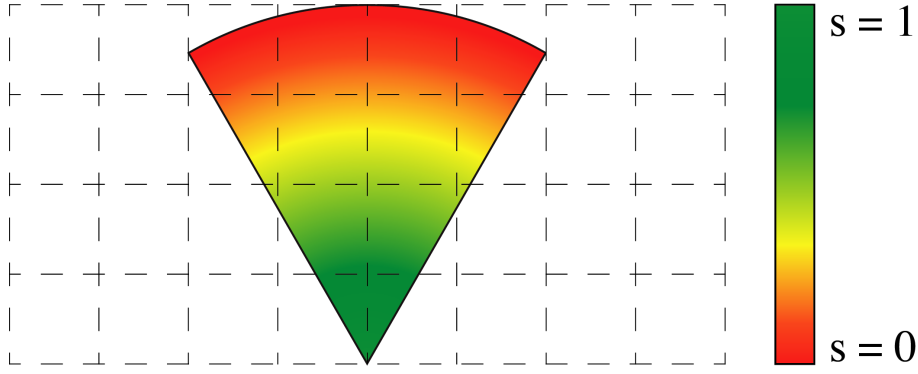


Figure 3.1: A fixed camera observes the environment without varying the spatial confidence for each cell at each time step.

3.3.2 Camera Models

We briefly describe the models adopted for the three different camera types: fixed cameras, PTZ cameras, and UAV-based cameras. We assume that all fixed and PTZ cameras are mounted at a fixed height, such that their spatial confidence metric depends only on the distance from the cell. All UAV-based cameras fly at a fixed altitude.

Fixed Cameras

Fixed cameras (see Fig. 3.1) provide a confidence matrix, which gradually decreases as the distance from the camera increases. Being (x, y) a point in the space at a distance d from a fixed camera, the value of the spatial confidence $s(x, y)$ is defined as

$$s(x, y) = \begin{cases} -\frac{1}{d_{max}} * d + 1 & \text{if } d < d_{max} \\ 0 & \text{if } d \geq d_{max} \end{cases} \quad (3.8)$$

with d_{max} being the distance from the camera, over which the spatial confidence is zero. Thus, the confidence value $s_{i,j}$ of cell $c_{i,j}$ is defined as

$$s_{i,j} = \max\{s(x, y)\}_{\forall(x,y) \in c_{i,j}}. \quad (3.9)$$

PTZ Cameras

PTZ cameras are modeled similarly to fixed cameras, with the additional capability to dynamically change the field of view (see Fig. 3.2).

UAV-based Cameras

For UAV-based cameras the FOV projection on the ground plane is different with respect to the previous models, as shown in Fig. 3.3. The spatial confidence of point

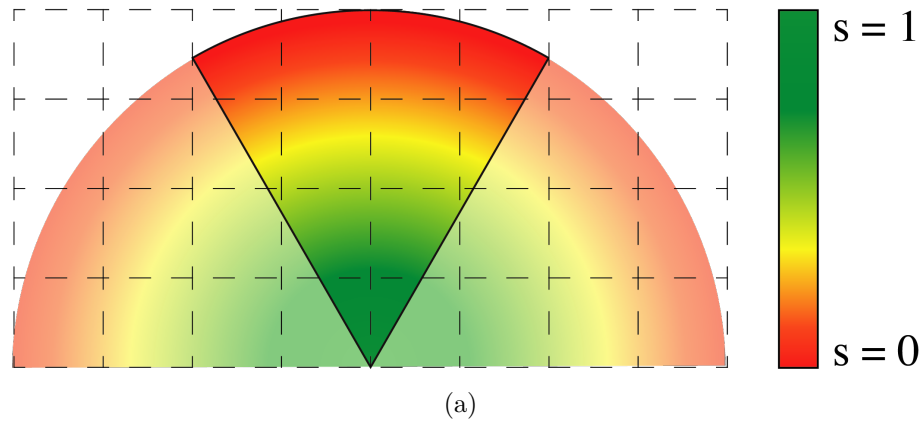


Figure 3.2: At each time step, a PTZ camera can pan its FOV in the range of 180° given a fixed initial position.

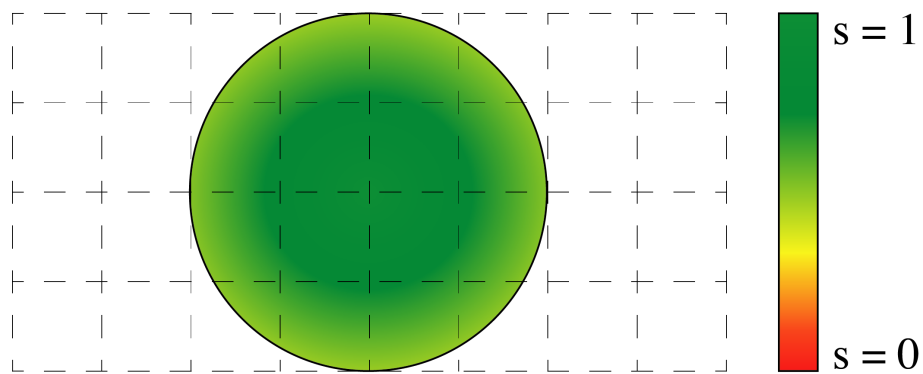


Figure 3.3: Example of the distribution of the spatial confidence in the area surveyed by an UAV.

(x, y) at a distance d from the UAV is computed as

$$s(x, y) = \begin{cases} -\frac{1}{d_{uav}} * d + 1 & \text{if } d < d_{uav} \\ 0 & \text{if } d \geq d_{uav}. \end{cases} \quad (3.10)$$

3.3.3 Reconfiguration Objective

The objective of the heterogeneous camera network is to guarantee the coverage of the scene while focusing on more densely populated areas. The priority metric defines the importance of each cell to be observed. A high value indicates that the cell is crowded or that we have low confidence in its current state, thus requiring an action.

In order to formalize the reconfiguration objective, a priority vector P is defined as

$$P^t = \{p_{1,1}^t, p_{1,2}^t, \dots, p_{i,j}^t, \dots, p_{I,J}^t\}. \quad (3.11)$$

The priority for each cell is defined as

$$p_{i,j}^t = \alpha * o_{i,j}^t + (1 - \alpha) f_{i,j}^I \quad (3.12)$$

where $0 \leq \alpha \leq 1$ represents a weighting factor to tune the configuration and $f_{i,j}^I$ represents the pre-defined ideal confidence for the cell.

The objective G of each camera, given its possible set of action, is to minimize the distance between the confidence vector and the priority vector

$$G = \min\{\|F^{t+1} - P^t\|\} \quad (3.13)$$

$$\begin{cases} \min\{F^{t+1} - F^I\} & \text{if } \alpha = 0 \\ \min\{F^{t+1} - O^t\} & \text{if } \alpha = 1 \end{cases} \quad (3.14)$$

Setting $\alpha = 1$ causes the network to focus on observing more densely populated areas with no incentive to explore unknown cells. In contrast, $\alpha = 0$ causes the network to focus on global coverage only without distinguishing the crowd density of the cells.

3.3.4 Update Function

At each time step t , the network has knowledge about the current observation vector O^t , the spatial confidence vector S^t , the time confidence vector L^t , and the overall confidence vector F^t . In order to progress to the next time step $t + 1$, an update function for these vectors is required.

The temporary spatial confidence vector S_{temp}^{t+1} is determined by the geometry of cameras at time $t + 1$. For each cell, the value $s_{temp,i,j}^{t+1}$ is the maximum spatial confidence value of all cameras observing the cell (i, j) . Cells that are not covered by any camera will have a spatial confidence value of 0.

We estimate the time confidence vector as follows. L_{time}^{t+1} is computed by applying Eq. 3.5 to each element of L^t . Another temporary time confidence vector L_{new}^{t+1} is computed setting to 1 the value of all cells currently observed, and setting to 0 all

other cells.

With the estimated vectors we compute two estimations of the overall confidence vector such that:

$$F_{time}^{t+1} = S^t * L_{time}^{t+1} \quad (3.15)$$

$$F_{new}^{t+1} = S_{temp}^{t+1} * L_{new}^{t+1} \quad (3.16)$$

The new overall confidence vector is then computed as

$$F^{t+1} = \max\{F_{new}^{t+1}, F_{time}^{t+1}\}_{\forall(i,j)}. \quad (3.17)$$

For each cell (i, j) in which $f_{new}^{t+1} > f_{time}^{t+1}$, we also need to update the last time the cell has been observed $t^0(i, j) = t + 1$, and the observation vector $o^t(i, j)$.

3.3.5 Local Camera Decision

In our approach, all the information vectors described in Section 3.3.1 are shared and known to all cameras. Each camera locally decides its next position using a greedy approach to minimize the cost defined in Eq. 3.13 in its neighborhood.

At each time step, each mobile, PTZ, and UAV-mounted camera select a neighborhood that can be explored. The UAV's neighborhood is defined as a square centered at the cell where the drone is currently placed (see Fig. 3.3). The PTZ neighborhood is a rectangle that covers the space in front of the camera as shown in Fig. 3.2.

For each cell in the neighborhood, we center a window W of size $N_w \times N_w$ on each cell $c_W \in W$ and we store in the cell the value

$$c_W = \sum \|f_{i,j}^{t+1} - p_{i,j}^t\|. \quad (3.18)$$

The UAV will then move toward the cell in its neighborhood with the largest c_W , and the PTZ steers its FOV to be centered on that cell. If two or more cells have the same value of c_W , the camera selects one of them randomly.

3.3.6 Evaluation Metrics

We define the Global Coverage Metric (GCM) for evaluating the network coverage capability as

$$GCM(t) = \frac{\sum_{\forall c_{i,j} | f_{i,j}^t > g} 1}{I * J} \quad (3.19)$$

with g being the threshold above which we consider the cell covered. We then average the results for the whole duration of the observation as follows:

$$GCM_{avg} = \sum_{t=0, \dots, t_{end}} \frac{GCM(t)}{t + 1} \quad (3.20)$$

We define the People Coverage Metric (PCM) for evaluating the network capability

to cover pedestrian in the scene as

$$PCM_{tot} = \frac{\sum_{\forall person \in c_{i,j} | f_{i,j}^t > p} 1}{totalPeople} \quad (3.21)$$

with p being the threshold above which we consider the cell covered.

3.4 Experimental Results

For the experiments we define an environment of size 60×60 meters. The scene is square-shaped exhibiting people passing by, cars, and vegetation. Pedestrians can enter and exit the scene from any point around the square. Each cell $c_{i,j}$ is a square of 1×1 meter. In this environment 2 fixed cameras, 2 UAVs and 2 PTZs are positioned as shown in Fig. 3.4(a). Sample images of the environment from a PTZ and a UAV-based camera are shown in Figures 3.4(b) and 3.4(c), respectively. For our experiments we simulate the movement of 400 pedestrians crossing the scene with the following parameters :

- $T_{max} = 3$ seconds
- $ped_{max} = 2$
- $d_{max} = 10$ meters
- fixed and PTZ cameras height = 5 meters
- UAV cameras height = 7 meters

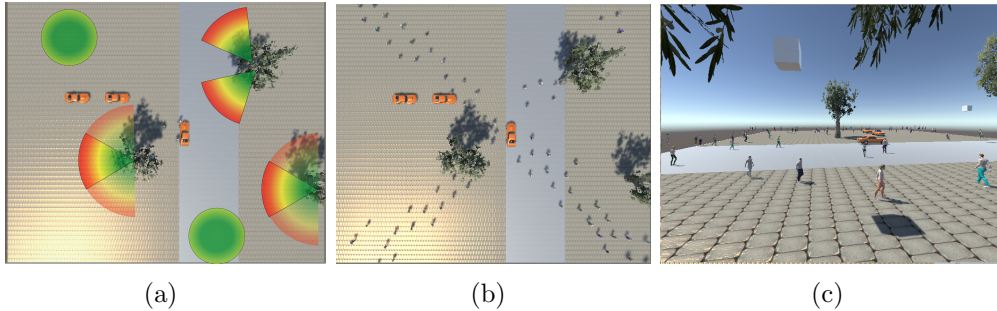


Figure 3.4: (a) Top view of the simulation environment including the camera positions. (b) Sample image from a UAV-based camera. (c) Sample image from a PTZ camera.

3.4.1 Quantitative Results

In this section, we present the quantitative results obtained with our model in the simulated environment. The goal is to evaluate the capabilities of the system to survey a crowded scene using the metrics defined in Sec. 3.3.6. We run 9 different simulation experiments with varying values of g , p , and α .

ID	g and p	α	GCM	PCM
1	0.2	0	12.4 %	17.4 %
2	0.2	0.5	14.3 %	20.5 %
3	0.2	1	10.4 %	13.5 %
4	0.01	0	42.9 %	47.6 %
5	0.01	0.5	30.3 %	33.1 %
6	0.01	1	22.9 %	28.2 %
7	0.01	0	43.1 %	45.6 %
8	0.01	0.5	28.7 %	54.4 %
9	0.01	1	26.1 %	61.2 %

Table 3.1: Simulation experiments. Legend: ID–experiment; g,p –cell coverage thresholds; GCM–global coverage metric; PCM–people coverage metric. Experiments 1-6 refer to a uniformly distributed crowd, experiments 7-9 refer to a crowd with directional motion properties.

The values for g and p indicate how reliable the information is about the position in space and pedestrians, respectively. A threshold of 0.2 indicates that our observation is at most 2.4 seconds old when taken with spatial confidence equal to 1. A threshold of 0.01 represents the cells and pedestrians about which we have a minimum level of information.

As a reference if all 6 cameras remain fixed, they are able to cover 6 % of the entire area with $g = 0.2$ and 12 % with $g = 0.01$. In experiments (3) and (6), α is set to 1, causing our camera network to focus only on observing pedestrians with no incentive to explore new areas in the environment. In experiments (1) and (4), α is set to 0 resulting in maximizing the coverage regardless of the position of pedestrians. In experiments (2) and (5), α is set to 0.5 aiming for balancing coverage and pedestrian tracking in crowded areas. We can observe that in experiments (1) and (4) we obtain the lowest values of GCM, which is expected since we are focusing on pedestrians. We also achieve the lowest scores in terms of PCM because cameras have no incentive in exploring new areas.

Experiments (7), (8), and (9) are conducted using a directional crowd (Fig. 3.4(b)). When the network focuses only on observation in (9), it obtains the best results in terms of PCM and the worst one in terms of global coverage GCM. As expected, we obtain the best results in terms of coverage of the environment (GCM) in experiments (3) and (6). Since the crowd is uniformly distributed in the space, we also obtain the best results in terms of PCM. In experiments (2) and (5), the network combines global coverage and crowd monitoring, the system underperforms compared with the scenes where $\alpha = 0$ and $\alpha = 1$.

3.4.2 Qualitative Results

In this section, we present the qualitative results obtained with our model in the simulated environment. The goal is to demonstrate, how our system is able to follow the crowd.

For this purpose, we simulate a single group of five pedestrians crossing the scene from the bottom left to the top right as shown in the sequence depicted in

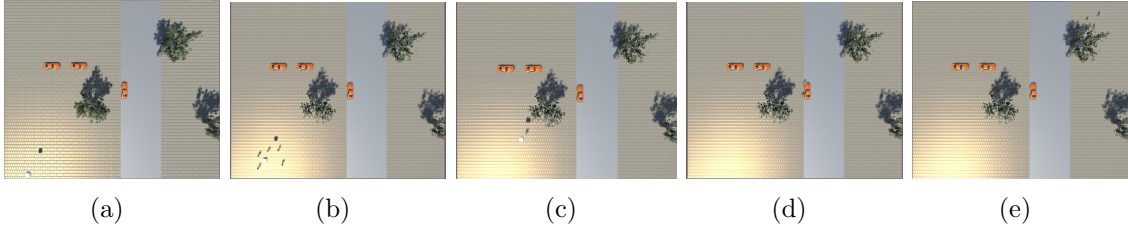


Figure 3.5: Image sequence of a group of pedestrian moving from the bottom left of the environment (a) to the top right (c). The image is captured by a top view camera during the simulation to demonstrate the tracking behavior of our network.

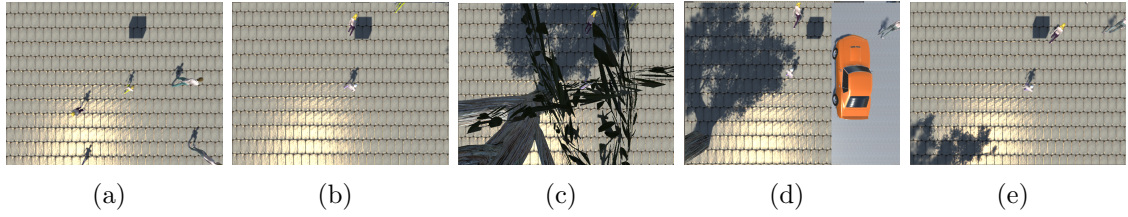


Figure 3.6: Image sequence of a group of pedestrian moving from the bottom left of the environment (a) to the top right (e) captured by a UAV surveying the scene.

Fig. 3.5. The UAV is able to closely follow the pedestrians in the environment, scoring a $PCM = 70.4\%$ and $GCM = 3.2\%$, as shown in Fig. 3.6. Fig. 3.7 shows how observation, priority and confidences maps are updated over time in order to guide the UAV in the tracking scenario.

Scenario	Priority P^t	Observation O^t	Time confidence L^t	Spatial confidence S^t	Overall confidence F^t
(1)					
(2)					
(3)					

Figure 3.7: Graphical representation of priority P^t , observation O^t , time confidence L^t , spatial confidence S^t and overall confidence F^t for 3 different scenarios: (1) Camera Network Sample, (2) Tracking sample at time $t = 0$, (3) Tracking sample at time $t = 10$. In (2) and (3) the UAV focuses on the observation matrix, such that the next priority map depends only on previous observations. Red represent the value 0, and green represents value 1.

3.5 Conclusions

In this chapter, we have presented a novel camera reconfiguration approach for crowd monitoring. Our approach allows heterogeneous camera networks to focus on high target resolution or on wide coverage. Although based on simplified assumptions for camera modeling and control, our approach is able to trade-off coverage and resolution of the network in a resource-effective way. In future research, network coordination will be improved relying on cooperative decision-making between cameras and assigning different policies (e.g., values of α) to different camera types.

We have shown how we can leverage the features provided by a crowd simulator for testing and developing a surveillance network, which is part of many infrastructures in our cities.

4 Toward anomaly detection in the wild: out of distribution detection

More recently, the trend in crowd analysis has shifted towards the characterization of the emerging behaviors resulting from motion. It is though worth noting in order to be defined as such, a *behavior* although referred to a collective event rather than to a single person, should still preserve the fundamental properties of responding to internal, external, conscious, or unconscious stimuli [106]. In fact, although considerably more complex compared to the case of a single moving person, also the crowd has to relate to the surrounding context, which is constrained on the one hand by the environment, including the presence of objects, obstacles, entry/exit gates [26] and on the other hand is ruled by the perception and motivation of the single subjects composing the crowd. In fact, when dealing with a mass of subjects the internal and external stimuli are even more pronounced due to the mutual influence that the different elements composing the crowd receive and transmit to their neighbors [111]. This makes the separation between an action and a behavior less marked, as, for example, an action of a member of the crowd (e.g., running, shouting) might trigger a stimulus to one or more neighbors, generating a collective behavior (e.g. panic).

For example anomalies fall into this category of events, and can be broadly defined as the deviation from what has been observed beforehand, or, in other words, what is *normal*; due to their nature, anomalies are rarely occurring in real test video sequences and are often staged. An exhaustive work in the area of crowd behavior analysis is presented by Solmaz et al. [139]. The authors propose a method to recognize five different types of behaviors, namely lane, circle, bottleneck, fountainhead, and blocking. The algorithm, is based on the detection of accumulation points, which are then used to trace back the motion of the crowd.

Analyzing the optical flow properties, Feng and Bhanu [48] follow a hierarchical approach that progressively and dynamically merges the tracklet information collected from the visual scene, so as to construct an interaction graph that defines the existence of a group. What makes a group a particularly interesting social configuration, is that it represents an intermediate structure in between the individuals and the crowd as a whole. In the group the contribution of each member is usually distinguishable and the interactions among the entities in the group is the discriminating feature to not only detect the class of behavior [32] but also to improve the detection and tracking [117].

As far as the detection of anomalies is concerned, the literature has mostly focused on the deviations on the motion properties of the crowd [108]. Relying on the indirect approach, the authors treat particles as humans and track their motion

over time. The interaction forces among the particles are used to define a so-called force flow, which is used as background information to model the regular flow of the crowd. Latent Dirichlet Allocation [25] is then used to train the model and thus detect the presence of anomalies. Bera et al. [16] propose a real-time framework to detect anomalies in low to medium-crowded scenes. Thanks to an efficient tracking algorithm and relying on the RVO motion model, the authors derive the global motion properties of the crowd. Analyzing then the local features of the moving agents, and highlighting when the value of the feature exceeds a threshold, anomalies can be identified and signaled.

In the area of deep learning, Shao et al. [137] propose to extend the use of CNNs also to the temporal domain, learning the crowd attributes by also leveraging on the temporal information.

In the context of deep learning, a network is usually trained on a fixed number of classes. At test time the network is able to recognise only the classes for which it was trained. Since there is a scarcity of samples, common deep learning frameworks cannot be trained to perform anomaly detection. The goal of out-of-distribution (OOD) detection is to handle novel situations where the test samples are drawn from a different distribution than the training data. If a network is trained with multiple scenes of common behaviors, an anomalies can be defined as an out distribution sample, which has not been seen by network before test time.

4.1 Introduction

Many conventional machine learning techniques are being designed and deployed under *closed-set* assumptions, meaning that training data contains samples from all the possible classes that the classifier will encounter. Of course, such assumption does not hold in many applications such as anomalies detection in crowded scenes; as it may not be possible to cover every potential input class in the training dataset. Thus, the goal of open-set classifiers is to detect out-of-distribution (OOD) samples; the input instances that do not belong to any of the training classes. In general, many of the OOD detection techniques try to either use the class membership probabilities as a measure of uncertainty [65, 94, 151], or define a measure of similarity between the input samples and the training dataset in a feature space [14, 161, 84].

As discussed in [84], the features extracted from a conventional softmax classifier follow a class-conditional Gaussian distribution. Such structure has been exploited in [14, 161, 84] to detect the outliers/OOD samples. However, general class-conditional Gaussian structures are not particularly appropriate for outlier rejection. That is due to the fact that the statistics of such distribution are not robust to perturbations, noisy samples, and outliers [45]. Furthermore, such structure is not particularly easily distinguishable in the feature space.

In this work, we claim that we can significantly improve the OOD detection performance by imposing the structure of in-distribution samples in the feature space. Particularly, if we embed the training samples, employing deep neural networks, such that the feature vectors belonging to each known class lie on a 1-dimensional subspace, OOD samples can be detected more robustly with higher probability. Such a union of 1-dimensional subspaces structure provides us with two main advantages.

First, due to compact representation in the feature space, OOD samples are less likely to occupy the same region as the known classes. In other words, a random vector in a high-dimensional space lies on a specific 1-dimensional line with probability 0. Second, we show that the first spectral component (first singular vector) of a 1-dimensional subspace is a robust representative of the its samples. We exploit these two desirable features and reject samples as OOD if they occupy the region corresponding to the training samples with probability 0. This region is identified by the set of the first singular vectors of the training classes. Furthermore, to estimate the probability, we use Monte Carlo sampling techniques used in Bayesian deep learning.

Our proposed method, unlike [113, 161], does not require any sample generation or reconstruction. Furthermore, we do not need extra information or a subset of OOD examples for hyperparameter tuning or validation. This is in contrast with many existing methods that use some subset of the OOD samples either during validation [94, 151, 84], or even during training [162]. Despite improving the results, the availability of such extra information is questionable in real applications. In summary, this work makes the following contributions:

- We show that if deep feature vectors lie on a union of 1-dimensional subspaces, the OOD samples can be detected with higher probabilities and more robustly;
- We demonstrate how we can impose such spectral structure onto the feature space, by incorporating the absolute cosine similarity into the softmax activation;
- We propose a technique to exploit the first singular vector of the feature vectors extracted from the training set to detect the out-of-distribution (OOD) samples;

The rest of the manuscript is organized as follows. Section 4.2 summarizes the related work and formally define the problem at hand. In Section 4.3 we present our motivation behind enforcing the feature vectors to lie on a union of 1-dimensional subspaces and how we can achieve it. Then, in Section 4.4, we explain how such structure can be used to distinguish between in- and out-of-distribution samples. Finally, Section 4.5 deals with the experimental results and Section 4.6 concludes the chapter.

4.2 Problem Statement and Related Work

Given a training dataset consisting of N sample-label pairs belonging to L known classes, our goal is to train a neural network such that at the test time it can be determined if an unlabeled sample is an out-of-distribution sample (not belonging to any of the L known classes) or not. It is known that conventional neural networks that are trained under closed-set assumption do not operate well under such conditions and may even have high confidence in regions in feature space far away from the training set [62, 44].

Existing approaches either allow the network to access the OOD datasets during training/validation steps [162, 94, 151, 84], or the OOD datasets are seen during the test phase [14, 65, 161].

In [162], the network is fine-tuned during the training to increase the distance between known and OOD distributions. Other methods [94, 151, 84] apply a perturbation on each sample at test time to exploit the robustness of their network in detecting in-distribution samples. However, they use part of the OOD samples to fine-tune the perturbation parameters, thus obtaining the maximum distance between in and out of distribution samples. *We argue that OOD detectors should be completely agnostic of unknown distributions, which is a more realistic scenario in the wild.*

Few existing approaches, such as [65, 14, 161], do not require the OOD samples neither during training or validation. Hendricks and Gimpel [65] provide a baseline for the open set detection, showing how the Softmax layer can be used to detect OOD samples, when its prediction score is below a threshold. In [14], the activation vectors at the penultimate layer of the network are employed to determine in and out of distribution samples. The standard Softmax layer is replaced with an OpenMax layer which is able to predict the probability of a sample belonging to the known distribution or not. In [161], the authors exploit open set classification in the latent space instead of working directly on the last layer of the network. Following this line of works, we assume that neither we have access to the OOD samples during training nor during validation.

In the face recognition domain [100, 152], they introduced angular losses to improve the deep face recognition task. In [100], the authors propose a new loss which enforces an angular (spectral) structure between training features. Doing so, the angular distance between known classes and test samples is exploited as a reliable metric for classification. In [152], cosine similarity score between training and test samples is employed to perform face identification and verification.

Inspired by [100, 152], we exploit a loss function based on cosine similarity. Thus, we are able to enforce a spectral structure in the latent space using only in-distribution classes at training and validation phases. At test time, the enforced structure allows us to distinguish between OOD and known samples, given how they are represented in the latent space by the enforced spectral structure.

4.3 Union of 1-dimensional subspaces as the in-distribution structure

In this section, we present our framework for detecting the out-of-distribution samples. First, we describe the motivation by discussing the advantages of enforcing our proposed structure onto the deep feature vectors. Then, motivated by our theoretical analysis, we propose techniques to enforce this structure.

We argue that OOD detection performance can be improved if the feature vectors from the known classes lie on a *union of 1-dimensional subspaces*. In short, such structure has two main properties that we can take advantage of for OOD detection:

1. Due to the compactness of samples in the feature space of a known class, OOD samples can be detected with higher probability and
2. First singular vector of the samples in each class can be used as a robust representative of that class and be employed to effectively distinguish between

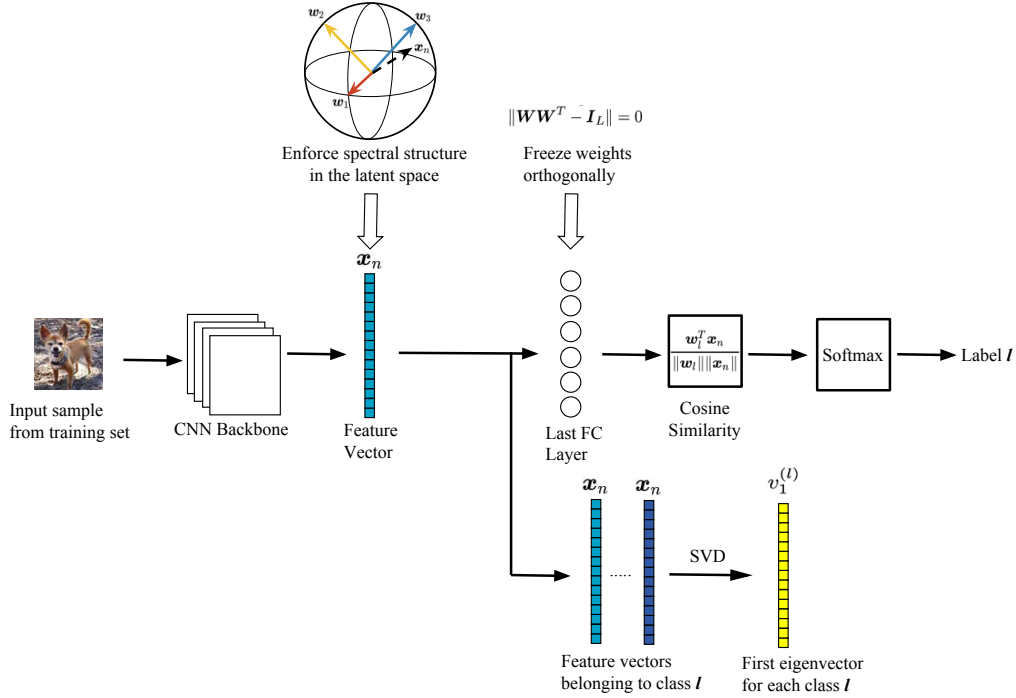


Figure 4.1: Overall architecture of the proposed framework at training time. A convolutional neural network (CNN) is used to map the input sample onto a feature space. Then, the cosine similarities between the extracted feature \mathbf{x}_n and the class vectors (rows of the last fully connected layer) \mathbf{w}_l are used to compute the class membership probabilities. \mathbf{w}_l s are set to predefined orthonormal vectors and are not updated during training. This enforces the desired structure, union of uncorrelated 1-dimensional subspaces, on the feature vectors.

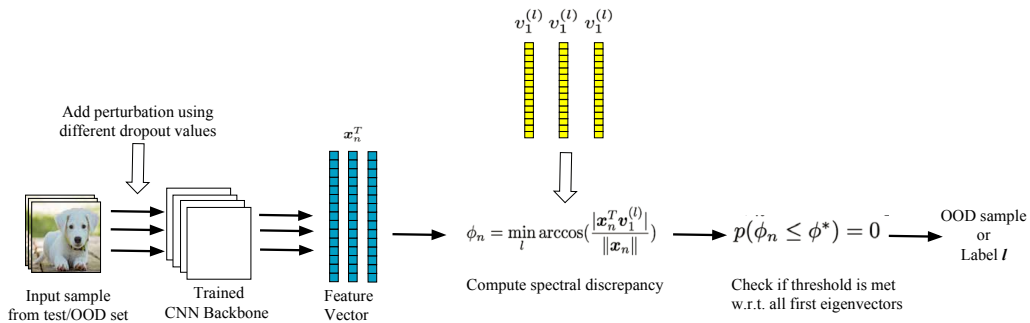


Figure 4.2: Overall architecture of the proposed framework at test time. At test time, we extract multiple times the features of each test sample, each time with a different dropout setting. The spectral discrepancy between the test samples and the first singular vector corresponding to each class are used to distinguish between the in- and out-of-distribution samples.

the in- and out-of-distributions samples.

Below, we discuss each of these advantages in more details.

Distribution-agnostic minimization of error probability: Calculating the error probability for OOD detection is a difficult task to carry out. This is due to the fact that, by definition, we do not have any information on the probability distribution of the OOD samples. However, it can be shown that the probability of error can be minimized by making the distribution of the known classes as compact as possible. Specifically, consider a binary classification problem, where both of the classes follow multivariate Gaussian distributions with different means and covariance matrices $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. It has been shown that the classification error probability p_e can be upper bounded as [47]:

$$p_e \leq \sqrt{p_1 p_2} e^{-B},$$

where p_1 and p_2 are the probability of samples belonging to each class and B is the Bhattacharyya distance:

$$B = \frac{1}{8} \boldsymbol{\Delta}^T \left(\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2} \right)^{-1} \boldsymbol{\Delta} + \frac{1}{2} \ln \left(\frac{\det(\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2})}{\sqrt{\det(\boldsymbol{\Sigma}_1) \det(\boldsymbol{\Sigma}_2)}} \right),$$

where $\boldsymbol{\Delta} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ is the distance between the means of the two distributions. The first term in B represents the Mahalanobis distance between $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, using $\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}$ as the covariance matrix. On the other hand, the second term is a measure of compactness of the distributions. The larger the $\det(\boldsymbol{\Sigma}_1)$ is, the more its corresponding samples are spread out. In the extreme case of $\det(\boldsymbol{\Sigma}_1) \rightarrow 0$, the samples lie on a low dimensional subspace and B goes to infinity. Thus, even without any knowledge on $\boldsymbol{\mu}_2$, $\boldsymbol{\Sigma}_2$, p_1 , and p_2 , one can increase B by making $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ as compact as possible. A compact $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ means less variation along the direction of $\boldsymbol{\Delta}$, which increases the first term, and a small $\det(\boldsymbol{\Sigma}_1)$, which increases the second term. This leads to an increase in B , and therefore an exponential reduction in p_e .

Intuitively, if we make feature vectors belonging to the known classes to occupy a tiny region in the space, we can detect any small deviations from this structure far more easily. Union of 1-dimensional subspaces has this property, as each class only covers a very tiny region, with no volume $\det(\boldsymbol{\Sigma}_1) \approx 0$, in the space.

First singular vector as robust representative: Union of 1-dimensional subspaces provides us with the additional benefit of robust outlier rejection. In the context of robust statistics, the first singular vector has been shown to be a great tool to define robust mean and covariance estimators [45]. In addition, it is well-known that the first singular vector of data points belonging to the same class represent the class very well [163]. It can be shown that the first singular vector is robust to perturbations and outliers as discussed in the following.

Let \mathbf{X}_l denote an $M \times N$ matrix containing N M -dimensional *feature vectors* belonging to class l . These feature vectors are extracted from the penultimate layer of the neural network and are the input to the fully connected layer. Furthermore,

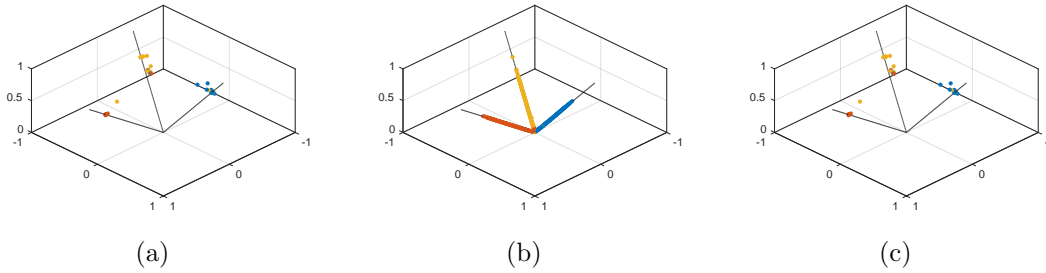


Figure 4.3: 3-dimensional representation of the features belonging to the first 3 classes of CIFAR10 training set, extracted from WideResNet with and without the proposed structure: (a) features extracted from a plain WideResnet, (b) features extracted WideResnet after enforcing the proposed structure, and (c) ℓ_2 -normalized features extracted WideResnet after enforcing the proposed structure. This shows that the classes have compact representation in terms of angular distance, which does not depend on the norm of the feature vectors. The lines represent the direction of the first singular vector corresponding to each class.

consider the autocorrelation matrix of the class l defined as $\mathbf{C}_l = \mathbf{X}_l \mathbf{X}_l^T$. Eigenvectors and eigenvalues of \mathbf{C}_l are the left singular vectors and the square of singular values of \mathbf{X}_l , respectively. Adding noise or adding a new noisy entry in \mathbf{X}_l perturbs \mathbf{C}_l , without changing its dimensions. To quantify the sensitivity of eigenvectors of \mathbf{C}_l against perturbations, we use the following remark.

Lemma 1 (from [163]) *Assume square matrix \mathbf{C} and its spectrum $[\lambda_i, \mathbf{v}_i]$. Then, the following inequality holds,*

$$\|\partial \mathbf{v}_i\|_2 \leq \sqrt{\sum_{j \neq i} \frac{1}{(\lambda_i - \lambda_j)^2}} \|\partial \mathbf{C}\|_F.$$

This means that the sensitivity of the i^{th} spectral component, \mathbf{v}_i , to perturbations, is inversely related to the gap between its corresponding eigenvalue λ_i and other eigenvalues $\lambda_j, j \neq i$. Therefore, we can define the sensitivity coefficient of the i^{th} eigenvector of a square matrix as $s_i \triangleq \sqrt{\sum_{j \neq i} \frac{1}{(\lambda_i - \lambda_j)^2}}$. In general, the first singular component \mathbf{v}_1 is the least sensitive direction to the perturbations. This is due to the fact that, in many scenarios, the gap between consecutive eigenvalues is decreasing (see [31] and references therein). Under such conditions it has been shown that $s_1 < s_i, \forall i \geq 2$, i.e., \mathbf{v}_1 is the least sensitive direction. However, we can further increase the robustness, by enforcing union of 1-dimensional subspaces structure onto the space of the features extracted from the data points. Specifically, If most of the energy of the data points in each class is concentrated along its corresponding first singular vector, that will result in large λ_1 and small $\lambda_i, i \geq 2$ for all the classes. Therefore, if the feature vectors belonging to the same class lie on a 1-dimensional subspace, we can use the first singular vector of \mathbf{X}_l as a robust representative of the class and to reject outliers.

4.3.1 Enforcing the Structure

In this section, motivated by our theoretical analysis, we propose a simple, yet effective, method to enforce our desired structure:

Intraclass Structure: As discussed earlier, we want the feature vectors of each class to lie on a 1-dimensional subspace. We achieve this by employing cosine similarity in classification, by modifying the softmax function to predict the membership probability using:

$$p_{ln} = \frac{e^{|\cos(\theta_{ln})|}}{\sum_l e^{|\cos(\theta_{ln})|}}, \quad (4.1)$$

where p_{ln} is the probability of membership of feature vector n in class l and $\cos(\theta_{ln}) = \frac{\mathbf{w}_l^T \mathbf{x}_n}{\|\mathbf{w}_l\| \|\mathbf{x}_n\|}$ is the cosine similarity between the learned feature vector \mathbf{x}_n , and the weights of the last, fully connected, layer corresponding to class l , i.e., \mathbf{w}_l . Note that, *unlike other methods which employ angular margin [152, 100], we use the absolute value of the cosine similarity to compute the class memberships*. This is due to the fact that the subspace membership, and therefore the class membership, does not change if a vector is multiplied by -1 . By employing such activation function, the feature vectors of each class are aligned to its corresponding weight vector \mathbf{w}_l . In other words, class l forms an almost 1-dimensional subspace along the direction of \mathbf{w}_l in the feature space. Therefore the final loss function to be minimized is:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N -\log\left(\frac{e^{|\cos(\theta_n^*)|}}{\sum_l e^{|\cos(\theta_{ln})|}}\right), \quad (4.2)$$

where θ_n^* is angle between the n^{th} feature vector and the weight vector corresponding to its true label.

Interclass Structure: By using the absolute cosine similarity as the classification criteria, we can ensure the feature vectors are angularly distributed in the space and form a union of 1-dimensional subspaces. To boost the interclass separation of the known classes, we need to decrease the interclass similarity, in terms of cosine similarity. This can be done by regularizing the weights $\mathbf{w}_l, \forall l$. As mentioned earlier, by using the cosine similarity as the input of the softmax, the samples of class l are distributed along the direction of \mathbf{w}_l . Thus, minimum interclass cosine similarity can be enforced by ensuring that \mathbf{w}_l s are orthogonal to each other. We can achieve this by simply initializing the weight matrix with orthonormal vectors and freezing them during the training. In other words, the feature extractor, i.e., the CNN backbone, is trained such that it can map each input samples in class l onto a predefined 1-dimensional subspace represented by the direction of \mathbf{w}_l .

Figure 4.1 shows the overall architecture of the proposed framework. The CNN backbone maps the input sample onto a low-dimensional space, where the known classes are represented by a set of orthonormal vectors. The cosine similarity between the extracted feature from the n^{th} input sample, \mathbf{x}_n , and the vector corresponding to the class subspace, \mathbf{w}_l , is used to determine the class membership probability and therefore the label.

Figure 4.3 demonstrates the effectiveness of the proposed framework in enforcing the desired structures. It shows a 3-dimensional embedding, obtained by PCA, of the feature vectors belonging to the first 3 classes of CIFAR10. The CNN backbone, WideResnet28, is trained on all the classes of CIFAR10 with and without enforcing the structure. Figure 4.3(a) shows that the feature vectors belonging to each class extracted from a plain WideResnet have a fairly isometric Gaussian

structure, meaning that they are spread out in different direction fairly uniformly. On the other hand, as shown in Figure 4.3(b), the feature vectors extracted from the same network trained using our proposed technique lie on a union of 1-dimensional subspaces. We also show the ℓ_2 -normalized feature vectors in Figure 4.3(c), since the class membership probability exploits cosine similarity, which is independent of the norm the feature vectors. This further shows the effectiveness of the proposed technique in enforcing the desired structure.

4.4 Out-of-distribution Detection Test

In this section, we discuss how we can exploit the desirable features of the proposed structure, namely compactness and robustness, to detect OOD samples. Assuming that the feature vectors belonging to the known classes lie on a union of exactly 1-dimensional subspaces, i.e., their corresponding region in the feature space has no volume. Therefore, OOD samples will be inside this region with probability 0. More specifically, probability of OOD samples being in the region corresponding to any of the known classes, which is probability of false negative p_{fn} , is zero. This can be seen using the Bhattacharyya bound, discussed in Section 4.3:

$$p_{fn} \leq p_e \leq \sqrt{p_1 p_2} e^{-B}.$$

Thus, if we make the known classes occupy a tiny region with no volume in the space, we will have $B \rightarrow \infty$ and therefore $p_{fn} \rightarrow 0$. We use this property and classify samples as OOD if they lie inside the region corresponding to any of the known classes with probability 0. More specifically, given an input instance \mathbf{i}_n , this probability can be estimated in robust manner, using the singular vectors of each class, as:

$$p(\phi_n \leq \phi^* | \mathbf{i}_n),$$

where ϕ_n is defined as:

$$\phi_n = \min_l \arccos\left(\frac{|\mathbf{x}_n^T \mathbf{v}_1^{(l)}|}{\|\mathbf{x}_n\|}\right), \quad (4.3)$$

which is the minimum angular distance of the test feature vector \mathbf{x}_n , corresponding to \mathbf{i}_n , with the first singular vector of any of the classes. We name this measure as *spectral discrepancy*. ϕ^* is a critical spectral discrepancy and defines the region belonging to the known classes. Smaller values of ϕ^* corresponds to more compact regions. In the extreme case of $\phi^* = 0$, the input instance \mathbf{i}_n will be detected as OOD, if it does not have the exact same direction as one of the singular vectors. $\mathbf{v}_1^{(l)}$ can be computed using the extracted features from training samples of class l . Time complexity order of computing the first singular vector is linear w.r.t both the number and the dimension of the feature vectors [38, 10].

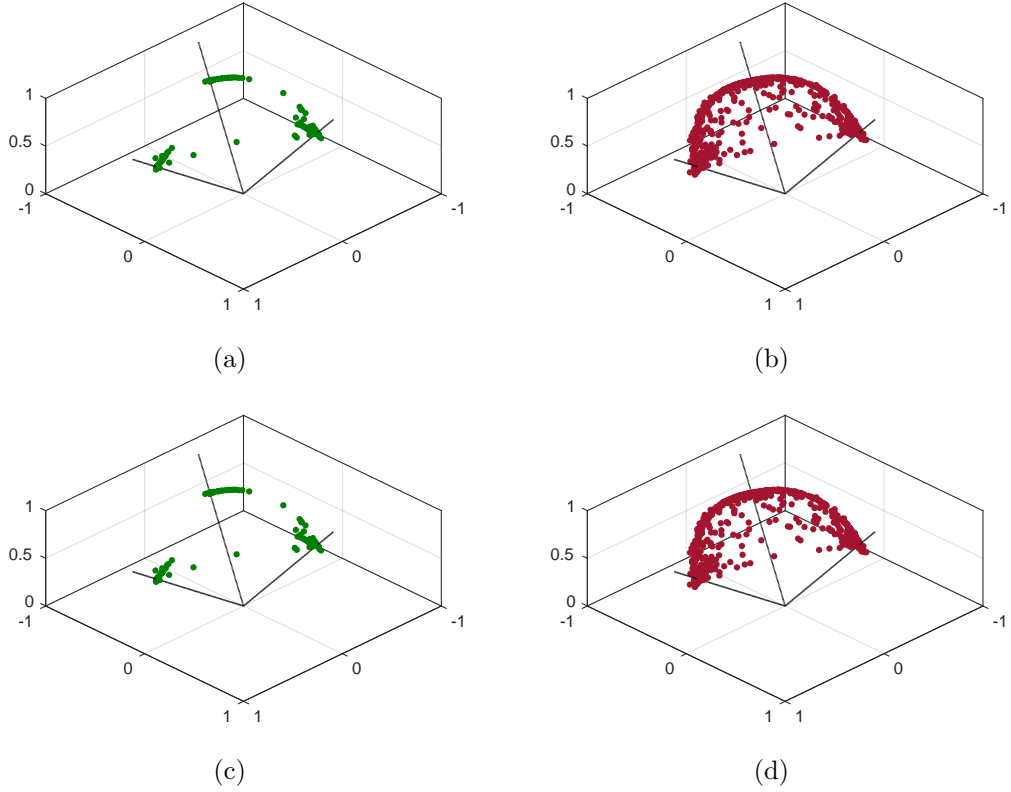


Figure 4.4: 3-dimensional representation of the features extracted from a plain WideResNet (top row) and the same network with our proposed structure (bottom row) trained on CIFAR10. (left column) in-distribution samples (CIFAR10 test set) that are close to the first 3 classes of CIFAR10 training set, and (right column) out-of-distribution samples (TinyImagenet) that are close to the same classes. The lines in the bottom row represent the direction of the first singular vector corresponding to each class. Our proposed structure can distinguish between the in- and out-of-distribution samples more effectively, i.e., OOD samples have significantly larger angular distance to their closest singular vector.

To estimate $p(\phi_n \leq \phi^* | \mathbf{i}_n)$, we use Monte Carlo sampling:

$$\begin{aligned}
 p(\phi_n \leq \phi^* | \mathbf{i}_n) &= \int_0^{\phi^*} p(\phi_n | \mathbf{i}_n) d\phi_n \\
 &\approx \frac{1}{T} \sum_{t=1}^T \mathbb{I}(\phi_n^t < \phi^*),
 \end{aligned} \tag{4.4}$$

where T is the number of the Monte Carlo samples and ϕ_n^t is the spectral discrepancy of the t^{th} Monte Carlo sample, given input instance \mathbf{i}_n . Furthermore, $\mathbb{I}(\cdot)$ is the indicator function and takes value 1 if $\phi_n^t < \phi^*$ and 0 otherwise. To obtain the samples we use the practical method proposed in [51, 50], which exploits dropout at test time to draw samples. ϕ^* is the decision parameter, which can be set to achieve a problem specific precision and/or recall requirements using different methods such as [99] or by using the training set (as will be discussed in Section 4.5).

There are some similarities between our approach and some of previous OOD

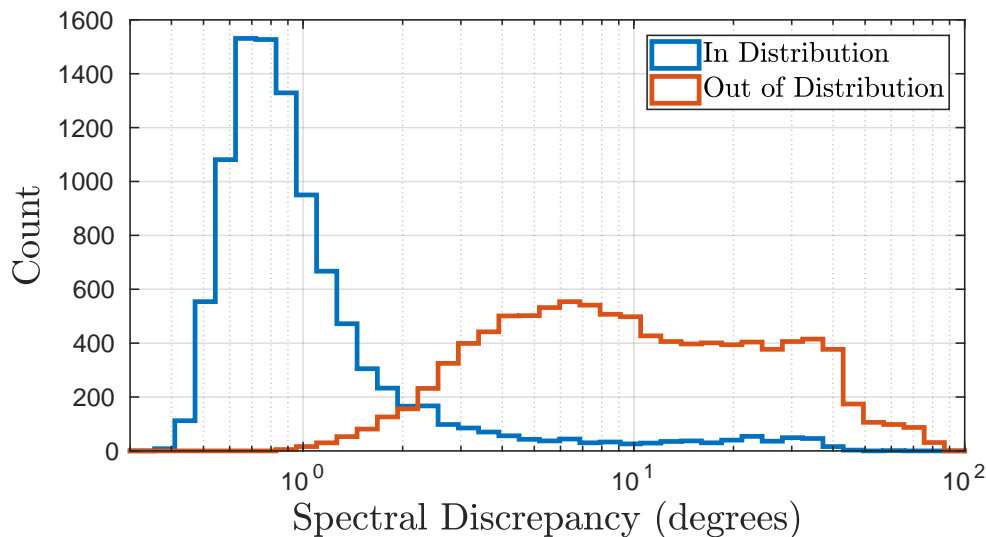


Figure 4.5: Histogram of the spectral discrepancy of the feature vectors corresponding to CIFAR10 test set (in-distribution) and TinyImagenet (out-of-distribution). The feature vectors are extracted from a WideResNet CNN backbone trained by enforcing the proposed structure.

detection frameworks, where the outlier samples are detected by computing how much they follow the structure of the inlier training samples. However, as discussed earlier, our enforced structure is more robust to perturbations in the training set. Furthermore, the first singular vector is a better representative of the class structure [163, 1], and therefore our proposed spectral discrepancy based on the first singular vector is a more reliable measure.

Figure 4.4 demonstrates the effectiveness of employing spectral discrepancy in distinguishing between in- and out-of-distribution samples. Similar to Figure 4.3, this figure shows a 3-dimensional representation of the features that are close to the first 3 classes of the CIFAR10, meaning that the classifier would classify them as one of these classes. The top row shows the features extracted from a plain WideResNet, without enforcing any structure. Also, comparing ID samples (Figure 4.4(a)) with OOD samples (Figure 4.4(b)), it is clear that both ID and OOD samples follow a very similar structure, which makes OOD detection more difficult. On the other hand, the bottom row illustrates the ℓ_2 -normalized features extracted from the WideResNet trained using our proposed structure. We plot the ℓ_2 -normalized features because our proposed spectral discrepancy in (4.3) only depends on the normalized features, i.e., $\frac{\mathbf{x}_n}{\|\mathbf{x}_n\|}$. Comparing the ID (Figure 4.4(c)) and OOD (Figure 4.4(d)) samples, it is easy to notice that most of the OOD samples have significantly larger angular distance to their closest singular vector, compared to the ID samples, which can be exploited to detect them more accurately.

To further demonstrate the discriminative ability of the proposed measure, Figure 4.5 compares the histogram of the spectral discrepancy of samples belonging to the ID (CIFAR10 test set) and OOD (TinyImagenet) datasets. It is evident that the spectral discrepancy of the ID samples are closer to 0, while it is very unlikely for OOD samples to have a spectral discrepancy smaller than a critical value.

Algorithm 1: Out-of-distribution detection using spectral discrepancy

Input: In-distribution training dataset, Number of classes L , OOD and in-distribution test dataset;
Output: OOD measure c_i for each sample of in-distribution and OOD datasets;
WideResNet and ResNet training;
while *Training* **do**
 Enforce intraclass structure: energy of each class L lies on a subspace
 Freeze weights in the last FC layer such that
 $\|\mathbf{W}\mathbf{W}^T - \mathbf{I}_L\| = 0$;
 Loss $\mathcal{L} = \frac{1}{N} \sum_{n=1}^N -\log\left(\frac{e^{|\cos(\theta_{i_n^*}^*)|}}{\sum_l e^{|\cos(\theta_{ln})|}}\right)$;
end
while *Test* **do**
 Extract features for training set and compute $v_1^{(l)}$ for each class in L
 Sample T feature vectors $\mathbf{x}_n^t, t = 1, \dots, T$ for the n^{th} test sample
 Compute ϕ_n^t for each sample \mathbf{x}_n^t
 Verify if $p(\phi_n \leq \phi^*) = 0$
end

4.5 Experiments

In this section, experimental settings and results are discussed. We report the details of our setup, namely in- and out-of-distribution dataset pairs, neural network architecture and evaluation metrics. We show how our method compares with respect to current state-of-the-art methods.

Neural Network Architecture: We deploy Wide ResNet [164] with depth 28, width 10, and dropout rate 0.3 as the backbone architecture for our method. All the network parameters are set as the original implementation in [164], except the last layer which is modified as discussed in Section 4.3. Stochastic gradient descent (SGD) is used to train the network for 200 epochs, by enforcing the structure discussed in Section 4.3. At the beginning of the training, the learning rate is set to 0.1 and it is then dropped by a factor of 10 at 50% and 75% of the progress. At the test time, unless otherwise stated, we draw 50 Monte Carlo samples to estimate $p(\phi_n \leq \phi^*)$ and to detect the OOD samples.

Datasets: We train the models on CIFAR-10 (contains 10 classes) and CIFAR-100 (contains 100 classes) [81] datasets, which respectively consist of 50,000 images for training and 10,000 images for testing, with an image size of 32×32 . The testing set is used as the in-distribution testing samples. For the out-of-distribution testing samples, we use the following datasets:

1. **TinyImagenet (TIN).** The Tiny ImageNet dataset [43] consists of 10,000 test images of size 36×36 belonging to 200 different classes, which are sampled from the original 1,000 classes of ImageNet [43]. As in [94, 151] we construct two datasets from TinyImagenet: TinyImagenet-crop (TINc) and TinyImagenet-resize (TINr), by either randomly cropping or downsampling each image to a size of 32×32 .

Training dataset	OOD dataset	FPR at 95% TPR ↓	Detection Error ↓	AUROC ↑	AUPR In ↑	AUPR Out ↑	F1 Score ↑
CIFAR10	TINc	11.1	8.0	97.2	97.7	96.5	91.7
	TINr	7.0	6.0	98.3	98.6	98.0	94.0
	LSUNc	10.5	7.6	97.5	98.0	97.07	92.2
	LSUNr	4.2	4.1	99.0	99.2	98.8	95.8
	UNFM	0.4	0.5	99.9	99.9	99.9	99.5
	GSSN	0.4	0.5	99.9	99.9	99.9	99.45
CIFAR100	TINc	45.1	21.5	86.8	88.4	85.0	78.5
	TINr	54.8	25.4	82.5	83.7	81.1	74.9
	LSUNc	48.0	21.7	85.6	87.0	83.0	78.1
	LSUNr	48.6	22.9	85.0	86.7	83.3	76.7
	UNFM	8.0	6.3	98.0	98.50	97.6	93.5
	GSSN	13.3	8.7	94.8	96.4	91.5	90.7

Table 4.1: Performance of the proposed framework for distinguishing in- and out-of-distribution test set data for the image classification task, using a WideResnet with depth 28 and width 10. \uparrow indicates larger value is better and \downarrow indicates lower value is better.

2. **LSUN.** The Large-scale Scene Understanding dataset (LSUN) [1] consists of 10,000 test images from 10 different scene categories. Like before, we randomly crop and downsample the LSUN test set to construct two datasets LSUN-crop (LSUNc) and LSUN-resize (LSUNr).
3. **Gaussian Noise.** The RGB values of every pixel in this synthetic dataset are sampled from a Gaussian distribution with mean 0.5 and variance 1, and then clipped into the range $[0, 1]$. Similar to previous datasets, this OOD set contains 10,000 32×32 images.
4. **Uniform Noise.** 10,000 32×32 images are generated by sampling the RGB values of every pixel from a uniform distribution on $[0, 1]$.

Baselines: We compare the performance of the proposed method with recent state-of-the-art techniques such as [102, 113, 161, 65]. These methods follow a similar settings and assumptions as ours. For a fair comparison, we do not compare with methods that use extra information, such as OOD samples, for either training or hyper-parameter tuning (e.g., [94, 162, 151, 84]).

Evaluation Metrics: We measure the effectiveness of our method in detecting in- and out-of-distribution samples using the same metrics as in [94, 151]. We use TP, TN, FP, FN to indicate true positives, true negatives, false positives, and false negatives, respectively.

1. **FPR at 95% TPR** indicates the false positive rate (FPR) at 95% true positive rate (TPR). True positive rate is defined as $TPR = TP / (TP+FN)$, and the false positive rate (FPR) is defined as $FPR = FP / (FP+TN)$.
2. **Detection Error** indicates the minimum misclassification probability. It is computed by the minimum misclassification rate over all possible critical spectral discrepancy values ϕ^* .
3. **AUROC**, defined as the Area Under the Receiver Operating Characteristic curve, is computed as the area under the FPR against TPR curve.

OOD dataset	TINc	TINr	LSUNc	LSUNr
In distribution dataset: CIFAR10				
Metric: AUROC				
SoftMax Pred. [65]	92.9	91.0	94.5	93.9
Ours	97.2	98.3	97.5	99.0
In distribution dataset: CIFAR10				
Metric: F1 score				
Counterfactual [113]	0.636	0.635	0.650	0.648
CROSR [161]	0.733	0.763	0.714	0.731
Ours	0.917	0.940	0.922	0.958
In distribution dataset: CIFAR100				
Metric: Detection Error				
Softmax Pred. [65]	35.8	42.1	39.5	43.6
LSLTR [102]	-	29.9	-	-
Ours	21.4	25.4	21.7	22.9

Table 4.2: A comparison of Out-of-Distribution detection results for different in- and out-of-distribution datasets. Results that surpass all competing methods are **bold**.

Structure	In Distribution Accuracy (%)	OOD Detection AUROC
No	96.0	95.2
Yes	95.4	98.3

Table 4.3: Ablation study of the proposed framework. The networks are trained on CIFAR10 and tested on CIFAR10 (in-distribution) and TINr (out-of-distribution). Spectral discrepancy is used in all the variants as the detection score.

4. **AUPR In**, defined as the Area Under the Precision-Recall curve, is computed as the area under the precision = $TP / (TP+FP)$ against the recall = $TP / (TP+FN)$ curve. For AUPR In, in-distribution images are treated as positive.
5. **AUPR Out** is similar to the metric AUPR-In. Opposite to AUPR In, out-of-distribution images are treated as positive in AUPR Out.
6. **F1 Score** is the maximum average F1 score over all possible critical spectral discrepancy values ϕ^* .

Table 4.1 summarizes the performance of the proposed technique over different combinations of the ID and OOD datasets. As expected, the synthetic OOD samples are easier to detect and our method consistently achieves F1 score of over 90% on these datasets. Furthermore, it is evident that our results are consistent over different real OOD datasets, meaning that our method can perform well for different types of OOD samples. To put these numbers into context, Table 4.2 compares our results with recent OOD detection techniques. Our proposed method is able to consistently outperform the competing methods over different datasets and evaluation metrics.

Table 4.3 investigates the impact of enforcing structure the on OOD detection

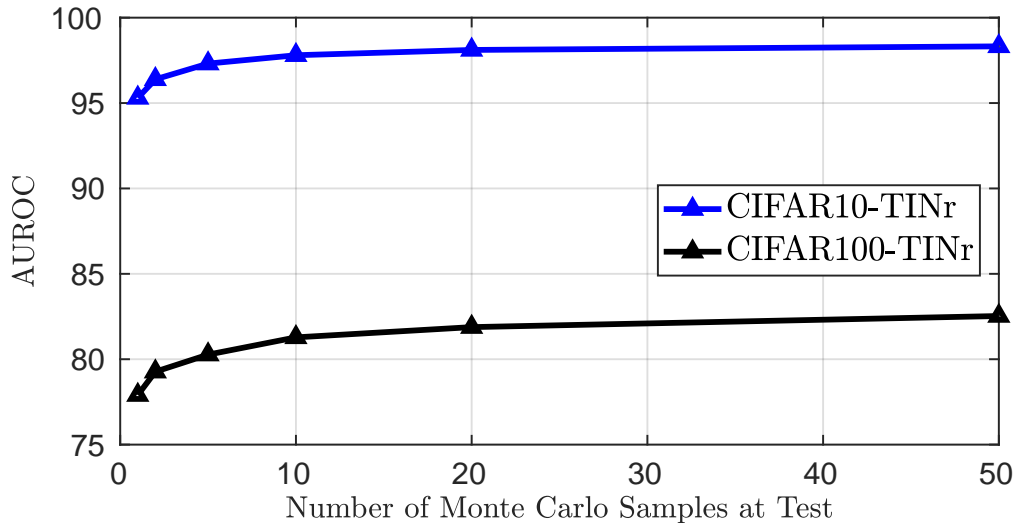


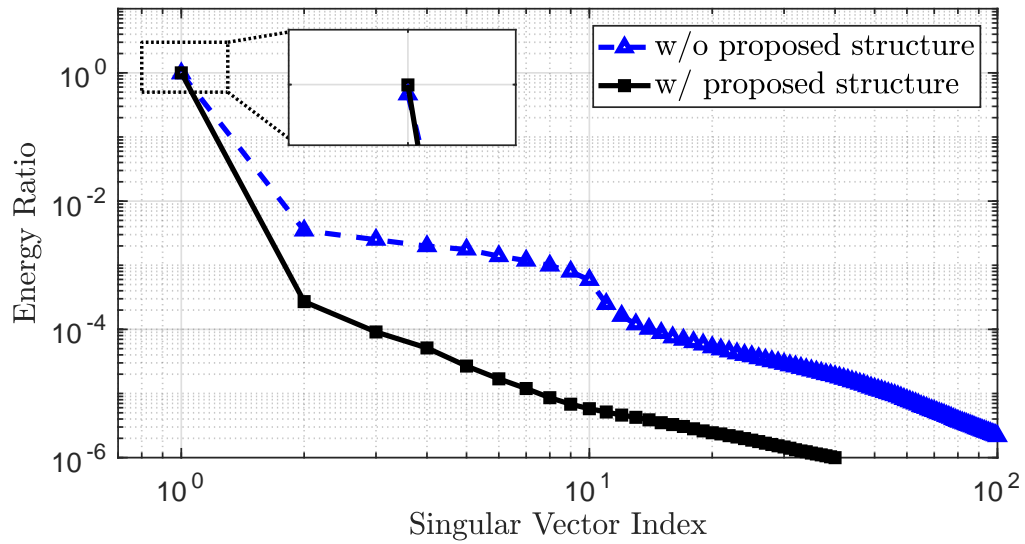
Figure 4.6: Area Under ROC curve using the proposed framework versus the number of the Monte Carlo samples used for estimating $p(\phi_n < \phi^*)$. The networks are trained on CIFAR10 and CIFAR100 and tested on TINr as the OOD dataset.

using spectral discrepancy. AUROC is calculated by using spectral discrepancy for the different variants. This table shows that while enforcing the proposed structure hurts the in-distribution classification accuracy and does not improve the representation ability of the network, it is an effective technique to distinguish between ID and OOD samples. This further confirms our hypothesis that the structure of the samples in the feature space plays an important role in OOD detection and the structure produced by conventional models is not necessarily suitable for this task.

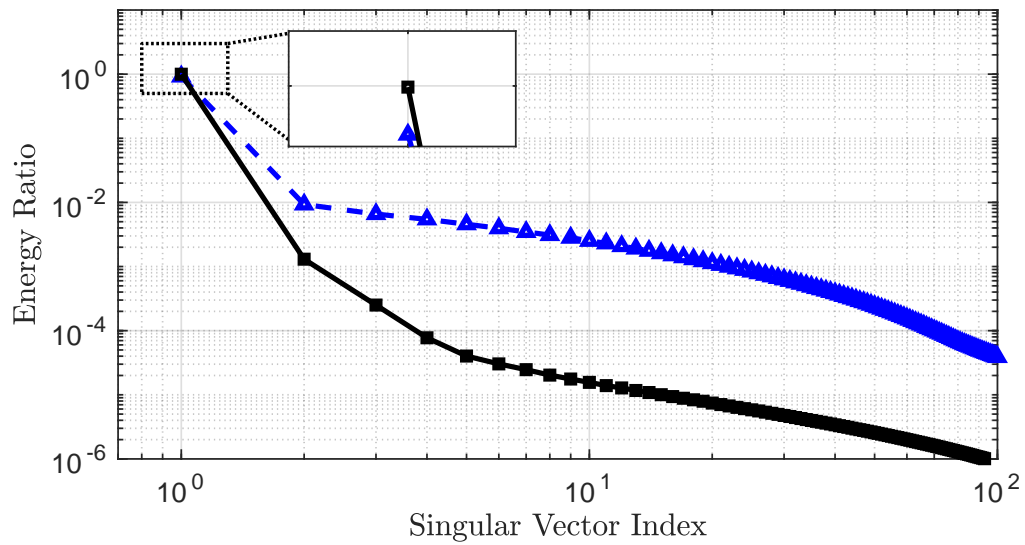
Figure 4.6 examines the number of Monte Carlo samples necessary for a good estimation of $p(\phi_n < \phi^*)$. It shows that having as low as 10 samples can improve the results significantly. However, as expected, having more samples always leads to better estimation and better performance. It is also worthwhile to mention that since the samples can be drawn concurrently, drawing more samples does not increase the running time much.

On the other hand, Figure 4.7 demonstrates the impact of the proposed training scheme on the structure of the feature vectors. This figure shows the percentage of the energy concentrated along each singular vector averaged over all the classes. The energy ratio along the i^{th} singular vector is calculated as $\frac{\lambda_i}{\sum_j \lambda_j}$. As discussed in Section 4.3, our goal is to make the feature vectors of each class to lie on a 1-dimensional subspace and to make the gap between the first eigenvalue λ_1 and other eigenvalues $\lambda_j, j > 1$ as large as possible. Figure 4.7 illustrates that the proposed training scheme can effectively achieve this by increasing the energy ratio along the first singular vector and reducing the energy concentrated along the rest of the singular vectors. Consequently, the first singular vector of each class will be more robust to outliers and a better representative of the class, which in turn makes the proposed spectral discrepancy a more reliable OOD detection measure.

Finally, as a guideline to set the value of the critical spectral discrepancy ϕ^* , Figure 4.8(a) shows the histogram of the spectral discrepancy for samples belonging

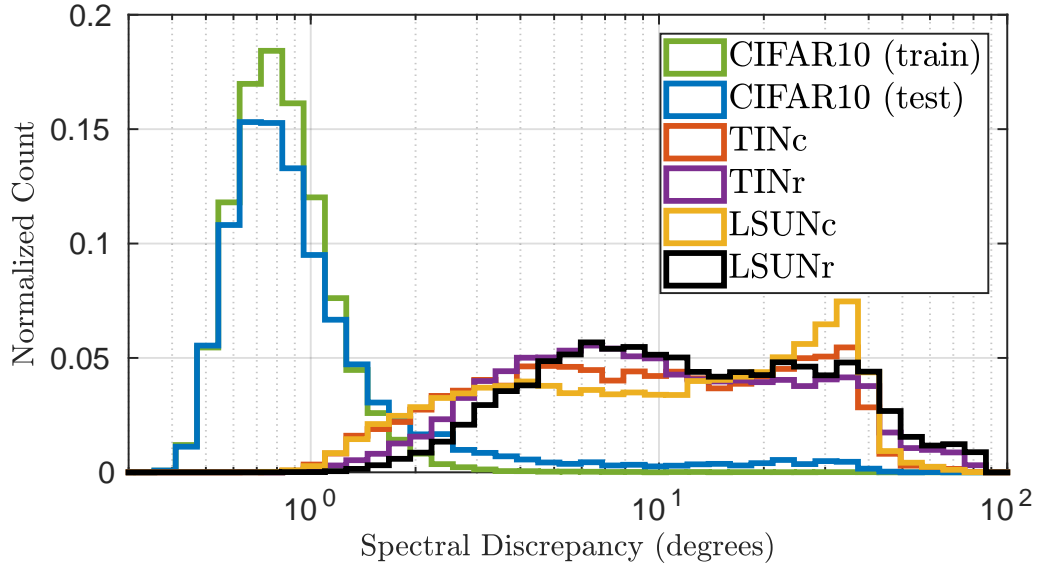


(a) CIFAR10

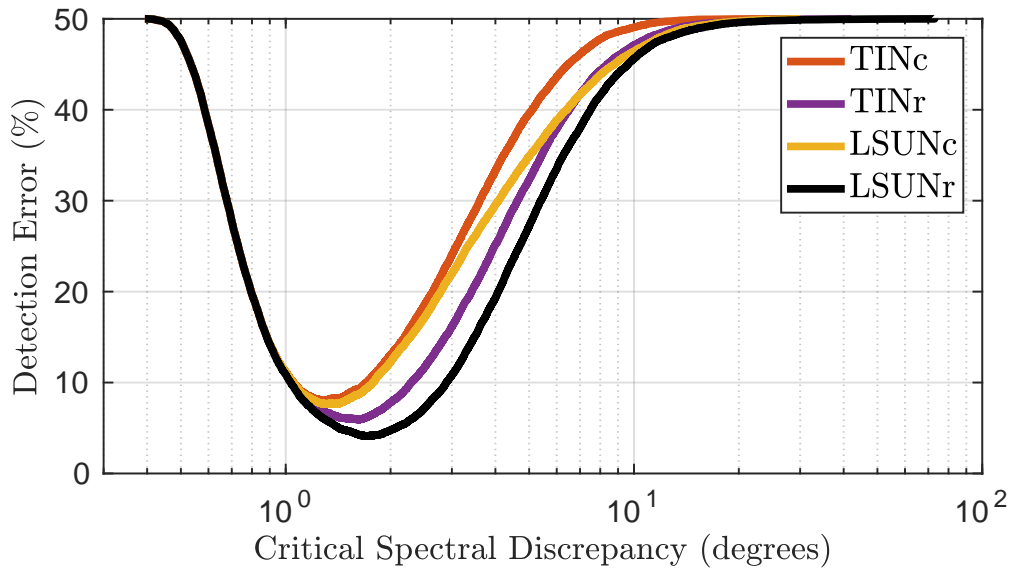


(b) CIFAR100

Figure 4.7: Energy Ratio of the training samples along the first 100 singular vectors of features extracted using plain WideResNet and the same network with our proposed structure trained on (a) CIFAR10 and (a) CIFAR100. Energy ratios are averaged over all the classes. The proposed structure increases the energy along the first singular vector from 98.3% to 99.9% for CIFAR 10 and from 91.8% to 99.8% for CIFAR100.



(a)



(b)

Figure 4.8: (a) Empirical probability distribution of the spectral discrepancy of samples belonging to CIFAR10 (in-distribution) and different out-of-distribution datasets. (b) out-of-distribution detection performance of the proposed method in terms of detection error for different values of critical spectral discrepancy ϕ^* . Both the spectral discrepancy histogram and the best ϕ^* do not change much for different datasets.

to CIFAR10, as the ID dataset, and different real OOD datasets. It is evident that samples from both the testing and training set of the ID dataset follow a very similar behaviour. Thus, the training set can be used to estimate the possible interval of spectral discrepancies for the ID samples. For instance, about 98% of the samples in CIFAR10 have a spectral discrepancy of less than 2 degrees. On the other hand, Figure 4.8(b) demonstrates the detection error for different values of the critical spectral discrepancy ϕ^* . This figure shows that best detection error is achieved by setting ϕ^* to a value in range $[1.3, 2]$ degrees, regardless of the OOD dataset. Hence, this figure shows that ϕ^* is not sensitive to the OOD dataset and can be set using only the training set. However, it should be mentioned that in general the best value for ϕ^* depends on the task at hand and the precision and/or recall requirements. As mentioned earlier, ϕ^* can also be set by many of the threshold estimation techniques such as [99].

4.6 Conclusions

In this chapter, we have argued how an anomaly detection problem can be posed as an out of distribution detection in the deep learning context. We argue that OOD samples can be detected far more easily if the training data is embedded into a low-dimensional space, such that the embedded training samples (or features) lie on a union of 1-dimensional subspaces. We show that such embedding of the in-distribution (ID) samples provides us with two main advantages. First, due to compact representation in the feature space, OOD samples are less likely to occupy the same region as the known classes. Second, the first singular vector of samples belonging to a 1-dimensional subspace is a robust representative of them. Our method does not require any extra information for hyperparameter tuning and does not rely on sample generation or reconstruction for OOD detection. The superiority of our proposed method is demonstrated by achieving new state-of-the-art results on various benchmark datasets.

5 Conclusions and Future Directions

In this work, we studied three main issues in crowd analysis, namely: (1) pedestrian trajectory prediction, (2) crowd simulation, and (3) anomalies detection as an out-of-distribution problem.

In the field of *trajectory prediction*, we proposed two approaches, a Group and Obstacle LSTM and a Group GAN. These methods focused on modeling the relationship between socially related and unrelated pedestrians. We showed how we can improve the results in the prediction task through better modeling of these relationships and of the interactions between pedestrians and obstacles around them. In the future development, we plan to extend the ability of our framework for longer-term predictions. The biggest limitation right now is the availability of long term footages and trajectories of pedestrians. This is due to the nature of fixed surveillance cameras, which have a limited Field of View, which cannot capture the trajectory of pedestrians passing by for a long period of time. We plan to collect a dataset of pedestrian trajectories in ego-vision. Using an ego-vision perspective would allow capturing long trajectories of the pedestrian carrying the camera while also providing a more human-like perspective on the interactions between who carries the camera and other pedestrians.

In the field of *crowd simulation*, we proposed two methods for improving the modeling of the high-level behavior of crowds. The first proposed data-driven approach to crowd simulation still relies on the Social Force Model for the local dynamics and a vector field for the global dynamics. Moreover, we proposed Virtual crowds, a deep-learning-based approach that showed how LSTM cells can be used to learn and simulate the motion of pedestrians. Then, we discussed how the appearance model of a virtual scene should be with respect to a real one. In order to guarantee the transferability between the simulated scenario and the real world, we need to account for a realistic camera capturing the scene. We presented Virtual Camera, which discussed how image sensors and lenses should be modeled in a simulated scenario. Finally, we showed how we can leverage on our simulated environment to deploy and test a crowd surveillance system. We deployed a network of heterogeneous cameras in a simulated environment, discussing different coverage policies and their trade-off. In future work, we plan to leverage our models of crowd dynamics to embed them in a more complex environment. A strong interest in the simulator field comes from the automotive industry, where there is a need for creating simulated scenarios to train and test self-driving cars. We plan to incorporate our crowd models in such a scenario, in order to have a more realistic and semantically rich simulation.

In the field of *anomalies detection*, we have presented a method for Out-of-distribution sample detection based on images. The main limitation of our approach is that it is image-based. Anomalies detection in crowded scenes should be performed

on videos rather than on images. Since our proposed approach directly operates in the feature space and it is agnostic to the input of the neural network, we plan to apply it to features extracted from video to try to detect anomalies in video footage of crowds.

Bibliography

- [1] Anish Agarwal, Devavrat Shah, Dennis Shen, and Dogyoon Song. On Robustness of Principal Component Regression. In *Advances in Neural Information Processing Systems 32*, pages 9889–9900. Curran Associates, Inc., 2019.
- [2] Javed Ahmed, MN Jafri, J Ahmad, and Muhammad I Khan. Design and implementation of a neural network for real-time object tracking. *Proceedings of World Enformatika Conference Machine Vision and Pattern Recognition*, 2005.
- [3] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.
- [4] Alexandre Alahi, Vignesh Ramanathan, and Li Fei-Fei. Socially-aware large-scale crowd forecasting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2203–2210, 2014.
- [5] Saad Ali and Mubarak Shah. A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007.
- [6] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, volume 3, page 6, 2018.
- [7] Gianluca Antonini, Michel Bierlaire, and Mats Weber. Discrete choice models of pedestrian walking behavior. *Transportation Research Part B: Methodological. Elsevier*, 40(8):667–687, 2006.
- [8] Pietro Azzari, Luigi Di Stefano, and Alessandro Bevilacqua. An effective real-time mosaicing algorithm apt to detect motion through background subtraction using a ptz camera. *Conference on Advanced Video and Signal Based Surveillance*, pages 511–516, 2005.
- [9] Timur Bagautdinov, Alexandre Alahi, François Fleuret, Pascal Fua, and Silvio Savarese. Social scene understanding: End-to-end multi-person action localization and collective activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4315–4324, 2017.

- [10] James Baglama and Lothar Reichel. Augmented implicitly restarted lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, 2005.
- [11] Lamberto Ballan, Francesco Castaldo, Alexandre Alahi, Francesco Palmieri, and Silvio Savarese. Knowledge transfer for scene-specific motion prediction. In *European Conference on Computer Vision*, pages 697–713. Springer, 2016.
- [12] Federico Bartoli, Giuseppe Lisanti, Lamberto Ballan, and Alberto Del Bimbo. Context-aware trajectory prediction. *arXiv preprint arXiv:1705.02503*, 2017.
- [13] Abhijit Bendale and Terrance E Boulton. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.
- [14] Abhijit Bendale and Terrance E. Boulton. Towards open set deep networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.
- [15] Aniket Bera, Sujeong Kim, and Dinesh Manocha. Efficient trajectory extraction and parameter learning for data-driven crowd simulation. In *Proceedings of the 41st Graphics Interface Conference*, pages 65–72. Canadian Information Processing Society, 2015.
- [16] Aniket Bera, Sujeong Kim, and Dinesh Manocha. Realtime anomaly detection using trajectory-level crowd behavior learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 50–57. 2016.
- [17] Aniket Bera, Tanmay Randhavan, and Dinesh Manocha. Aggressive, tense, or shy? identifying personality traits from crowd videos. *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 112–118, 2017.
- [18] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.
- [19] Alessandro Bevilacqua and Pietro Azzari. High-quality real time motion detection using ptz cameras. *International Conference on Video and Signal Based Surveillance*, pages 23–23, 2006.
- [20] Niccoló Bisagno and Nicola Conci. Virtual camera modeling for multi-view simulation of surveillance scenes. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2170–2174. IEEE, 2018.
- [21] Niccoló Bisagno, Nicola Conci, and Bernhard Rinner. Dynamic camera network reconfiguration for crowd surveillance. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, pages 1–6, 2018.
- [22] Niccoló Bisagno, Nicola Conci, and Bo Zhang. Data-driven crowd simulation. *Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, pages 1–6, 2017.

- [23] Niccoló Bisagno, Nicola Garau, Andrea Montagner, and Nicola Conci. Virtual crowds: An lstm-based framework for crowd simulation. In *International Conference on Image Analysis and Processing*, pages 117–127. Springer, 2019.
- [24] Niccolo Bisagno, Bo Zhang, and Nicola Conci. Group lstm: Group trajectory prediction in crowded scenarios. In *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.
- [25] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
- [26] Paulo Vinicius Koerich Borges, Nicola Conci, and Andrea Cavallaro. Video-based human behavior understanding: A survey. *IEEE transactions on circuits and systems for video technology*, 23(11):1993–2008, 2013.
- [27] Jean-Yves Bouguet. Camera calibration tool-box for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/, 2002.
- [28] Cheryl Campanella Bracken and Paul Skalski. Presence and video games: The impact of image quality and skill level. *Annual International Workshop on Presence*, pages 28–29, 2006.
- [29] Gary R Bradski. *Computer vision face tracking for use in a perceptual user interface*. Citeseer, 1998.
- [30] Duane C Brown. Decentering distortion of lenses. *Photogrammetric Engineering and Remote Sensing*, 1966.
- [31] Daguang Chen, Tao Zheng, and Hongcang Yang. Estimates of the gaps between consecutive eigenvalues of Laplacian. *Pacific Journal of Mathematics*, 2016.
- [32] Zhongwei Cheng, Lei Qin, Qingming Huang, Shuicheng Yan, and Qi Tian. Recognizing human group action by layered model with multiple cues. *Neuro-computing*, 136:124–135, 2014.
- [33] Ernest Cheung, Tsan Kwong Wong, Aniket Bera, Xiaogang Wang, and Dinesh Manocha. Lcrowdv: Generating labeled videos for simulation-based crowd behavior learning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2016.
- [34] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [35] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. *arXiv preprint arXiv:1412.1602*, 2014.

- [36] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [37] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.
- [38] P Comon and G H Golub. Tracking a few extreme singular values and vectors in signal processing. *Proceedings of the IEEE*, 78(8):1327–1343, 1990.
- [39] Nicola Conci, Niccoló Bisagno, and Andrea Cavallaro. On modeling and analyzing crowds from videos. In *Computer Vision for Assistive Healthcare*, pages 319–336. Elsevier, 2018.
- [40] Alexander Eugen Conrady. Decentred lens-systems. *Monthly notices of the royal astronomical society*, 79(5):384–390, 1919.
- [41] Nicolas Courty and Thomas Corpetti. Crowd motion capture. *Computer Animation and Virtual Worlds*, 18(4-5):361–370, 2007.
- [42] Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. *Advances in Neural Information Processing Systems (NIPS)*, pages 1504–1512, 2015.
- [43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 6 2009.
- [44] Akshay Raj Dhamija, Manuel Günther, and Terrance E. Boult. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems*, 2018.
- [45] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Being robust (in high dimensions) can be practical. In *34th International Conference on Machine Learning, ICML 2017*, 2017.
- [46] Funda Durupinar, Jan Allbeck, Nuria Pelechano, and Norman Badler. Creating crowd variation with the ocean personality model. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1217–1220. International Foundation for Autonomous Agents and Multiagent Systems, 2008.
- [47] Moataz M.H. El Ayadi, Mohamed S. Kamel, and Fakhri Karray. Toward a tight upper bound for the error probability of the binary Gaussian classification problem. *Pattern Recognition*, 2008.
- [48] Linan Feng and Bir Bhanu. Understanding dynamic social grouping behaviors of pedestrians. *IEEE Journal of Selected Topics in Signal Processing*, 9(2):317–329, 2015.

- [49] Gian Luca Foresti, Petri Mähönen, and Carlo S Regazzoni. *Multimedia video-based surveillance systems: Requirements, Issues and Solutions*, volume 573. Springer Science & Business Media, 2012.
- [50] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *PMLR*, 2016.
- [51] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian Active Learning with Image Data. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1183–1192, International Convention Centre, Sydney, Australia, 2017. PMLR.
- [52] Weina Ge, Robert T Collins, and R Barry Ruback. Vision-based analysis of small groups in pedestrian crowds. *IEEE transactions on pattern analysis and machine intelligence*, 34(5):1003–1016, 2012.
- [53] Julio Godoy, Ioannis Karamouzas, Stephen J Guy, and Maria L Gini. Moving in a crowd: Safe and efficient navigation among heterogeneous agents. *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 294–300, 2016.
- [54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [55] J.M. Grant and P.J. Flynn. Crowd scene understanding from video: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 13(2), 2017.
- [56] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- [57] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.
- [58] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [59] Stephen J Guy, Jur Van Den Berg, Wenxi Liu, Rynson Lau, Ming C Lin, and Dinesh Manocha. A statistical similarity measure for aggregate crowd dynamics. *Transactions on Graphics (TOG). ACM*, 31(6):190, 2012.
- [60] Edward T Hall, Ray L Birdwhistell, Bernhard Bock, Paul Bohannon, A Richard Diebold Jr, Marshall Durbin, Munro S Edmonson, JL Fischer, Dell Hymes, Solon T Kimball, et al. Proxemics [and comments and replies]. *Current anthropology*, 9(2/3):83–108, 1968.

- [61] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.
- [62] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.
- [63] Dirk Helbing, Anders Johansson, and Habib Zein Al-Abideen. Dynamics of crowd disasters: An empirical study. *Physical review E. APS*, 75(4):046109, 2007.
- [64] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.
- [65] Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *Proceedings of International Conference on Learning Representations*, 2017.
- [66] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [67] Min Hu, Saad Ali, and Mubarak Shah. Learning motion patterns in crowded scenes using motion flow field. In *ICPR*, pages 1–5, 2008.
- [68] Ronny Hug, Stefan Becker, Wolfgang Hübner, and Michael Arens. On the reliability of lstm-mdl models for pedestrian trajectory prediction. In *Proceedings of the VIIth International Workshop on Representation, analysis and recognition of shape and motion FroM Image Data (RFMI 2017), Póvoa de Varzim, Portugal*, pages 21–23, 2017.
- [69] Kenji Irie, Alan E McKinnon, Keith Unsworth, and Ian M Woodhead. A technique for evaluation of ccd video-camera noise. *Transactions on Circuits and Systems for Video Technology*, 18(2):280–284, 2008.
- [70] Julio Cezar Silveira Jacques, Adriana Braun, John Soldera, Soraia Raupp Musse, and Cláudio Rosito Jung. Understanding people motion in video sequences using voronoi diagrams. *Pattern Analysis and Applications*, 10(4):321–332, 2007.
- [71] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.
- [72] Michael Jones and Paul Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3(14):2, 2003.

- [73] Julio Cezar Silveira Jacques Junior, Soraia Raupp Musse, and Claudio Rosito Jung. Crowd analysis using computer vision techniques. *IEEE Signal Processing Magazine*, 27(5):66–77, 2010.
- [74] Sangkyu Kang, Joon-Ki Paik, Andreas Koschan, Besma R Abidi, and Mongi A Abidi. Real-time video tracking using ptz cameras. *International Conference on Quality Control by Artificial Vision*, 5132:103–112, 2003.
- [75] Asif Khan, Bernhard Rinner, and Andrea Cavallaro. Cooperative Robots to Observe Moving Targets: A Review. *IEEE Transactions on Cybernetics*, 48(1):187–198, 2018.
- [76] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [77] Ven Jyn Kok, Mei Kuan Lim, and Chee Seng Chan. Crowd behavior analysis: A review where physics meets biology. *Neurocomputing*, 177:342–362, 2016.
- [78] V.J. Kok, K.L. Mei, and C.S. Chan. Crowd behavior analysis: A review where physics meets biology. *Neurocomputing*, 177:342–362, 2016.
- [79] Krishna Reddy Konda and Nicola Conci. Optimal configuration of ptz camera networks based on visual quality assessment and coverage maximization. *International Conference on Distributed Smart Cameras*, pages 1–8, 2013.
- [80] Parth Kothari and Alexandre Alahi. Human trajectory prediction using adversarial loss. 2019.
- [81] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.
- [82] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.
- [83] Isah A Lawal, Fabio Poiesi, Davide Anguita, and Andrea Cavallaro. Support vector motion clustering. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(11):2395–2408, 2017.
- [84] Kimin Kibok Lee, Kimin Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 2018.
- [85] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [86] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

- [87] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer Graphics Forum*, volume 26, pages 655–664. Wiley Online Library, 2007.
- [88] Alon Lerner, Yiorgos Chrysanthou, Ariel Shamir, and Daniel Cohen-Or. Data driven evaluation of crowds. *International Workshop on Motion in Games*. Springer, pages 75–83, 2009.
- [89] Peter Lewis, Lukas Esterle, Arjun Chandra, Bernhard Rinner, and Xin Yao. Learning to be different: Heterogeneity and efficiency in distributed smart camera networks. In *Proc. IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 209–218, 2013.
- [90] T. Li, H. Chang, M. Wang, B.B. Ni, R.C. Hong, and S.C. Yan. Crowded scene analysis: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(3):367–386, 2015.
- [91] Teng Li, Huan Chang, Meng Wang, Bingbing Ni, Richang Hong, and Shuicheng Yan. Crowded scene analysis: A survey. *IEEE transactions on circuits and systems for video technology*, 25(3):367–386, 2015.
- [92] Yi Li, Marc Christie, Orianne Siret, Richard Kulpa, and Julien Pettré. Cloning crowd motions. *Eurographics Symposium on Computer Animation (SIGGRAPH)*. ACM, pages 201–210, 2012.
- [93] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2019.
- [94] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. In *International Conference on Learning Representations*, 2018.
- [95] Dahua Lin, Eric Grimson, and John Fisher. Learning visual flows: A lie algebraic approach. *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pages 747–754, 2009.
- [96] Dahua Lin, Eric Grimson, and John Fisher. Modeling and estimating persistent motion with geometric flows. *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pages 1–8, 2010.
- [97] Sheng-Fuu Lin, Jaw-Yeh Chen, and Hung-Xin Chao. Estimation of number of people in crowded scenes using perspective transformation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 31(6):645–654, 2001.
- [98] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

- [99] Si Liu, Risheek Garrepalli, Thomas G. Dietterich, Alan Fern, and Dan Hendrycks. Open category detection with PAC guarantees. In *35th International Conference on Machine Learning, ICML 2018*, 2018.
- [100] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. SphereFace: Deep hypersphere embedding for face recognition. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.
- [101] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2537–2546, 2019.
- [102] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-Scale Long-Tailed Recognition in an Open World. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2019.
- [103] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
- [104] Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M Kitani. A game-theoretic approach to multi-pedestrian activity forecasting. *arXiv preprint arXiv:1604.01431 v2*, 2016.
- [105] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1975–1981. IEEE, 2010.
- [106] Abraham Harold Maslow, Robert Frager, James Fadiman, Cynthia McReynolds, and Ruth Cox. *Motivation and personality*, volume 2. Harper & Row New York, 1970.
- [107] Ramin Mehran, Brian Moore, and Mubarak Shah. A streakline representation of flow in crowded scenes. *European Conference on Computer Vision*, pages 439–452, 2010.
- [108] Ramin Mehran, Alexis Oyama, and Mubarak Shah. Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 935–942. IEEE, 2009.
- [109] Stefano Messelodi and Carla Maria Modena. Boosting fisher vector based scoring functions for person re-identification. *Image and Vision Computing*, 44:44–58, 2015.
- [110] Christian Micheloni, Bernhard Rinner, and Gian Luca Foresti. Video Analysis in PTZ Camera Networks - From master-slave to cooperative smart cameras. *IEEE Signal Processing Magazine*, 27(5):78–90, 2010.

- [111] Mehdi Moussaïd, Niriaska Perozo, Simon Garnier, Dirk Helbing, and Guy Theraulaz. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one*, 5(4):e10047, 2010.
- [112] Soraia R Musse, Cláudio R Jung, Julio Jacques, and Adriana Braun. Using computer vision to simulate the motion of virtual agents. *Computer Animation and Virtual Worlds*, 18(2):83–93, 2007.
- [113] Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11210 LNCS, pages 620–635, 2018.
- [114] Sachin Patil, Jur Van Den Berg, Sean Curtis, Ming C Lin, and Dinesh Manocha. Directing crowd simulations using navigation fields. *Transactions on Visualization and Computer Graphics. IEEE*, 17(2):244–254, 2011.
- [115] Nuria Pelechano, Jan M Allbeck, and Norman I Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108. Eurographics Association, 2007.
- [116] Nuria Pelechano, Jan M Allbeck, and Norman I Badler. Virtual crowds: Methods, simulation, and control. *Synthesis Lectures on Computer Graphics and Animation*, 3(1):1–176, 2008.
- [117] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You’ll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, IEEE 12th International Conference on*, pages 261–268. IEEE, 2009.
- [118] Fatih Porikli. Achieving real-time object detection and tracking under extreme conditions. *Journal of Real-Time Image Processing*, 1(1):33–40, 2006.
- [119] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.
- [120] F. Z. Qureshi and D. Terzopoulos. Surveillance in virtual reality: System design and multi-camera control. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
- [121] Faisal Qureshi and Demetri Terzopoulos. Smart camera networks in virtual reality. *Proceedings of the IEEE*, 96(10):1640–1656, 2008.
- [122] Faisal Z Qureshi and Demetri Terzopoulos. Surveillance camera scheduling: A virtual vision approach. *Multimedia systems*, 12(3):269–283, 2006.

- [123] Faisal Z Qureshi and Demetri Terzopoulos. Surveillance in virtual reality: System design and multi-camera control. *Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
- [124] Sidney F Ray. *Applied photographic optics: Lenses and optical systems for photography, film, video, electronic and digital imaging*. Focal Press, 2002.
- [125] Martin Reisslein, Bernhard Rinner, and Amit Roy-Chowdhury. Smart Camera Networks. *IEEE Computer*, 47(5):23–25, 2014.
- [126] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [127] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. *European Conference on Computer Vision*, pages 102–118, 2016.
- [128] Bernhard Rinner, Lukas Esterle, Jennifer Simonjan, Georg Nebehay, Roman Pflugfelder, Gustavo Fernandez Dominguez, and Peter R Lewis. Self-aware and self-expressive camera networks. *IEEE Computer*, 48(7):21–28, 2014.
- [129] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proceedings of the European Conference on Computer Vision*, pages 549–565. Springer, 2016.
- [130] Mikel Rodriguez, Josef Sivic, Ivan Laptev, and Jean-Yves Audibert. Data-driven crowd analysis in videos. In *Computer vision (ICCV), 2011 IEEE international conference on*, pages 1235–1242. IEEE, 2011.
- [131] Allison Ryan, Marco Zennaro, Adam Howell, Raja Sengupta, and J Karl Hedrick. An overview of emerging results in cooperative uav control. In *Proc. 43rd IEEE Conference on Decision and Control*, volume 1, pages 602–607, 2004.
- [132] MS Ryoo and JK Aggarwal. Stochastic representation and recognition of high-level group activities. *International journal of computer Vision*, 93(2):183–200, 2011.
- [133] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. *arXiv preprint arXiv:1806.01482*, 2018.
- [134] Amir Sadeghian, Ferdinand Legros, Maxime Voisin, Ricky Vesel, Alexandre Alahi, and Silvio Savarese. Car-net: Clairvoyant attentive recurrent network. In *The European Conference on Computer Vision (ECCV)*, September 2018.
- [135] Paul Scovanner and Marshall F Tappen. Learning pedestrian dynamics from the real world. *International Conference on Computer Vision (ICCV)*. IEEE, pages 381–388, 2009.

- [136] Mubarak Shah, Omar Javed, and Khurram Shafique. Automated visual surveillance in realistic scenarios. *IEEE MultiMedia*, 14(1), 2007.
- [137] Jing Shao, Chen-Change Loy, Kai Kang, and Xiaogang Wang. Slicing convolutional neural network for crowd video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5620–5628. 2016.
- [138] Jing Shao, Chen Change Loy, and Xiaogang Wang. Learning scene-independent group descriptors for crowd understanding. *IEEE transactions on circuits and systems for video technology*, 27(6):1290–1303, 2017.
- [139] Berkan Solmaz, Brian E Moore, and Mubarak Shah. Identifying behaviors in crowd scenes using stability analysis for dynamical systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):2064–2070, 2012.
- [140] Sybren Stuvell, Nadia Magnenat-Thalmann, Daniel Thalmann, A Frank van der Stappen, and Arjan Egges. Torso crowds. *IEEE transactions on visualization and computer graphics*, 2016.
- [141] Meng Keat Christopher Tay and Christian Laugier. Modelling smooth paths using gaussian processes. *Field and Service Robotics. Springer*, pages 381–390, 2008.
- [142] Geoffrey R Taylor, Andrew J Chosak, and Paul C Brewer. Ovvv: Using virtual worlds to design and evaluate surveillance systems. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [143] Daniel Thalmann. *Crowd simulation*. Wiley Online Library, 2007.
- [144] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. 1991.
- [145] Peter Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: the case for cooperation. *International Conference on Robotics and Automation (ICRA). IEEE*, pages 2153–2160, 2013.
- [146] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160–1168, 2006.
- [147] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.
- [148] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. IEEE International Conference on*, pages 1928–1935. IEEE, 2008.
- [149] D. Varshneya and G. Srinivasaraghavan. Human trajectory prediction using spatially aware deep attention models. <https://arxiv.org/abs/1705.09436>, 2017.

- [150] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.
- [151] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L. Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.
- [152] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. CosFace: Large Margin Cosine Loss for Deep Face Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.
- [153] Benjamin Watson, Alinda Friedman, and Aaron McGaffey. Measuring and predicting visual fidelity. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 213–220. ACM, 2001.
- [154] David Wolinski, S J Guy, A-H Olivier, Ming Lin, Dinesh Manocha, and Julien Pettré. Parameter estimation and comparative evaluation of crowd simulations. *Computer Graphics Forum. Wiley Online Library*, 33(2):303–312, 2014.
- [155] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.
- [156] Kota Yamaguchi, Alexander C Berg, Luis E Ortiz, and Tamara L Berg. Who are you with and where are you going? In *CVPR 2011*, pages 1345–1352. IEEE, 2011.
- [157] Evşen Yanmaz, Saeed Yahyanejad, Bernhard Rinner, Hermann Hellwagner, and Christian Bettstetter. Drone networks: Communications, coordination, and sensing. *Ad Hoc Networks*, 68:1–15, 2018.
- [158] Hongxun Yao, Andrea Cavallaro, Thierry Bouwmans, and Zhengyou Zhang. Guest editorial introduction to the special issue on group and crowd behavior analysis for intelligent multicamera video surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):405–408, 2017.
- [159] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Understanding pedestrian behaviors from stationary crowd groups. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3488–3496, 2015.
- [160] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Pedestrian behavior modeling from stationary crowds with applications to intelligent surveillance. *IEEE transactions on image processing*, 25(9):4354–4368, 2016.

- [161] Ryota Yoshihashi, Shaodi You, Wen Shao, Makoto Iida, Rei Kawakami, and Takeshi Naemura. Classification-Reconstruction Learning for Open-Set Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [162] Qing Yu and Kiyoharu Aizawa. Unsupervised Out-of-Distribution Detection by Maximum Classifier Discrepancy. In *The IEEE International Conference on Computer Vision (ICCV)*, 10 2019.
- [163] Alireza Zaeemzadeh, Mohsen Joneidi, Nazanin Rahnavard, and Mubarak Shah. Iterative Projection and Matching: Finding Structure-Preserving Representatives and Its Application to Computer Vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5414–5423, 2019.
- [164] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *British Machine Vision Conference 2016, BMVC 2016*, 2016.
- [165] Zhengyou Zhang. A flexible new technique for camera calibration. *Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.
- [166] Mingbi Zhao, Stephen John Turner, and Wentong Cai. A data-driven crowd simulation model based on clustering and classification. *International Symposium on Distributed Simulation and Real Time Applications. IEEE*, pages 125–134, 2013.
- [167] Jinghui Zhong, Wentong Cai, Linbo Luo, and Haiyan Yin. Learning behavior patterns from video: A data-driven framework for agent-based crowd modeling. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 801–809. International Foundation for Autonomous Agents and Multiagent Systems, 2015.
- [168] Bolei Zhou, Xiaoou Tang, and Xiaogang Wang. Coherent filtering: Detecting coherent motions from crowd clutters. *Computer Vision–ECCV*, pages 857–871, 2012.
- [169] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

List of Publications

Journal publications and book chapters

- Conci, Nicola, Niccoló Bisagno, and Andrea Cavallaro. "On modeling and analyzing crowds from videos." *Computer Vision for Assistive Healthcare*. Academic Press, 2018. 319-336.

Conference publications

- Bisagno, Niccoló, Bo Zhang, and Nicola Conci. "Group lstm: Group trajectory prediction in crowded scenarios." *Proceedings of the European conference on computer vision (ECCV)*. 2018.
- Bisagno, Niccoló, Nicola Conci, and Bo Zhang. "Data-Driven crowd simulation." *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2017.
- Bisagno, Niccoló, and Nicola Conci. "Virtual camera modeling for multi-view simulation of surveillance scenes." *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018.
- Bisagno, Niccoló, Nicola Conci, and Bernhard Rinner. "Dynamic Camera Network Reconfiguration for Crowd Surveillance." *Proceedings of the 12th International Conference on Distributed Smart Cameras (ICDSC)*. 2018.
- Bisagno, Niccoló, Nicola Garau, Andrea Montagner, and Nicola Conci. "Virtual Crowds: An LSTM-Based Framework for Crowd Simulation." *International Conference on Image Analysis and Processing (ICIAP)*. Springer, Cham, 2019.

Abstracts and demos

- Rossi, Giulia, Niccoló Bisagno, and Aronne Armanini. "Dry granular flows of monodisperse particles: an optical method to compute the particles flow field." *EGU General Assembly Conference Abstracts*. Vol. 20. 2018.
- Bisagno, Niccoló, and Cristian Iacovlev. "Camera network optimization: maximize coverage in a 3D virtual environment." *Proceedings of the 13th International Conference on Distributed Smart Cameras (ICDSC)*. 2019.