Scalable Methods For Single and Multi Camera Trajectory Forecasting

by

Olly Styles

Thesis

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

Department of Computer Science

September 2021

Contents

List of Tables

List of Figures

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisors, Dr. Victor Sanchez and Dr. Tanaya Guha. Their patience and encouragement throughout my PhD studies made this thesis possible. I am immensely grateful to them for treating every new idea with enthusiasm and an open mind, which made my research experience so positive and rewarding.

I would like to thank my friends who have made my time at The University of Warwick so enjoyable. I am also grateful to my friends at The University of Sydney, Michigan State University, and Nanyang Technological University for welcoming me during my brief visits with such kindness.

Finally, I would like to thank my parents, Conrad and Jane Styles. I owe my achievements to their love and support.

Declarations

I hereby declare that this dissertation is original work and has not been submitted for a degree or diploma or other qualification at any other University. Parts of this thesis have been previously published by the author in the following:

- [1] Olly Styles, Arun Ross, and Victor Sanchez. Forecasting pedestrian trajectory with machine-annotated training data. In *Intelligent Vehicles Symposium (IV)*, 2019
- [2] Olly Styles, Tanaya Guha, and Victor Sanchez. Multiple object forecasting: Predicting future object locations in diverse environments. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020
- [3] Olly Styles, Tanaya Guha, Victor Sanchez, and Alex Kot. Multi-camera trajectory forecasting: Pedestrian trajectory prediction in a network of cameras. In *Computer Vision and Pattern Recognition Workshops (CVPR-W)*, 2020
- [4] Olly Styles, Tanaya Guha, and Victor Sanchez. Multi-camera trajectory forecasting with trajectory tensors. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021

Abstract

Predicting the future trajectory of objects in video is a critical task within computer vision with numerous application domains. For example, reliable anticipation of pedestrian trajectory is imperative for the operation of intelligent vehicles and can significantly enhance the functionality of advanced driver assistance systems. Trajectory forecasting can also enable more accurate tracking of objects in video, particularly if the objects are not always visible, such as during occlusion or entering a blind spot in a non-overlapping multi-camera network. However, due to the considerable human labour required to manually annotate data amenable to trajectory forecasting, the scale and variety of existing datasets used to study the problem is limited.

In this thesis, we propose a set of strategies for pedestrian trajectory forecasting. We address the lack of training data by introducing a scalable machine annotation scheme that enables models to be trained using a large Single-Camera Trajectory Forecasting (SCTF) dataset without human annotation. Using newly collected datasets annotated using our proposed methods, we develop two models for SCTF. The first model, Dynamic Trajectory Predictor (DTP), forecasts pedestrian trajectory from on board a moving vehicle up to one second into the future. DTP is trained using both human and machine-annotated data and anticipates dynamic motion that linear models do not capture. Our second model, Spatio-Temporal Encoder-Decoder (STED), predicts full object bounding boxes in addition to trajectory. STED combines visual and temporal features to model both object-motion and ego-motion.

In addition to our SCTF contributions, we also introduce a new task: Multi-Camera Trajectory Forecasting (MCTF), where the future trajectory of an object is predicted in a network of cameras. Prior works consider forecasting trajectories in a single camera view. Our work is the first to consider the challenging scenario of forecasting across multiple non-overlapping camera views. This has wide applicability in tasks such as re-identification and multi-target multi-camera tracking. To facilitate research in this new area, we collect a unique dataset of multi-camera pedestrian trajectories from a network of 15 synchronized cameras. We also develop a semi-automated annotation method to accurately label this large dataset containing 600 hours of video footage. We introduce an MCTF framework that simultaneously uses all estimated relative object locations from several camera viewpoints and predicts the object's future location in all possible camera viewpoints. Our framework follows a *Which-When-Where* approach that predicts in *which* camera(s) the objects appear and *when* and *where* within the camera views they appear. Experimental results demonstrate the effectiveness of our MCTF model, which outperforms existing SCTF approaches adapted to the MCTF framework.

Sponsorships and Grants

All research works contained within are funded by the UK EPSRC (grant no. EP/L016400/1).

Portions of this work were completed during secondments at The University of Sydney, Michigan State University, and Nanyang Technological University. The secondments were funded by the EU Horizon 2020 project IDENTITY (Project No. 690907).

The Nanyang Technological University phase of this research is also partly supported by the National Research Foundation, Singapore, under its AI Singapore Programme (Award no. AISG-100E-2018-018).

The experimentation and development of methods proposed in this thesis was supported in part by hardware provided by NVIDIA.

Acronyms

ADAS	Advanced Driver Assistance System.
ADE	Average Displacement Error.
AIOU	Average Intersection-Over-Union.
AP	Average Precision.
AV	Autonomous Vehicle.
BEV	Bird's Eye-View.
CA	Constant Acceleration.
CCTV	Closed-Circuit Television.
CNN	Convolutional Neural Network.
CV	Constant Velocity.
CVAE	Conditional Variational Auto-Encoder.
DTP	Dynamic Trajectory Predictor.
FC	Fully-Connected.
FDE	Final Displacement Error.
FIOU	Final Intersection-Over-Union.
FPS	Frames Per Second.
GAN	Generative Adversarial Network.
GPS	Global Positioning System.
GRU	Gated Recurrent Unit.
HOG	Histogram of Oriented Gradients.
IOU	Intersection-Over-Union.

IP Internet Protocol.

KLT Kanade–Lucas–Tomasi.

LDS Linear Dynamic System.

LiDAR Light Detection and Ranging.

LKF Linear Kalman Filter.

LSTM Long Short-Term Memory.

mAP Mean Average Precision.

MCTF Multi-Camera Trajectory Forecasting.

MOF Multiple Object Forecasting.

MOT Multi-Object Tracking.

MOTF Multiple Object Tracking and Forecasting.

MSE Mean Squared Error.

MT-MCTF Multi-Target Multi-Camera Trajectory Forecasting.

NTP Network Time Protocol.

Re-ID Re-Identification.

RGB Red, Green, Blue.

RNN Recurrent Neural Network.

ROC Receiver Operating Characteristic.

SCTF Single-Camera Trajectory Forecasting.

SIOU Soft Intersection-Over-Union.

SLDS Switching Linear Dynamic System.

SOT Single Object Tracking.

STED Spatio-Temporal Encoder-Decoder.

tCNN Transposed Convolutional Neural Network.

TTC Time To Collision.

VAE Variational Auto-Encoder.

VRU Vulnerable Road Users.

WNMF Warwick-NTU Multi-Camera Forecasting.

Symbols

$\|\|a-b\|\|_2$	The euclidean distance (ℓ_2) between two locations, a and b
$\min(\mathcal{S})$	The minimum value in set $\mathcal{S}$
t	A single timestep
L_t	An object location at timestep t, comprised of a tuple (x, y)
$\hat{L_t}$	An object location predicted by a model
$L_{a:b}$	A sequence of locations from timestep a to timestep b
k	The number of trajectories predicted by a multi-modal trajectory prediction model
$\mathcal{C}$	The set of cameras in a multi-camera system
c	A single camera in a multi-camera system
$\|\mathcal{C}\|$	The number of cameras in a multi-camera system
$\mathcal{C}^+$	The set of cameras in which the target appears
$\mathcal{I}$	The set of all objects-of-interest i in a video
w_t	Width of an object bounding box
h_t	Height of an object bounding box
b_t^i	A bounding box for object i at timestep t, comprised of the center, width, and height coordinates (x_t, y_t, w_t, h_t)
f_t	A video frame at timestep t
F_t	An optical flow frame timestep t
E_t^i	The first 20 bounding boxes of an object track
D_t^i	The last 20 bounding boxes of an object track
C_E	The camera associated with an entrance tracklet E_t^i
C_D	The camera associated with an departure tracklet D_t^i
t_E	The first timestep of an entrance tracklet
t_D	The final timestep of an departure tracklet
$R(a)$	The RE-ID features for tracklet a
δ	Tracklet similarity threshold
γ	Maximum camera transition time
$\mathcal{P}$	A set of cross-camera transition pairs, comprised of entrance tracklet and departure tracklet tuples $\{(E^t, D^t)\}$
v_t^x	Object velocity along the x-axis

v_t^y	Object velocity along the y-axis
v_t	Object velocity, comprised of v_t^x and v_t^y
P_t	A vector representing the residual between a ground truth object location and the location predicted by the constant velocity assumption.
h	A hidden state vector.
ℓ_1	Mean absolute error loss
ℓ_2	Mean squared error loss
$\boldsymbol{d_i}$	An detection object i with associated bounding box b
$\mathbf{Z}$	A Trajectory Tensor
$\boldsymbol{H}_i$	A Heatmap for object i
$\mathcal{T}$	A set of future timesteps to be predicted
$\mathcal{T}^+$	A set of future timesteps in which the target appears
$\mathcal{I}$	A set of grid cells
$\mathcal{I}^+$	A set of grid cells in which the target appears

Chapter 1

Introduction

Much work in computer vision and robotics has focused on building perception systems that make sense of the world. These systems range from understanding the environment through classifying images, tracking objects, segmenting scenes, and many others. Substantial progress has been made in the perception task, opening new opportunities surrounding how to use these perceptions effectively. Significant challenges remain in the step of transforming perceptions into useful, actionable information. Among these is the challenge of vision-based forecasting: reasoning about the future given the perception information available in the present.

1.1 Anticipating the Future

The ability to use information to reason about the future is a critical problem in many intelligent systems, and some define this ability to be one of the main components of intelligence [5]. The task has been long studied in many contexts, from weather forecasting [6] to market demand [7], stock market predictions [8] to patient outcome prediction [9]. Vision-based forecasting, specifically, is a difficult task for several reasons [10]. Firstly, as many vision-based forecasting systems rely on information from perception systems such as object detectors, errors in perception may be propagated forward. Secondly, the future is inherently uncertain and can prove challenging to model effectively. Thirdly, vision-based forecasting datasets are limited in both size and variety. These challenges have limited progress in vision-based forecasting; however, researchers have devoted more attention to the task after considerable advancements in vision-based perception.

In this thesis, we explore the area of vision-based forecasting in the context of video data. Specifically, we explore the nascent area of video-based human trajectory forecasting. Given a video containing individuals moving through an environment, human trajectory forecasting is the task of predicting the future

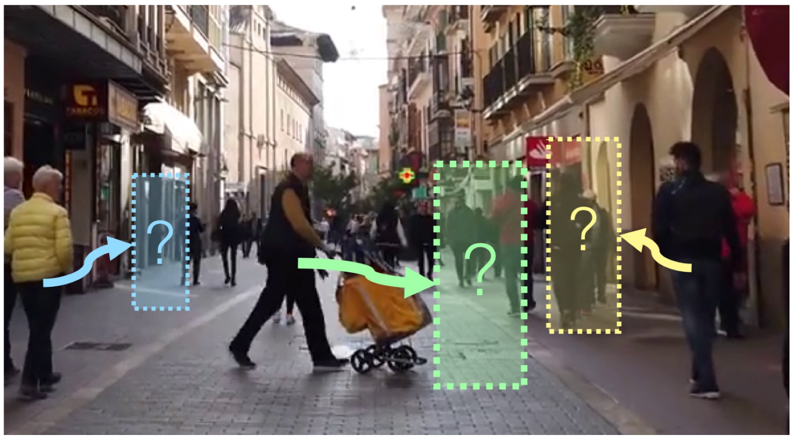


Figure 1.1: **Where will they move next?** Humans effortlessly anticipate the future of objects by relying on prior experiences. How can a machine accomplish the same task?

location of each individual. A visual representation of trajectory forecasting from an egocentric (object-level) viewpoint is shown in Figure 1.1. A history of past observations is used to predict a future trajectory, generally in the order of seconds. This is a task that humans do effortlessly yet proves challenging for machines.

1.2 Motivations

Making progress in human trajectory forecasting will unlock new possibilities in a variety of applications. One of the most promising and widely researched applications is Autonomous Vehicles (AVs), where the prediction of the intent of Vulnerable Road Users (VRU) (i.e., pedestrians and cyclists) is a critical safety feature and one of the key technical challenges that remains unsolved [11]. The extent to which a vehicle is considered autonomous is classified into 5 levels according to the SAE J3016 standard [12]. At level 0, a vehicle is entirely operated by a human driver. At level 1, vehicles have some level of driver assistance, such as cruise control and anti-locking braking systems. Level 2 vehicles are able to drive fully autonomously in some scenarios, such as on highways, but a human driver must constantly supervise the vehicle and be able to take over control at any moment. At level 3, a vehicle is able to drive autonomously in some scenarios, and a human driver still needs to be able to take control of the vehicle when alerted. However, the human driver does not need to monitor the vehicle continually and is given notice when manual takeover is required. Level 4 vehicles do not require any human driver for safe operation but are limited to a particular geographical area. Level 5 vehicles are considered fully autonomous and can operate without a human driver in all environments and in all conditions.

Human trajectory forecasting systems can be applied in vehicles with autonomy levels 0 and higher. At automation level 0, a human driver may be alerted if a potential collision with a VRU is detected by on-board sensors equipped with trajectory forecasting capabilities [13]. In level 2 vehicles, systems may use the same system to apply automated braking or steering in dangerous scenarios [14]. At level 3 and above, on-board sensors and trajectory forecasting systems can be used for full motion planning and control of the vehicle [15].

Trajectory forecasting has also been used to improve object tracking [16] and crowd monitoring [17]. In their recent survey on trajectory forecasting [18], Rudenko et al. identify five surveillance applications for trajectory forecasting: person retrieval, perimeter protection, traffic monitoring, crowd management, and retail analytics. The trajectory forecasting task is relatively easy for humans to perform well in a variety of situations. Consider the scene in Figure 1.1. Prior experience informs our predictions about where the people in this scene are likely to move in the next few seconds. For example, in this scene, we may look to the surrounding environmental context, the pedestrians' gait, and known priors about how pedestrians tend to navigate a crowded street through personal experience. This reasoning occurs quickly and without significant effort for humans but remains challenging to solve computationally.

Rasouli et al. investigate the interaction between pedestrians and drivers [19, 20]. The authors find that pedestrians look at the vehicle in over 90% of all crossing events on unmarked crossings, suggesting that visual attention is a crucial indicator of crossing intent. The authors also investigate the influence of the time between consecutive vehicles, known as gap acceptance, on pedestrian crossing behaviour. Very few pedestrians will cross when the size of the gap between consecutive vehicles, also known as Time To Collision (TTC), is less than 3 seconds, while almost all pedestrians will cross when the TTC is 7 seconds or more. These findings are consistent with others [21, 22]. Events between these TTC values are therefore of most interest, as some pedestrians will cross whereas others will not.

Let us consider more precisely which features may be useful for trajectory prediction. Schmidt et al. investigated how well humans could predict whether a pedestrian will cross the street given footage from a few seconds prior to the event [21]. Some participants were provided with an original video, while others were given footage with various parts masked to assess the importance of each of these features. Examples of these image masks are shown in Figure 1.2. The authors found that all masks resulted in poorer judgment by the participants, and full body occlusion leads to considerable performance degradation. The full body occlusion mask if the worst performing using trajectory information alone without any additional visual cues. This finding suggests that visual

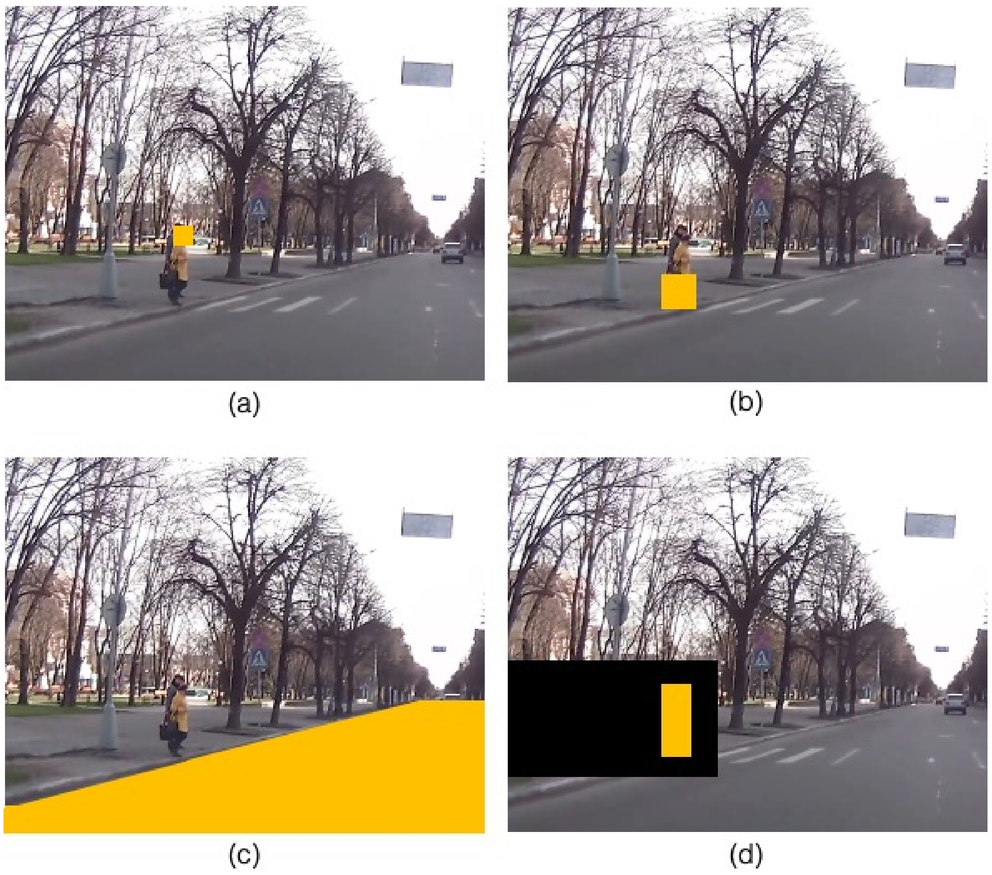


Figure 1.2: **What information do we need to predict if a pedestrian wants to cross?** (a) Head, (b) lower-body, (c) road, and (d) full occlusion of visual features leads to confusion for human observers. Figure recreated from Schmidt et al. [21].

information, in addition to the trajectory, informed the participants of the pedestrian's future movements. However, few existing trajectory forecasting methods consider visual information. The object's past trajectory and trajectories of other nearby objects are often the only inputs to existing trajectory forecasting models.

Detecting a human's current location is a prerequisite for real-world applications of human trajectory forecasting models. The task of pedestrian detection, that is, identifying the location of all humans in an image, has seen considerable breakthroughs in the past decade, and state-of-the-art models are now approaching human-level performance [23]. These breakthroughs are in part responsible for the opportunity to develop forecasting models that predict future pedestrian locations given detections obtained from a pedestrian detector. Pedestrian detection errors will propagate through a forecasting model, so accurate detection is critical.

1.3 Challenges

Despite the substantial progress in trajectory forecasting in recent years, several technical challenges remain. We identify four main challenges as follows:

Lack of large labelled datasets. Although attention to trajectory forecasting has grown, the field suffers from a lack of datasets. Labelling data for trajectory forecasting is a tedious and labour intensive process. For each frame in the video, each object must have its current position labelled. Annotating data in a multi-camera scenario is yet more labour intensive as objects must be identified in multiple camera views in addition to their position labels. Some earlier methods used dynamics-based forecasting approaches that do not rely on data-driven machine learning to address the lack of data. However, breakthroughs in data-driven methods in other computer vision problems prompted a greater focus on data-driven methods for trajectory forecasting. Data-driven methods are now the basis for many current state-of-the-art methods, despite the small size of datasets compared to, for example, video classification [24]. There is a crucial link between the quality and size of the training dataset and the performance of data-driven models [25]. The availability of large high-quality datasets has the power to push forward a research area. For example, the ImageNet dataset played a vital role in the breakthrough of Convolutional Neural Networks (CNNs) [26] for image classification.

Visual feature extraction. The work of Schmidt et al. [21] demonstrates the usefulness of visual features, in addition to the past trajectory, when humans forecast pedestrian trajectories. However, best practices for extracting these features is an outstanding problem. Common visual feature extraction issues include occlusion, motion blur, and scale variation. These issues all contribute to the challenge of visual feature extraction for trajectory forecasting.

Uncertainty in future predictions. Uncertainty is an intrinsic feature of the trajectory forecasting problem. Humans, in particular, may change direction suddenly with little prior warning. Furthermore, an object may have multiple plausible future trajectories, such as upon entering a junction. As a result, uncertainty in future location grows as we predict further into the future. Modelling this uncertainty effectively is a challenge for forecasting systems, particularly as most datasets provide a single ground-truth future for each trajectory.

Pedestrian interactions. The motion of a pedestrian is influenced by other pedestrians nearby. This influence is often referred to as social force [27], whereby nearby pedestrians exert a certain amount of social force on other pedestrians. This influence ranges from simple rules of collision avoidance to more nuanced social dynamics. Furthermore, these social rules may change in different circumstances, such as large crowds [28]. Modelling pedestrian interactions has been a focus of trajectory forecasting research but remains challenging.

1.4 Thesis scope

In this thesis, we present strategies to address the first three challenges discussed in the previous section.

Lack of large labelled datasets. We propose two new methods for annotating large datasets amenable to trajectory forecasting. As human annotation is limited by human labour, we propose methods for machine-aided and fully machine-annotated datasets. The methods do not perform as well as fully manual data annotation, and are therefore not a replacement human annotation. However, they facilitate the collection of considerably larger datasets, which supplement smaller, human-annotated datasets.

Visual feature extraction. We propose a method based on optical flow to extract visual features for trajectory forecasting. We do not use human skeletal features, such as those extracted using a human pose estimation algorithm like Openpose [29]. Although considerable progress has been made in the human pose estimation task, state-of-the-art models are still not completely robust to challenges such as occlusion, lighting variations, and low resolutions [30]. These challenges are commonplace in trajectory forecasting datasets, which introduce noise in the pose estimation results and therefore also introduce noise to the trajectory forecasting process. We find that optical flow is more robust to these challenges compared to state-of-the-art human pose estimation algorithms. Throughout, we compare our methods to the work of Yagi et al. [31], a trajectory forecasting method that uses human pose estimation features. We find that our method outperforms the methods based on skeletal features. Following previous trajectory forecasting work [31–33], we predict up to 2 seconds into the future. We do not consider other methods for visual feature extraction, such as head pose estimation [34] as these features also suffer from noise introduced by prediction errors.

Uncertainty in future predictions. We propose a method for modelling uncertain future predictions in human trajectory forecasting by predicting the likelihood of a target appearing in a given location, rather than predicting a single discrete path. We use a multi-camera dataset for this task, and predict up to 12 seconds into the future. This is longer than the more conventional 1-2 seconds found in single-camera trajectory forecasting works [31–33]. Our uncertainty predictions enables us to forecasting further into the future, as growing uncertainty is modelled.

1.5 Contributions

The contributions of this thesis are as follows:

1. **Methods for collecting trajectory forecasting datasets with less human labelling (Chapter 3)**

    1.1. *Method for generating machine-annotated training data for trajectory forecasting.* We propose a machine-annotation scheme for gathering training data for trajectory forecasting models in the absence of human annotators. Our method enables the pre-training of trajectory forecasting models on large machine-annotated datasets that generalizes to human-annotated datasets.

    1.2. *Machine-aided annotation method for multi-camera trajectory forecasting.* We propose a machine-aided annotation scheme for gathering training data for trajectory forecasting models in a multi-camera setting. Our method significantly reduces the manual overhead of labelling multi-camera forecasting datasets, allowing us to create a dataset considerably larger than existing alternatives.

    1.3. *Trajectory forecasting datasets.* We collect three trajectory forecasting datasets: BDD-10k, Citywalks, and WNMF. The datasets are collected using our proposed machine-annotation and machine-aided annotation methods. The datasets are either markedly larger or more diverse than existing datasets, and each is made publicly available to the research community.

2. **Models for egocentric trajectory forecasting (Chapter 4)**

    2.1. *Dynamic Trajectory Predictor.* We propose Dynamic Trajectory Predictor, a trajectory prediction model trained using vehicle-mounted cameras. Dynamic Trajectory Predictor is trained using both machine-annotated and human-annotated data and extracts motion features from optical flow.

    2.2. *Spatio-Temporal Encoder-Decoder.* We propose Spatio-Temporal Encoder-Decoder, a trajectory prediction model trained using egocentric cameras. Unlike most previous work, Spatio-Temporal Encoder-Decoder predicts future object scale in addition to trajectory.

3. **Models for Multi-Camera Trajectory Forecasting (Chapter 4)**

3.1. *Multi-Camera Trajectory Forecasting* We introduce Multi-Camera Trajectory Forecasting, a new formulation of the trajectory forecasting task where multiple camera views are considered simultaneously. We introduce the task at three levels of granularity: In *which* camera will the object appear, *when* will the object appear, and *where* within the identified camera views will the object appear.

3.2. *Trajectory Tensors.* We propose Trajectory Tensors, a novel data representation for multi-camera trajectories. Trajectory Tensors overcome many of the shortcomings of the coordinate trajectory data representation in multi-camera settings. Trajectory Tensors enable us to elegantly model an object's location with respect to multiple camera views simultaneously.

3.3. *Multi-Camera Trajectory Forecasting models.* We propose 3 model architectures for Multi-Camera Trajectory Forecasting that use the Trajectory Tensor data representation. Each model uses data in 1 or more camera views to predict In *which* camera, *when*, and *where* the target will appear in other camera views.

1.6 Thesis Outline

This thesis is organised as follows:

- **Chapter 2: Literature review**
  This chapter reviews the history and current state-of-the-art in four key areas:
  - Vision-based forecasting
  - Trajectory forecasting
  - Pedestrian detection, tracking, and re-identification
  - Automating data annotation for machine learning tasks

  We also provide an extensive review of the current trajectory forecasting datasets used by the research community.

- **Chapter 3: Optimizing data annotation for trajectory forecasting**
  In this chapter, we propose methods for collecting trajectory forecasting datasets at scale through semi-automated data annotation. We propose methods for both single-camera and multi-camera data annotation and introduce three datasets collected using our proposed annotation methods. Our datasets are later used in Chapter 4 and Chapter 5.

- **Chapter 4: Single-camera trajectory forecasting**
  In this chapter, we introduce two methods for single-camera trajectory forecasting from an egocentric viewpoint. Our models use the past trajectory and visual features to predict an object's future location from an egocentric perspective. We train and evaluate these models using the datasets collected in Chapter 3 as well as an existing egocentric forecasting dataset.

- **Chapter 4: Multi-camera trajectory forecasting**
  In this chapter, we propose a novel formulation of the trajectory forecasting task: Multi-Camera Trajectory Forecasting. We introduce three new problem formulations and present results using several baselines and adapted state-of-the-art Single-Camera Trajectory Forecasting (SCTF) models. We also introduce a novel data representation for multi-camera trajectories and present experimental results on our new dataset introduced in Chapter 3. Our data representation is accompanied by a prediction model which predicts a multi-camera trajectory.

- **Chapter 6: Conclusion and future work**
  In the final chapter, we summarise the contributions of this thesis. We then discuss applications of our work and new research directions made possible by the presented contributions.

Chapter 2

Literature review

In this chapter, we survey works related to the contributions of this thesis. The chapter is organised as follows: Section 2.1 presents a taxonomy of tasks and methods for vision-based forecasting and summarises key works in the area. Section 2.2 gives an in-depth overview of the trajectory forecasting task. We review the history and current state-of-the-art solutions for the task and the prevalent datasets used in the area. Section 2.3 summaries three related areas relevant to our work: Pedestrian detection, tracking, and Re-Identification (Re-ID). Finally, Section 2.4 surveys developments in methods that reduce the need for manual data annotation in machine learning.

2.1 Vision-Based Forecasting

Traditionally, computer vision methods have focused on extracting information from images and videos in the present moment. Perception tasks such as image classification, activity recognition, pose estimation, semantic segmentation, and others are concerned with the present or past, i.e., gaining insight that will tell us what is happening or what has happened. Vision-based forecasting, in contrast, seeks to predict what will happen next. Tasks in this nascent area have seen comparatively less attention than their perception task counterparts, which are often prerequisites for practical vision-based forecasting applications. However, considerable breakthroughs in perception have prompted researchers to ask questions about how to use this data to build systems with more comprehensive scene understanding.

Computer vision tasks can be broken down into different levels of abstraction. For example, the task of semantic segmentation operates at a low level of abstraction by labelling each pixel's class in an image. Image classification, in contrast, operates at a higher level of abstraction where the task is to identify the category to which an image belongs. In this section, we organise vision-based forecasting according to 5 different categories, from the lowest level of

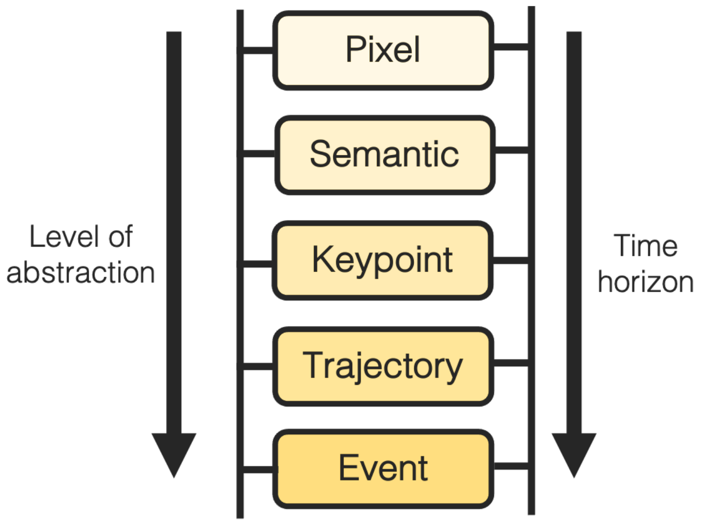


Figure 2.1: **Vision-based forecasting taxonomy.** We group vision-based forecasting tasks into 5 categories. Less abstract tasks, such as pixel or semantic forecasting, are generally feasible only for short time horizons, whereas more abstract tasks enable longer-term forecasting.

abstraction (pixel) to the highest (event) as shown in Figure 2.1. Examples of each of these sub-problems is shown in Figure 2.2. In this section, we survey key work in each sub-field of this incipient research area.

2.1.1 Pixel Level

At the lowest level of abstraction, pixel-level forecasting involves directly predicting future pixel values in video. Typically, a model is given a sequence of video frames to predict future video frames. The predicted frames are then in turn used for other downstream tasks such as predicting future actions [35], detecting anomalies [36], or predicting future optical flow [37]. As video data which does not need to be labelled for pixel forecasting is easy to obtain, pixel forecasting models are also trained on large datasets as a method of self-supervised [38] representation learning. The goal is to learn a generalisable feature representation that will boost performance on tasks where limited training data is available.

Due to the high level of uncertainty in predicting individual pixel values, methods typically predict short time horizons. However, Wichers et al. [39] propose a model to predict up to 20 seconds into the future by jointly training an encoder to extract frame feature embeddings, an embedding predictor to forecast future frame embeddings, and a decoder to generate future video frames given the predicted embedding and the last observed video frame. As a dataset with a static background is used, the model also learns to distinguish between foreground and background pixels as a by-product.

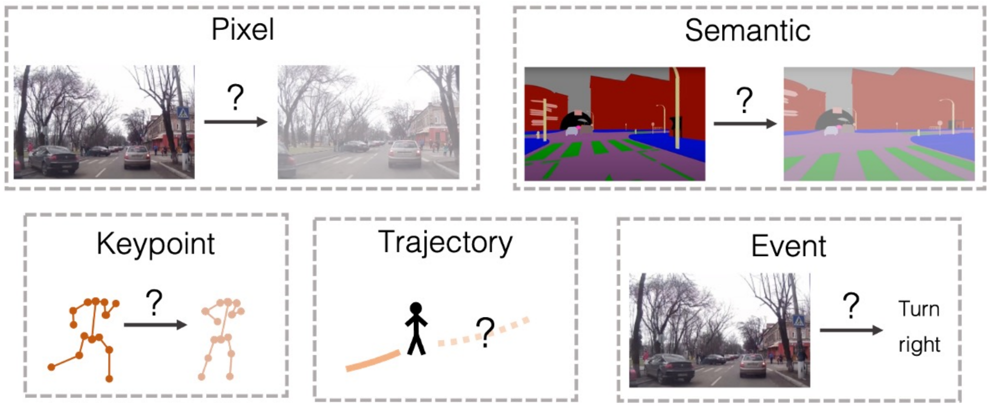


Figure 2.2: **Vision-based forecasting.** Examples of each of the 5 sub-tasks in our vision-based forecasting taxonomy.

Pixel-level forecasting has also been used for video anomaly detection. Here, the goal is to identify sections of video that deviate from a set of “normal” video sequences. Liu et al. [36] use a Convolutional Neural Network (CNN) to predict future video frames and use the difference between the predicted ground-truth frame to compute a frame-wise anomaly score. It is expected that frames with a large prediction error are anomalous. Yao et al. [40] adopt a similar approach but compute anomaly score maps for each frame, enabling the precise location of the suspected anomaly to be identified.

2.1.2 Semantic Level

Semantic segmentation is a fine-grained perception task where an object class (e.g. vehicle, building, person) label is assigned to each pixel in an image. This perception task has seen much attention [41], and was first extended to the forecasting domain in 2016 by Luc et al. [42]. The authors introduce segmentation forecasting as the task of predicting future frame segmentation in a video sequence given current and past video frames. The authors predict semantic segmentations of future video frames on the Cityscapes [43] autonomous driving dataset, showing that predicting segmentation masks yields better performance than directly predicting Red, Green, Blue (RGB) frames (i.e. using one of the methods introduced in the previous subsection) and then segmenting these. However, due to the fine-grained nature of semantic segmentation, prediction performance falls considerably for predictions over 0.5 seconds. This result highlights the importance of selecting the correct level of abstraction for the task's difficulty, which is closely related to the prediction time horizon. Similarly, Terwilliger et al. [44] use optical flow features for semantic forecasting and can produce more accurate forecasts up to 0.5 seconds into the future. Luc et al. [45] extend their earlier semantic forecasting work to instance segmentation forecasting, in which each object instance is individually predicted. This more challenging problem involves assigning both pixel and object identifiers to each pixel in future video frames. For example, rather than labelling all pixels belonging to the “car” object class with the “car” label, each unique object is given a unique label, e.g., “car-1”, “car-2”, ... etc.

The fine-grained nature of segmentation forecasting makes long term predictions of over 0.5 seconds very challenging to learn. The pedestrian object class, in particular, is extremely challenging for long term prediction due to highly dynamic motion and deformable parts. In some applications, such as object tracking [46] and collision avoidance [31], such fine-grained prediction is not essential. For these applications, forecasting at the keypoint, trajectory, or event level rather than full segmentation may be sufficient and are better suited to longer-term predictions.

2.1.3 Keypoint Level

Keypoint forecasting models predict the locations of a set of keypoints for an object, commonly human joint keypoints. Typically, human joints are represented by several keypoints representing the location of the feet, knees, hips, shoulders, elbows, hands, neck, and face [30]. In some datasets, more detailed keypoints are given that may include facial features or individual fingers and toes [47]. Methods for human pose estimation generally either detect the person first, then detect the keypoints (top-down) or detect keypoints first and then group them into a person detection (bottom-up) [30, 47]. Predicting the future locations of humans based on their joints has several applications, such as self-driving vehicles [48] and for anticipating human actions [49]. Models may be either action-specific or action-agnostic. Action-specific models require the label of the action being performed, whereas action-agnostic models do not. Action-specific models are either trained separately for different actions or jointly and then use the provided action label as an additional input to the model [50]. Action-agnostic models are generally more useful for real-world applications as action labels are often unavailable at inference time. However, action-agnostic models are more challenging to generalise than their action-specific counterparts [50].

The previously mentioned keypoint forecasting algorithms generally forecast local movements only, i.e., the pose is predicted but not the future trajectory. This limitation is addressed by Mangalam et al. [48] who propose an RNN-based approach to predict pose and trajectory jointly. The authors propose a two-stream prediction model, where global motion (i.e., the individual’s trajectory) and local motion (i.e., the relative location of keypoints in relation to each other) are predicted separately and then merged. This approach avoids issues resulting from the difference in keypoint displacement magnitude resulting from global movements (large displacements) and local movements (small displacements).

2.1.4 Trajectory Level

Trajectory forecasting is the task of predicting the future trajectory of an object in a sequence of video frames. The centre point of an object is represented x and y values in a coordinate space, and models predict the future location given a sequence of past locations. As the central theme of this thesis, we cover trajectory forecasting in detail in Section 2.2.

2.1.5 Event Level

At the highest level of abstraction, event-based forecasting involves predicting the occurrence of a future event. The range of possible applications of event-based forecasting is huge, from predicting traffic accidents [51] to patient outcomes in medical diagnosis [9]. The time horizon of event-based forecasting is hugely varied, on the order of seconds [52], years [53] or even decades [9].

Of the works in event-based forecasting, human activity prediction is one of the tasks to see the most attention. Human activity prediction is the problem of anticipating human actions such as waving or jumping in video with limited information. There are two definitions commonly used in the activity prediction literature: (1) Classify an action as early in the sequence as possible *before* the action has begun. (2) Classify an action as early in the sequence as possible *after* the action has begun. Figure 2.3 depicts these two definitions graphically. Note the similarity but subtle distinction between these two definitions. The two are often used interchangeably in activity prediction literature. One reason for this is that the precise starting point of an action is ambiguous, so the difference between classifying an action before it begins and shortly after the beginning is minimal.

Using definition 1 of human activity prediction, Vondrick et al. [54] predict future visual representations (features) and then use the predicted representations to anticipate future human actions. Learning to predict representations can be done in a self-supervised (discussed further in Section 2.4.2) manner using video data, assuming a model for generating a visual representation is available. The authors use AlexNet [26] to extract a visual representation for each frame and then predict representations of future frames using a CNN. The predicted future representation is then assigned an action label using an action classification model.

Jain et al. [52] anticipate 5 actions of a human driver: straight, left turn, right turn, left lane change, and right lane change using data from Global

Positioning System (GPS), vehicle dynamics, and a driver-facing camera. Features for each modality are computed separately using a Long Short-Term Memory (LSTM) [55] network and then fused using a fully connected layer. The future action is labelled at each timestep. The model is trained with a novel exponentially increasing loss function that helps to prevent overfitting. The exponential loss assigns a lower weight to loss values for frames early on in a video clip and higher weights to later video frames. Intuitively, actions are easier to predict closer to their occurrence, so mistakes made in later video frames are penalised more heavily. Therefore, during training, frames early in the video clips do not impact the model weight updates as much as the later frames.

Suzuki et al. [51] extend the exponential loss introduced by Jain et al. by modifying the loss as model training progresses. At the beginning of the training, the loss is equivalent to the exponential loss. As training progresses, the loss is modified such that it gives higher weights to earlier frames. This approach is inspired by the curriculum learning paradigm for machine learning, which demonstrates that the order in which examples are presented to a machine learning algorithm can significantly impact the final performance of the model. In their seminal paper, Bengio et al. [56] propose the idea of guiding the training process inspired by how humans learn. Easier examples are presented before more challenging examples to keep the learning process in an optimum state in which training examples are neither too easy nor too difficult. Similarly, by increasing the influence of earlier frames as training progresses, easier examples can be learnt first (close to the event) before moving to more challenging examples. This method is used to forecast traffic accidents with a new traffic incident dataset, introduced with the results. The choice of how much to adapt the loss function at each training epoch is essential, and finding an optimal value is non-trivial. To consistently find reasonable values for the loss, the loss reaches its peak at the average time at which actions were

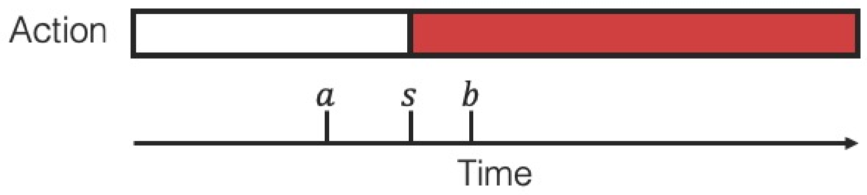


Figure 2.3: **Action prediction definitions.** Consider an action that begins at time s. Action prediction is sometimes referred to as the problem of anticipating the action before it occurs, such as at time a (Definition 1) or as early as possible during the action, such as at time b (Definition 2). Although the time t is labelled by human annotators, the precise start time of an action can be difficult to define precisely.

anticipated of the previous epoch on the validation set. This scheme results in an adaptive loss function that is modified based on current model performance.

Aliakbarian et al. [57], propose a model to detect actions as soon as possible in a clip, rather than to forecast future actions that have not yet occurred. Therefore, standard action recognition datasets [58–60] are used for evaluation. They introduce a new loss consisting of two terms. The first penalises false negatives uniformly through the clip. The second penalises false positives with a linearly increasing magnitude over time. Similarly to the exponential loss, the rationale behind this decision is that the ground-truth action label may be ambiguous early in a video clip. However, the exponential loss results in early frames getting small loss values, and thus predictions are not made early in the clip. The introduced loss encodes this idea while encouraging the mode to make predictions early by penalising false negatives consistently throughout the clip. The loss introduced performs better than the exponential loss on the benchmark datasets for earlier frames, with similar performance for later frames.

In this thesis, we introduce models that forecast at the trajectory level of abstraction in Chapter 4 and both trajectory and event level in Chapter 5.

2.2 Trajectory Forecasting

Trajectory forecasting is the task of predicting a future sequence of locations for an object given its current and past location information.

Problem formulation. Formally, we define an object location at a particular timestep t as L_t, comprised of a coordinate $(x, y)_t$ in the image space. The coordinates do not include the vertical (z) axis and are generally assumed to occupy the ground plane. The goal of a trajectory forecasting model is to predict a sequence of locations $L_{t+1:m}$ for a given object, where m is the number of future timesteps to predict. Generally, models predict a single sequence of locations $L_{t+1:m}$ for each object. However, an increasing number of works consider the multi-modal trajectory problem, where methods predict multiple possible future trajectories to model the uncertainty in future trajectory.

Most existing trajectory forecasting works predict future trajectories in the image coordinate space. This is a convenient way to collect datasets, as human annotators need only label locations in the dataset video. However, image space predictions have several disadvantages. Image-space predictions are susceptible to distortion from the camera lens and viewpoint. This is mitigated by using cameras with rectilinear lenses and shooting footage from a directly overhead view, such as from a drone. For datasets with a moving camera with both egomotion and object motion, the two motion sources are not considered separately, and must therefore be modelled jointly. Ego motion is often modelled using a distinct camera motion estimation algorithm [31] or using optical flow [61]. A few datasets, such as the crowd dataset [62], are suitable for trajectory forecasting in the world coordinate space. However, such datasets are uncommon as collection requires accurate camera calibration. The Crowd dataset uses depth sensors to project locations from the image space to the world space. In this thesis, we predict trajectories in the image coordinate space.

The features used for trajectory forecasting include the past and current object locations $L_{t-n:t}$ but are not limited to locations alone. For example, methods commonly also use the locations of other nearby objects, visual information from the scene, and structural constraints to make better-informed predictions.

Evaluation metrics. The two most common metrics for evaluating trajectory forecasting models are the Average Displacement Error (ADE) and Final Displacement Error (FDE), which are defined for a predicted trajectory $\hat{L}_{t+1:m}$ as follows:

$$ADE = \frac{\sum_t^m ||\hat{L}_t - L_t||_2}{m}, \tag{2.1}$$

$$FDE = ||\hat{L}_m - L_m||_2, \tag{2.2}$$

where $||a, b||_2$ is the Euclidean (ℓ_2) distance between two locations a and b. As uncertainty in future location grows with time, FDE is expected to be greater than ADE. In addition to ADE and FDE, the Mean Squared Error (MSE) is also sometimes used, computed by taking the square of the differences between the predicted and ground truth trajectories for all timesteps, then taking the mean:

$$MSE = \frac{\sum_t^m (\hat{L}_t - L_t)^2}{m}, \tag{2.3}$$

The MSE more heavily penalizes large errors than the ADE.

Multi-modal trajectory forecasting models are typically evaluated by the best-of-k ADE and best-of-k FDE metrics. These metrics alongside standard ADE and FDE metrics are shown in Figure 2.4. Best-of-k ADE and best-of-k FDE are formally defined as follows:

$$best\text{-}of\text{-}k\ ADE = \min(\frac{\sum_t^m ||\hat{L}_t^i, L_t||_2}{m}) \mid i = 1, 2, ..., k, \tag{2.4}$$

$$best\text{-}of\text{-}k\ FDE = \min(||\hat{L}_m^i, L_m||_2) \mid i = 1, 2, ..., k, \tag{2.5}$$

Where k is the number of generated future trajectories. These metrics

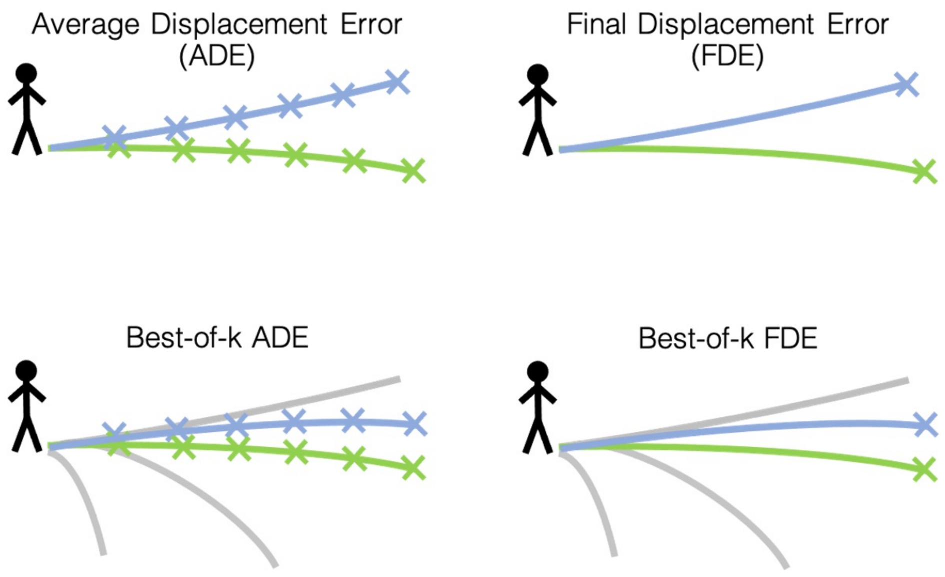


Figure 2.4: **Trajectory forecasting metrics.** Four widely adopted trajectory forecasting metrics. Ground truth trajectories are shown in green, predicted trajectories in blue, and discarded predictions in grey. In the best-of-k setting, a model predicts several trajectories and only the best prediction is selected.

overcome the problem of penalising plausible but incorrect trajectories by taking only the trajectory closest to the ground truth out of k predictions and have become standard multi-modal metrics.

Application domains. The ability to accurately predict the future location of a moving object spans many application domains. For example, trajectory forecasting models are used in Advanced Driver Assistance Systems (ADASs), which are now commonplace in modern vehicles [65]. An ADAS can prevent collisions with other road users by combining methods for detecting the current location of other road users and then predicting their future location. If a high collision likelihood is detected, the system then alerts the human driver or even applies automatic braking and steering [65]. Such systems also pave the way towards fully autonomous vehicles that do not require a human driver. Human trajectory forecasting has also been used to predict potential hazards at construction sites for enhanced worker safety [66], robotic navigation [67], to predict the movements of players in sports [68], and to assist in object tracking, particularly through occlusion [69, 70]. Although humans are the most common class of objects to predict, trajectory forecasting methods are also applied to other objects such as ships [71], animals [72], and road vehicles [73].

Camera viewpoints. We categorise trajectory forecasting methods into the broad categories of Bird's Eye-View (BEV) approaches, egocentric approaches, and multi-camera surveillance approaches. Trajectory forecasting from a

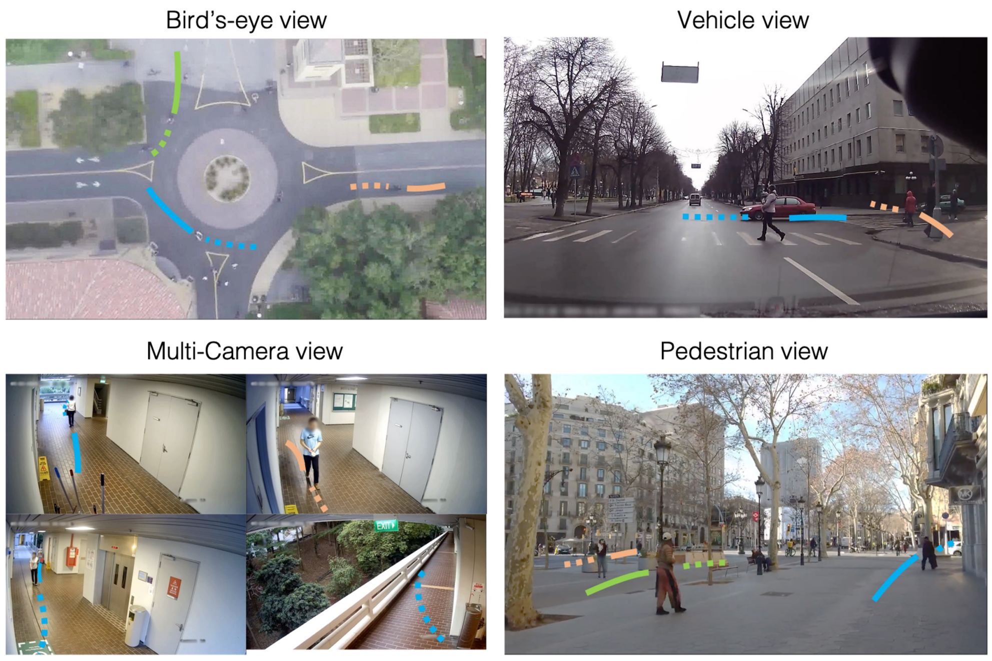


Figure 2.5: **Example human trajectory forecasting viewpoints.** Given a past trajectory (solid line), the goal of trajectory forecasting is to predict a future trajectory (dashed line). Shown are the Stanford Drone dataset [63] (upper left), the Joint Attention in Autonomous Driving (JAAD) dataset [64] (upper right), the Warwick-NTU Multi-Camera Forecasting dataset (lower left) and the Citywalks dataset (lower right). The Warwick-NTU Multi-Camera Forecasting and Citywalks datasets will be introduced in Chapter 3.

BEV uses a fixed overhead camera. From an egocentric viewpoint, trajectory forecasting uses an object-level viewpoint. Multi-camera surveillance methods use multiple cameras mounted in a typical Closed-Circuit Television (CCTV) setting. Examples of these viewpoints from four trajectory forecasting datasets are shown in Figure 2.5, where we further divide egocentric-view forecasting into vehicle and pedestrian view, both of which are studied in the literature. Each viewpoint has its own unique set of advantages and disadvantages.

In this section, we review the history and current state-of-the-art in BEV, egocentric, and multi-camera trajectory forecasting. We then review the most widely-used trajectory forecasting datasets currently available.

2.2.1 Bird's Eye-View Forecasting

BEV trajectory forecasting methods have seen considerable attention from the research community in recent years. Using a BEV is practical for modelling crowd motion patterns and interactions with the environment. Furthermore, a BEV does not suffer from challenges relating to scale and perspective, as the area captured by the camera is assumed to be a flat plane. This camera view allows researchers to focus on feature extraction and trajectory modelling in a straightforward setting.

Early Works. Early works on trajectory forecasting started with hand-crafted models based on Newton's Laws of Motion. For example, Constant Velocity (CV) and Constant Acceleration (CA) models are commonly used to extrapolate past object locations into the future. These motion models are often used with a Bayesian filter, such as a Kalman filter [74], which recursively updates location priors to filter measurement noise. For example, Elnagar [75] predicts the trajectory of moving objects using a CA assumption with a Kalman filter. Pellegrini et al. [76] use a CV motion assumption but also consider the interactions between multiple objects. The path of each object is updated if a collision is likely to occur. This idea is based on the social force model [27], which describes pedestrian motion in crowds in terms of their internal motivations and interactions with other pedestrians.

Data-Driven Approaches. Data-driven trajectory forecasting approaches have dominated the literature since larger datasets, and better models have been developed. Today, data-driven approaches consistently outperform hand-crafted approaches due to the ease of extracting prior knowledge from datasets. For example, humans prefer to walk on pavements rather than the road [77]. This knowledge can be easy for a data-driven model to learn directly from a dataset of labelled trajectories; however, encoding this information by hand would involve generating a detailed environment map.

Alahi et al. introduced the pioneering Social-LSTM [32] trajectory forecasting model in 2016. Social-LSTM extracts features from trajectories using a recurrent model. The authors introduce social pooling, which pools together the hidden states of nearby pedestrians. Social pooling enables the model to learn social dynamics, such as collision avoidance, directly from data rather than manually engineering this constraint. The learned approach outperforms a CV model with Kalman filtering, and a social force-based model [78], paving the way for more data-driven approaches.

Many data-driven works have followed Social-LSTM, introducing new ideas to address different challenges. For example, the social pooling layer pools hidden states of nearby pedestrians but does not model the interactions of the entire scene. Choi et al. [79] instead model interactions globally for all pedestrians as well as environmental constraints by dividing the entire scene into discrete grid cells and extracting spatio-temporal features in each grid cell. Environmental constraints are also considered the Deep Context Map model proposed by Gilitschenski et al. [80] where a dedicated context encoder extracts features relevant to the specific scene. Sadeghian et al. [81] propose CAR-Net, which uses a combination of CNN and LSTM layers with a visual attention module. Unlike many existing works, CAR-Net simultaneously predicts both vehicle and pedestrian trajectories. CAR-Net is trained using the ℓ_2 distance between ground truth and predicted trajectories. This loss function may overly penalise sensible but incorrect predictions and will be discussed later in this subsection.

Most of the works discussed so far rely on variations of Recurrent Neural Network (RNN) models for feature extraction, which are well-suited to time series prediction problems. However, alternative deep learning layers such as CNNs, originally created for image data, are also sometimes applied to time series problems [82]. Yi et al. [83] propose Behaviour-CNN, a trajectory forecasting model based on convolutional rather than recurrent layers. Given a 3-dimensional volume encoding past object locations, Behaviour-CNN predicts the future trajectory of an object using convolutional layers. Experimental results show that Behaviour-CNN accurately predicts target exit locations in a scene with multiple exits, and the location predictions also reduce the number of fragmented tracks when used as a prior for an object tracking algorithm.

Minoura et al. [17] reformulate the trajectory forecasting problem for large crowds, a task they call crowd density forecasting. Rather than predicting a trajectory for each object in the scene, crowd density forecasting involves anticipating how a crowded scene will unfold by predicting the relative person density in each part of the scene. The authors propose a model that outperforms existing trajectory forecasting methods designed to predict individual trajectories when applied to a crowd dataset.

Multi-Modal Approaches. The distribution of future trajectories is often termed multi-modal, meaning the that distribution of future trajectories has multiple modes. For example, as a pedestrian approaches an intersection, they may have the option of turning left or right. Multiple modes may seem equally likely; however, single-future models such as those discussed above predict a single trajectory. Multiple possible futures can be problematic when using a loss function based on a single future such as ℓ_2 distance, which heavily penalises reasonable but incorrect forecasts, e.g., turning right rather than left at an intersection. Rather than predicting a single trajectory, a growing body of works generate multiple possible trajectories, commonly referred to as multi-modal trajectory forecasting.

Rather than using the ℓ_2 loss, several works use an adversarial loss to train models capable of generating multiple possible future paths using a Generative Adversarial Network (GAN) [33, 84] or a Variational Auto-Encoder (VAE) [73, 85]. In this setting, one network predicts future trajectories (the generator), and a second network differentiates between real and predicted trajectories (the discriminator). The better job the discriminator does at differentiating trajectories, the higher the value of the adversarial loss. This training schedule encourages models to generate multiple plausible trajectories rather than minimising the ℓ_2 distance between predicted and ground-truth trajectories.

Many approaches predict one timestep at a time using an auto-regressive prediction approach. This strategy leads to the accumulation of errors as trajectories are predicted further into the future. Mangalam et al. [85] attempt to address this issue by first predicting several potential trajectory endpoints and then generating a possible path to reach these endpoints. This approach is motivated by the idea that humans have target locations (goals) that they wish to reach and then plan a path to reach that location. The approach improves prediction performance in a multi-modal setting, particularly for later timesteps.

Despite widespread usage, the best-of-k metrics used for multi-modal trajectory forecasting evaluation introduced in equations 2.4 and 2.5 have come under some criticism. Schöller et al. [86] show that a simple constant velocity model outperforms several state-of-the-art multi-modal trajectory forecasting models when evaluated under best-of-k metrics. Furthermore, predicting several implausible trajectories will not be penalised, providing at least one of the k generated trajectories is close to the ground truth. Evaluation is challenging as typically only a single ground truth trajectory exists despite multiple plausible futures, and the best-of-k metrics remain the standard. However, Liang et al. avoid this issue by generating a simulated dataset where each trajectory has multiple futures [87]. The proposed dataset overcomes the problems associated with a single future per trajectory at the cost of using simulated rather than real-world data.

2.2.2 Egocentric-View Forecasting

Compared to BEV forecasting, egocentric trajectory forecasting has seen comparatively less attention by the research community. This is partly due to the lack of standard benchmark datasets and additional challenges relating to modelling egomotion and changes in perspective. Similarly to BEV forecasting, earlier methods used hand-crafted approaches, whereas most more recent works take a data-driven approach.

Hand-Crafted Approaches. Attention to egocentric-view forecasting began with vehicle-mounted cameras. Given the absence of large pedestrian trajectory datasets, previous works have modelled the dynamic motion of pedestrians using Linear Dynamic System (LDS) that combine the assumptions of CV or CA with a filtering algorithm such as the Kalman filter [88]. To model non-linear, dynamic motion, a Switching Linear Dynamic System (SLDS) uses a discrete Markov chain to select between multiple LDS at each timestep based on past observations. However, the SLDS is limited to *reacting* to pedestrian motion rather than *anticipating* a change in dynamics. To address this issue, existing works [89, 90] focus on additional cues such as pedestrian head pose, motion state, and road scene context or use a non-linear filtering algorithm such as the unscented Kalman filter [91].

Data-Driven Approaches. Similarly to the trends in BEV forecasting, an increasing number of more recent works take a data-driven approach to egocentric forecasting. Yagi et al. [31] introduce one of the first data-driven approaches for egocentric forecasting. The authors propose a model combining features from the pedestrian's pose, estimated ego-motion, and past location information. Their model outperforms the CV baseline and the Social-LSTM [32] model adapted for egocentric forecasting. However, the model relies on pose estimation features. Obtaining accurate pedestrian pose estimation is sometimes impractical and not always guaranteed to be accurate, especially in low-resolution or low-lighting scenarios. Bhattacharyya et al. [92], use an LSTM to predict the future location of pedestrian bounding boxes from a vehicle-mounted camera by first estimating future vehicle ego-motion and then using these estimates with observed bounding boxes to forecast the location of future bounding boxes. Experimental results show that predicting vehicle ego-motion improves the accuracy of pedestrian bounding box predictions. Yao et al. [61] use a Gated Recurrent Unit (GRU) encoder-decoder model to predict future bounding boxes of other vehicles using a vehicle-mounted camera. Their proposed approach uses past vehicle bounding boxes, optical flow, and future ego-motion as inputs to the model.

Yao et al. [93] use egocentric trajectory forecasting to detect traffic anomalies from a vehicle-mounted camera. The authors compared predicted object bounding boxes with observed bounding boxes, hypothesising that a large deviation between these may indicate a traffic anomaly, such as a traffic accident. Park et al. [94] consider trajectory forecasting from an egocentric viewpoint but instead predict the trajectory of the individual wearing the camera rather than other objects in the scene. We consider this task distinct from egocentric trajectory forecasting, where the trajectories of other targets in the scene are predicted.

The lack of available training data limits all data-driven approaches. We review datasets currently available for both BEV and egocentric trajectory forecasting in Section 2.2.4.

2.2.3 Multi-Camera Trajectory Forecasting

The trajectory forecasting methods discussed above all consider targets as viewed from a single-camera view. However, there is a growing number of computer vision systems applied in multi-camera settings, such as intelligent

CCTV systems that actively monitor an area of interest using several cameras [95]. Alahi et al. [62] predict the origin and destination of pedestrians using a dense multi-camera setup. The authors use cameras with BEV and depth sensors to map tracks to the world coordinate space. This approach makes it easy to apply existing trajectory forecasting methods using the trajectories in the world coordinate space; however, dense camera setups with depth sensors are not standard in real-world multi-camera systems. To the best of our knowledge, no other works explore trajectory forecasting in a multi-camera setting.

2.2.4 Trajectory Forecasting Datasets

Many datasets have been created for trajectory forecasting, each focusing on different aspects of the problem. This thesis presents three trajectory forecasting datasets created to address gaps in previous publicly available trajectory forecasting datasets. To contextualise our new datasets, we review the most widely used existing trajectory forecasting datasets below. A full list of datasets is shown in Table 2.1. Our newly-created datasets shown below the dashed line will be introduced in Chapter 3. Sample frames from selected datasets suitable for trajectory forecasting are shown in Figure 2.6.

BEV Datasets. Datasets with an overhead or Bird's Eye-View (BEV) perspective are the most common and widely used in the trajectory forecasting literature. Using an overhead view makes datasets highly suitable for modelling social and environmental factors. The footage is usually either captured from directly overhead (nadir-view) using a drone or from the perspective of a surveillance camera. For this review, we consider both nadir and surveillance footage as BEV footage. We summarise key datasets below:

- UCY [96]. One of the first benchmark datasets to see wide adoption, UCY contains two scenes filmed from a nadir viewpoint. One scene contains high pedestrian density, whereas the other is more sparse.
- ETH [76]. ETH followed UCY with two more scenes for trajectory forecasting. The two datasets are often used together in BEV forecasting research.
- Town center [97]. The town centre dataset consists of 5 minutes of footage shot from a surveillance view in a town centre. The dataset contains full object bounding boxes.
- VIRAT [98]. The VIRAT dataset was created originally for the detection and classification of human actions from a surveillance viewpoint. However, due to the large size (29 hours of footage) and trajectory labels, VIRAT is also used for trajectory forecasting.

Table 2.1: **Trajectory forecasting datasets.** Note that some of the listed datasets are originally created for other tasks, such as tracking and Re-ID, but are also suitable for trajectory forecasting. Our new datasets introduced in this thesis are tabulated below the dashed line. A scene is video which captures a distinct location.

Dataset	Year	Viewpoint	Object type	Scenes	Duration (hours)	Publicly available
UCY [96]	2007	Nadir	Pedestrian	3	0.28	✓
ETH [76]	2009	Nadir	Pedestrian	2	0.42	✓
Town center [97]	2011	Surveillance	Pedestrian	1	0.08	✗
VIRAT [98]	2011	Surveillance	Pedestrian	16	29	✓
KITTI [99]	2012	Vehicle	Pedestrian & vehicle	50	6	✓
Crowd [62]	2014	Multi-camera surveillance	Pedestrian	132	?	✓
Daimler [88]	2013	Vehicle	Pedestrian	68	0.21	✓
Grand central [100]	2015	Surveillance	Pedestrian	1	0.55	✗
Duke-MTMC [101]	2016	Multi-camera surveillance	Pedestrian	8	14	✗
MOT-16 [46]	2016	Egocentric & surveillance	Pedestrian & vehicle	14	0.13	✓
SDD [63]	2016	Nadir	Pedestrian & vehicle	8	5	✓
EgoMotion [94]	2016	Egocentric	Pedestrian	26	9.1	✗
JAAD [64]	2017	Vehicle	Pedestrian	346	1	✓
FPL [31]	2018	Egocentric	Pedestrian	88	4.5	✗
Cityflow [102]	2019	Multi-camera surveillance	Vehicle	40	3.3	✓
MOT-20 [103]	2020	Surveillance	Pedestrian	8	0.14	✓
nuScenes [104]	2020	Vehicle	Pedestrian & vehicle	1000	333	✓
Forking paths [87]	2020	Surveillance & Nadir	Pedestrian	4	3.2	✓
BDD-10K [1] (uses videos from [105])	2019	Vehicle	Pedestrian	10,000	111	✓
Citywalks [2]	2020	Egocentric	Pedestrian	358	2	✓
WNMF [3]	2020	Multi-camera surveillance	Pedestrian	15	600	✓

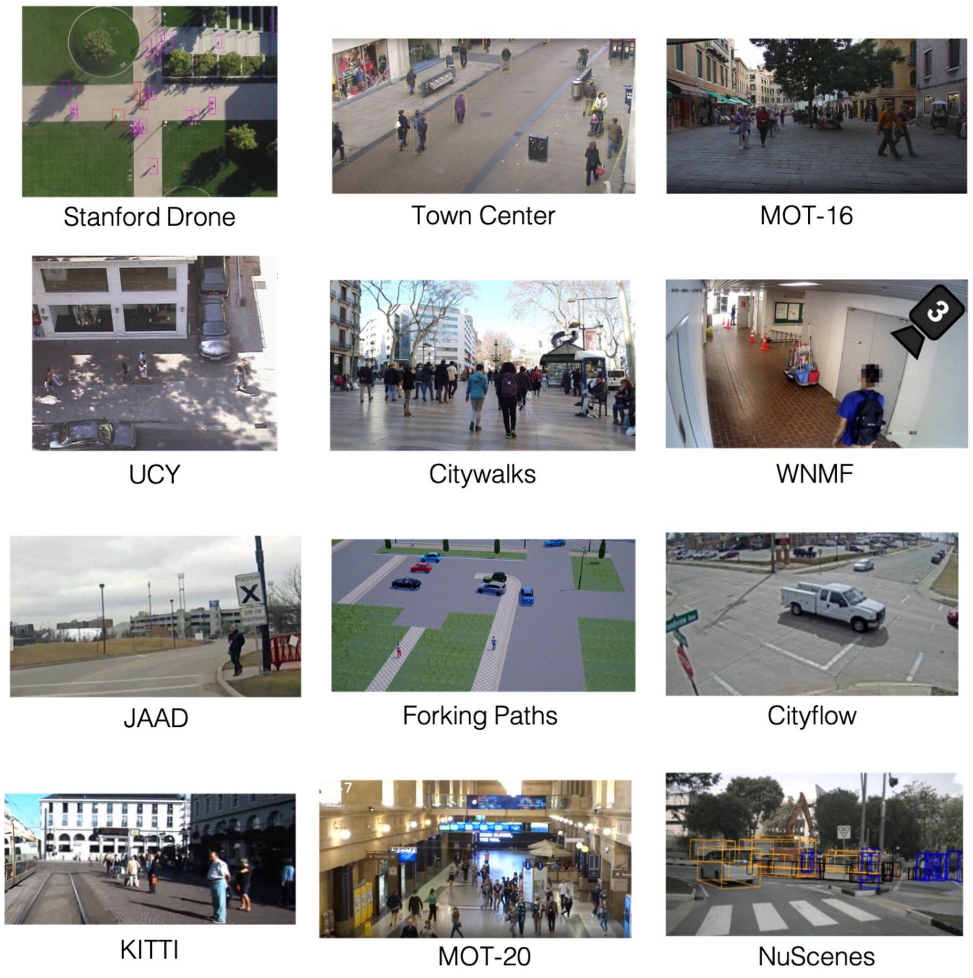


Figure 2.6: **Selection of datasets suitable for trajectory forecasting.** Datasets shown are Stanford Drone [63], Town Centre [97], MOT-16 [46], UCY [96], Citywalks [2], WNMF [3], JAAD [64], Forking Paths [87], Cityflow [102], KITTI [99], MOT-20 [103], and NuScenes [104].

- Grand Central [100]. The Grand Central dataset contains around 30 minutes of footage from a surveillance camera in Grand Central Station, New York. The footage contains high pedestrian density with fully annotated object trajectories. While initially released publically, the dataset is no longer available at the time of writing.

- Stanford Drone Dataset (SDD) [63]. The Stanford Drone Dataset, released in 2016, features footage shot from drones in 8 different scenes. The dataset is similar to ETH and UCY but on a larger scale and has been widely used as a BEV trajectory forecasting benchmark dataset since its release.

- MOT-20 [103]. MOT-20 is the latest version of the multi-object tracking challenge datasets, following MOT-16. MOT-20 features 8 sequences of 3 crowded scenes from a surveillance camera view.

- Forking paths [87]. Forking paths is a simulated dataset for multi-modal trajectory forecasting. Human annotators control agents in a synthetic world such that one past trajectory is associated with several future trajectories. The synthetic scenes are derived from scenes in the VIRAT [98] dataset.

Egocentric Datasets. Few public datasets exist for object-level view trajectory forecasting. Many existing works use either in-house datasets or object tracking datasets which are re-purposed for egocentric forecasting. We identify the following existing datasets suitable for egocentric trajectory forecasting:

- MOT-16 [46]. Originally created for Multi-Object Tracking (MOT), MOT-16 contains annotated pedestrian bounding boxes from both egocentric and BEV cameras. The dataset contains 14 video sequences, of which 11 are from an egocentric perspective.

- EgoMotion [94]. Park et al. gather the EgoMotion dataset for future localisation. This task involves predicting the future trajectory of the person wearing the camera rather than the other objects in the scene. We consider this task distinct from egocentric trajectory forecasting, in which models predict the trajectory of other objects in the scene.

- FPL [31]. The FPL dataset contains 4.5 hours of footage recorded from a chest-mounted camera in a variety of environments. Although highly suitable for egocentric trajectory forecasting, the original video footage is not available to download due to privacy restrictions.

There are also several public datasets consisting of data from vehicle-mounted cameras:

- KITTI [99]. The KITTI dataset is one of the most widely-used datasets for autonomous driving research. Similarly to MOT-16 [46], KITTI was initially created for object tracking but can also be suitable for trajectory forecasting. Bounding box annotation for both vehicles and pedestrians are provided for around 6 hours of footage.

- Daimler [88]. The Daimler dataset was one of the first widely-used publically available for trajectory forecasting from a vehicle-mounted camera. The dataset contains 55 clips depicting four classes of pedestrian movement: Crossing, stopping (at the curbside), starting (to walk laterally) and bending in (crossing at a diagonal).

- Joint Attention in Autonomous Driving (JAAD) [64]. JAAD was collected by Rasouli et al. to study how vehicle-pedestrian interactions in diverse scenarios. The dataset contains 346 short clips, generally depicting a pedestrian crossing event or other interaction with a driver. Bounding boxes and tracking annotations are also provided.

- nuScenes [104]. nuScenes is a large and comprehensive autonomous driving dataset released in 2020 with data from 6 cameras, 5 radar, and a Light Detection and Ranging (LiDAR) sensor mounted on a vehicle. The dataset contains 1000 scenes with 3D annotated object bounding boxes for both other vehicles and pedestrians.

Multi-Camera Datasets. Multi-camera datasets suitable for trajectory forecasting are limited. Most existing datasets commonly used for trajectory forecasting, such as those discussed above, consist of a single camera view. Collecting multi-camera datasets is a considerable challenge, as all cameras need to be synchronised and trajectories annotated across multiple camera views. A multi-camera setting means that the same object must be re-identified in all camera views in addition to object locations. Owing to these data collection challenges, there are few multi-camera datasets available to the research community.

We identify three existing multi-camera datasets:

- Duke-MTMC [101]. The Duke-MTMC dataset was collected outdoors on the Duke University campus. It was initially indented for multi-target multi-camera tracking of individuals across the eight cameras in the network. However, complete trajectory information is labelled, making it also suitable for MCTF. The dataset attracted considerable attention in the months following its release, as it was one the largest multi-camera multi-target dataset with synchronised cameras and multi-camera identity labels. However, the Duke-MTMC dataset was removed in 2019 following an investigation by the Financial Times newspaper citing privacy concerns [106].

- SAIVT-SoftBio [107]. The SAIVT-SoftBio dataset was created for person Re-ID. The authors gather footage featuring 152 identities using an eight-camera setup inside a building. Although the dataset contains cross-camera trajectory information making it suitable for MCTF, there is only a single multi-camera trajectory per identity, which makes the dataset relatively small compared to other trajectories forecasting datasets. Furthermore, the building furniture is not consistent across all clips, which impacts trajectories.

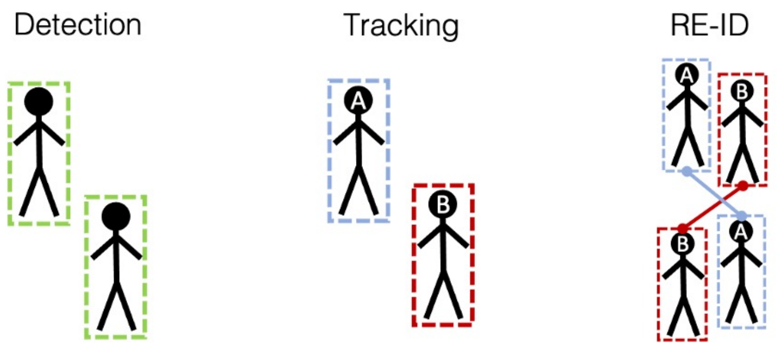


Figure 2.7: **Detection, tracking, and Re-ID.** Detection involves locating each target in a scene using object bounding boxes (left). Tracking involves assigning unique and consistent identity labels to each target detected in a video (center). Re-ID involves matching targets of the same identity detected in different camera views (right).

- Crowd dataset [62]. The Crowd dataset was collected inside a train station using 132 cameras. Cameras use a BEV and are equipped with depth sensors to obtain 3D detection information. The 3D detections facilitate the generation of tracks in the real-world coordinate space, which allows for multi-camera tracking. Such camera setups are not widespread, as accurate depth and geometric information are not commonly available.

In this thesis, we collect two new datasets. The entire process from gathering raw data, cleaning it, annotating it, and exporting it into a machine-readable format is a time consuming process. Furthermore, comparisons with existing methods are not available and must be re-implemented for the new dataset. However, the datasets listed above are all limited in terms of either size, variety, or annotation.

2.3 Pedestrian Detection, Tracking, and Re-Identification

Detecting the current location of a pedestrian and tracking these locations across time is an essential prerequisite to forecasting their trajectory. In this section, we review current state-of-the-art methods for detecting and tracking pedestrians. We also review current methods for person Re-ID, which is a crucial component in many pedestrian tracking systems, as appearance features are used to match pedestrian tracks when trajectory information alone is insufficient. This is particularly beneficial when tracking across multiple cameras. An overview of these three tasks is shown in Figure 2.7.

2.3.1 Pedestrian Detection

Pedestrian detection is a specific instance of the more general problem of object detection. Given an image, object detection aims to localise a set of object classes using a rectangle to represent the area it covers. While object detection involves both localising and classifying the object in question, the problem of pedestrian detection has only one object class, namely, pedestrians. This area has been of interest to researchers as it removes the classification component and can, therefore, focus on the localisation problem entirely. Due to the wide range of applications from surveillance [108] to robotics [67], many pedestrian detection datasets have been created, and standard evaluation metrics have been established to compare algorithms.

The standard metric for assessing pedestrian detection performance is the Average Precision (AP) at different Intersection-Over-Union (IOU) thresholds. Given a predicted and a ground truth bounding box annotated manually by a human expert, the IOU is the area of the intersection of the two bounding boxes divided by the area of the union of the two boxes. Common practice dictates that an IOU score greater than some threshold dictates a true positive. This threshold may range from a forgiving value of 0.5 to a more strict value of 0.8 [23]. Figure 2.8 shows a visual example of calculating the IOU.

Pedestrian detection can be considered the problem of finding a function to map the high dimensional image space to a much lower dimension coordinate space in which each coordinate represents a point on a rectangle in which a pedestrian is enclosed. Early work in finding this function use established computer vision features such as Histogram of Oriented Gradients (HOG) gradients [109]. However, state-of-the-art performance on pedestrian detection benchmarks has shifted in recent years from using classical computer vision feature representations such as HOG to representations learned by a CNN. Rather than creating features with human engineering using domain knowledge, CNNs use gradient descent to model machine learning problems as convex optimisation and optimise a loss function to find an effective feature representation.

Among the most popular datasets in this area are KITTI [99], Caltech-USA [110], Citypersons [111], and EuroCity Persons [112]. Although substantial progress has been made, even today's best performing pedestrian detection systems are still below human-level performance. In 2018, Zhang et al. [23] published a review of pedestrian detection progress and comparison with human-level performance. The authors conclude that methods are rapidly improving and show no sign of saturation on standard benchmark datasets. Despite excellent improvements, pedestrian detectors still fall short of human-level performance. Challenges include fake humans (e.g., statues), vertical

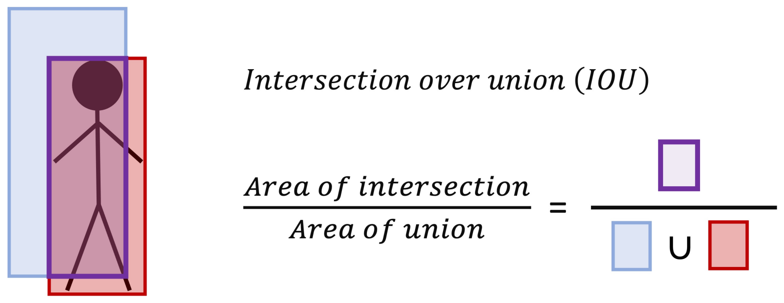


Figure 2.8: **Example pedestrian detection.** An example ground-truth bounding box is shown in red, a predicted bounding box in blue, and the overlap between the two in purple. A detection is considered successful if the ratio between the overlapping (purple) area and the union of the two boxes (blue and yellow) is greater than a certain threshold. This ratio is known as the Intersection-Over-Union (IOU).

structures, far-away pedestrians, occlusions, low image contrast, blur, and others. Since the publication of the review, more models bring the state-of-the-art closer to human-level detection performance [113–115]. If progress continues at a steady rate, these systems will soon surpass human-level performance on these benchmarks. This development opens the new problem of how to use these detections for more comprehensive scene understanding. Note that this comparison is in a single-frame setting, i.e., individual video frames without temporal context. Applying a tracking method to a video sequence following single-frame detection results can correct errors by using temporal information between video frames. Such tracking methods, therefore, improve performance further and are discussed in the next section.

2.3.2 Tracking

Tracking is the task of following the trajectory of an object(s) in a video sequence. Either a single object (Single Object Tracking (SOT)), or multiple objects (Multi-Object Tracking (MOT)) are tracked. In the SOT setting, the bounding box of an object is given in the first frame of a video sequence, and methods aim to track the object through the video. The SOT problem has seen extensive research. Basic methods used a window around the bounding box of the target in the previous frame to search for the target's new location and select the box with the highest cross-correlation value with the image intensity values [116]. The Kanade–Lucas–Tomasi (KLT) tracker selects features to track and then uses optical flow estimation to align features across frames [117]. Today's state-of-the-art trackers generally use deep neural networks for both features extraction and matching [118–120]. Methods designed to track pedestrians specifically follow the MOT setting, as methods for SOT are designed for tracking arbitrary objects, rather than pedestrians specifically.

We will focus our attention on MOT methods in this section.

In the MOT setting, multiple objects of a specific class (commonly pedestrians) are tracked. No prior information is given on the number of objects in the scene or their initial location, and methods must therefore detect the location of each object before tracking. Each unique object is assigned an identity number (or tracking ID) which should remain consistent for the duration of the video. Models are expected to track an arbitrary number of targets simultaneously. Methods for MOT typically follow a tracking-by-detection paradigm that relies heavily on the accuracy of a single-frame pedestrian detection model, such as those introduced in Section 2.3.1, followed by methods to associate the detections across video frames. This approach divides tracking into two distinct stages. The detector is applied to each frame individually in the first stage, generating temporally independent detection results. In the second stage, detections are associated across time by assigning a track ID to each bounding box. The bounding box locations may also be updated during this association stage, for example, by filtering the locations using a Kalman filter [121].

Reasonable MOT performance can be obtained with high-quality single-frame detections and simple constant velocity motion assumptions [121]. High-quality object detections combined with a high video frame rate simplify the matching process; however, several challenges remain. For example, two tracks that cross paths may result in an identity switch, where track IDs are switched between objects. Visual appearance is also often to alleviate this issue [122]. In this setting, a model extracts visual features associated with each target, such that the features from the same target across time are more similar than the features from a different target. The feature extraction and similarity metric computation are key components of person Re-ID, discussed in the following subsection.

Constructing more sophisticated methods capable of modelling non-linear motion can improve tracking performance, particularly in scenarios with occlusion [16]. However, trajectory forecasting for improved tracking is challenging due to small datasets, which results in forecasting models overfitting the trajectories in the training dataset. One approach proposed to overcome this issue is to consider the future trajectory as a binary classification problem [69] or using explicit external memory to avoid memorisation [123]. These methods alleviate the overfitting issue, but learning to predict non-linear trajectories is limited by training datasets' size.

2.3.3 Person Re-identification

Person Re-ID is the task of identifying the same individual in two different images, often from the perspective of different camera views. Given an image of an individual (the probe image), the task is to associate the image with another of the same individual from a set of gallery images containing several different identities. The probe and gallery images often exhibit variation in pose, lighting, occlusion, and scale. These variations make person Re-ID a challenging task. However, widespread applications for both single and multi-camera tracking have resulted in many benchmark datasets for person Re-ID [124] along with better and better solutions. Methods generally consist of two stages. The first stage involves extracting a visual feature vector from the probe image and each gallery image. A good feature vector encodes information useful for identifying individuals, such as personal appearance, clothing, and accessories. The second stage involves computing a feature distance score between the probe and each gallery feature vector associated with each image by using a distance metric, such that images of the same individual have smaller distances than images of different individuals. The top-n accuracy is given to evaluate performance, which corresponds to the proportion of data samples where the probe and gallery images depict the same individual given at most n guesses. For example, top-1 accuracy means only the image with the smallest distance is considered a match, whereas top-5 means the Re-ID is considered successful if the target individual appears in any of the five images with the shortest distance to the probe image. This metric is used along with others, such as Mean Average Precision (mAP) [124].

Substantial progress has been made in person Re-ID over recent years. For example, on the commonly used Market [125] benchmark, rank 1 performance (i.e. correct Re-ID given 1 guess only) has improved from 47.3% in 2015 [125] to 98.3% in 2021 [126]. Most work on person Re-ID has focused on image-level matching, i.e., associating images of the same identity without additional context. Current state-of-the-art methods [127, 128] exploit this visual cue without other sources of information. However, image-level similarity matching is just a single component of a comprehensive Re-ID system. Persons must first be detected and tracked before matching, usually resulting in multiple frames per detection, along with associated trajectory data. Incorporating trajectory information for Re-ID in a multi-camera setting has seen comparatively little attention, in part due to a lack of publicly available datasets. One notable exception is the recent work of Wang et al. [129]. The authors demonstrate that by retroactively utilising trajectory information and visual features, their approach can attain state-of-the-art Re-ID results on the prevalent Duke-MCMT benchmark dataset [101]. The Duke-MCMT dataset is no longer available to the research community [106].

Jain et al. [130] study the impact of scaling multi-camera tracking to large networks and show that filtering the search space to high traffic areas can considerably reduce the search space at only a small cost in recall. In addition to using the traffic level in each area, trajectory information has also been used for both Re-ID [129] and multi-camera tracking [131]. Using trajectories for these tasks can supplement existing appearance-based models by providing a second source of information for matching objects across camera views. However, previous works [129–133] reactively use trajectory information, i.e., the object must have already been observed in multiple camera views. These methods are therefore only suitable for offline tracking.

2.4 Learning with Less Human Labelling

Supervised learning has been a prominent paradigm requiring accurate annotation of datasets, commonly completed manually through painstaking human effort. Due to the massive quantities of data necessary to train state-of-the-art machine learning models effectively, several alternative means of supervision have been proposed. Pre-training neural network models on the large Imagenet dataset [134], before fine-tuning on a target dataset, has become the de-facto standard in settings where annotated data is limited. Alternative means of building large annotated datasets for pre-training i.e. transfer learning, such as mining social media websites [135] have also been proposed. This section reviews advancements in two methods used for reducing the burden of human labour in data annotation related to our contributions: machined-aided annotation and self-supervised learning.

2.4.1 Machine-Aided Data Annotation

To ease the manual effort required in manual data annotation, several methods have been proposed that use an algorithm to partially label the data in part and then use a human in the loop to correct the annotations. For example, object detection and tracking algorithms are trained using ground-truth object bounding boxes drawn around objects of interest, which are time-consuming to obtain by manually drawing a large number of boxes. ViTBAT [136] is a tool for labelling several object bounding boxes which combines tracking and classification labelling and allows the annotator to label several objects with the same class label simultaneously. Bounding boxes are interpolated between video frames, reducing the total number of bounding boxes the annotator needs to draw. ViTBAT also allows single object coordinates rather than boxes to be labelled and interpolated between frames, which can be used for quicker annotation of trajectory forecasting datasets. The tool assumes that the transformation between two bounding boxes or coordinates is linear between frames. Therefore, frames must be annotated with sufficiently small temporal windows, as frames annotated too sparsely will contain errors when the linear motion assumption does not hold. ViTBAT reduces but does not eliminate the human labour required to annotate an object tracking or trajectory forecasting dataset.

The goal of human action recognition is to assign a human action label to each video automatically. Kinetics [24] is a large dataset for human action recognition, where each video has an associated class of human action (e.g., running, brushing teeth). Kinetics contains over 300k videos depicting humans performing some action and a label of the action being performed. The dataset consists of trimmed videos; i.e., the videos are short 10-second clips centred around the action being performed. To aid in the trimming of the videos, the authors train a CNN to classify images of human actions. The image classifier is then used to obtain a confidence score for each frame in the video, and a 10-second trimmed video is extracted from the section surrounding the most confident video frame. Human annotators then verify that the video is trimmed correctly, rather than trimming the video manually, which considerably reduces the annotation time.

Semantic segmentation is the task of assigning each pixel in an image to an object class. Annotating data for semantic segmentation is notoriously time-intensive and can take up to 90 minutes per 2040×1016 image [43], even with expert annotators. Due to this, several methods have been proposed to ease the annotation procedure. LabelMe [137] is a tool that enables the annotator to draw a polygon around an object rather than labelling each pixel individually. Poly RNN+ [138] uses a CNN and RNN model to estimate the boundaries of an object from a user-provided bounding box, which the annotator may then correct. This approach reduces the number of clicks required to segment an object and, therefore, also an entire image. Lin et al. [139] propose to annotate images sparsely using a checkerboard pattern rather than labelling every pixel in the image. Their experiments demonstrate that using just 12% of the available training data with the Cityscapes semantic segmentation dataset [43] with the proposed annotation pattern results in a model with performance 98% of a model trained with the entire dataset. Each of the discussed methods eases the burden on human annotators, thereby enabling data to be collected quickly and cheaply. However, these methods still use a human annotator in the loop.

2.4.2 Self-Supervised Learning

Self-supervised learning aims to bypass the data labelling annotation process entirely by using some signal in the data as the supervision. In self-supervision, some subset of a dataset is withheld during the training process, and a model is trained to predict the withheld data. No manual data annotation is required, as the labels are present in the data itself. In this way, a model may exploit large-scale datasets without expensive annotation. However, the problem and data must be amenable to self-supervised learning, where some property of the data is withheld in the training process. Consider the problem of image colourisation. The goal of image colourisation is to predict the colour of each pixel in an image given a greyscale version of an image, such as a photograph taken using an old greyscale camera. In order to train a machine learning model to accomplish this task, we require a greyscale image (input data) and a corresponding colour image (target). Zhang et al. [140] create a model for this task by converting existing colour images to greyscale and train a deep neural network that generalises to greyscale images without corresponding colour versions. The proposed model is trained entirely using colour images that have been converted to greyscale, meaning no manual labelling is necessary. Other works follow the same self-supervision procedure for other tasks. For example, Ledig et al. create higher-resolution versions of images with image super-resolution [141], Chung et al. synchronise audio and video for automated lip-sync error detection [142], and Yu et al. remove unwanted portions of images using image inpainting [143]. Each work uses self-supervised training to create effective models without manual labelling of training data.

Some classically supervised tasks, such as image classification, are not as easily framed in a self-supervised setting. However, numerous works have shown the advantages of learning data representations using self-supervised training as a pre-training step to target supervised learning tasks. For example, Noroozi et al. [144] select 8 patches in an image surrounding a central randomly selected patch. The authors then randomly jumble the patches and train a model to recover the original patch structure. The weights from the pre-trained model are then used rather than a random weight initialisation on several classic supervised learning tasks such as image classification, object detection, and semantic segmentation. Initialising with the learned weights performs considerably better than the same model with randomly initialised weights. A plethora of other works follow a similar self-supervised representation learning followed by fine-tuning [145–148], and an extensive comparison of self-supervised representation learning effectiveness for common computer vision benchmarks is detailed in the recent survey on self-supervised learning by Jing et al. [38]. The survey finds that many forms of self-supervised pre-training improve target supervised learning task performance on various datasets. However, the effectiveness heavily depends on the choice of self-supervision and target tasks.

An alternative kind of self-supervision relies on cross-modal relationships as the supervisory signal. For example, in video, this can involve learning video-audio correspondence [149], using pixel values to predict optical flow

[150], or using data collected by one sensor (such as a camera) to predict the data collected by another sensor (such as a vehicle inertial measurement unit) [151, 152]. While some methods are used to learn the transformation between two modalities [150], modality transformation can be very challenging in some cases. Some methods alternatively learn to distinguish if two data samples of different modalities are corresponding or not [153], or temporally synchronise two modalities [149].

Self-supervision avoids the expensive human annotation component of supervised learning in certain situations and is, therefore, well-suited to address problems with limited annotated data for which a suitable supervisory signal can be obtained directly from data. Our proposed machine annotation scheme introduced in Chapter 3 enables us to leverage the power of self-supervision for pedestrian trajectory forecasting. To the best of our knowledge, our work is the first to apply a self-supervised training strategy to pedestrian trajectory forecasting. Our machine-aided annotation scheme for multi-camera data also introduced in Chapter 3 uses some similar components as the machine-annotation scheme. However, a human annotator refines the annotations, so this method is not considered an instance of self-supervised learning.

2.5 Summary

This chapter presented an overview of existing research related to our contributions in human trajectory forecasting and automating data annotation. Firstly, we contextualised trajectory forecasting by considering other vision-based forecasting works. The vision-based forecasting literature is growing steadily thanks to advancements in vision-based perception models, along with more extensive and more diverse datasets. We highlight the importance of predicting the future at the right level of abstraction, paying careful attention to the predictive time horizon and task when designing vision-based forecasting systems.

Secondly, we discussed literature on trajectory forecasting. By grouping works into BEV forecasting and egocentric approaches, we see a notable paucity of egocentric forecasting approaches and datasets compared to BEV works, which is the problem we will focus on in Chapter 4. We group works into both BEV and egocentric trajectory forecasting to hand-crafted and data-driven approaches, observing a similar trend towards data-driven methods for both problems. We also review trajectory forecasting approaches amenable to multiple camera views. This task has generally not been addressed in prior works, which will be covered in detail in Chapter 5. We summarise existing trajectory forecasting datasets used in the literature and observe a lack of large, diverse datasets for trajectory forecasting.

Thirdly, we review the literature on three perception tasks related to our contributions: pedestrian detection, tracking, and Re-ID. The three tasks are required prerequisites to many real-world applications of trajectory forecasting, and our review reveals substantial progress over recent years. Although not directly related to our contributions, methods for detection, tracking, and Re-ID are used throughout this thesis. These methods paved the way for much of our work, in particular, the data annotation methods proposed in Chapter 3.

Finally, we review literature proposing methods for minimising the human labelling overhead for training supervised machine learning models. We identify several works in both machine-aided data annotation and self-supervised learning, but a lack of works that employ these methods to the trajectory forecasting task. In Chapter 3, we propose methods for the machine-annotation and machine-aided annotation of trajectory forecasting datasets.

Chapter 3

Optimizing Data Annotation for Trajectory Forecasting

Datasets have a history of driving progress in computer vision. Those substantially larger or more comprehensive than those preceding them, such as ImageNet [134], Kinetics [24], and Cityscapes [43] have resulted in models that push forward the state-of-the-art. However, the trajectory forecasting task suffers from a paucity of large, publicly available datasets, particularly for egocentric and multi-camera trajectory forecasting. As discussed in Chapter 2, much of the existing work on pedestrian trajectory forecasting considers the problem from a birds-eye view using footage from a fixed overhead camera, often considering each pedestrian as a single point in space [32, 83, 84]. This setting is effective for modelling crowd motion patterns and interactions with the environment. However, by simplifying each pedestrian as a point in space, salient visual features such as person appearance, pose, posture, and individual characteristics are not considered. Prior research has shown that these features are of importance for trajectory prediction in settings such as anticipating if a pedestrian will cross the road [19, 21]. Furthermore, overhead perspectives are often not available in practical applications, such as for self-driving vehicles with vehicle mounted cameras. As a result, trajectory forecasting from an object-level perspective has been studied in recent years [31, 61, 154, 155]. However, it suffers from a lack of large, high-quality datasets.

This chapter introduces strategies for collecting and labelling datasets suitable for training and evaluating egocentric trajectory forecasting models while minimising the manual annotation effort required by a human annotator. Using our proposed strategies, we generate new annotations from an existing dataset (BDD-10K) and collect two new datasets (Citywalks and Warwick-NTU Multi-Camera Forecasting), which will be used throughout this thesis. Each of our datasets are publicly available to download to facilitate future research on trajectory forecasting. We propose a machine-annotation scheme for our single-camera data annotation with no human in the loop, and a machine-aided annotation scheme for our multi-camera data annotation which uses a human in the loop to assure annotation quality.

The rest of this chapter is organised as follows. In Section 3.1, we introduce our machine-annotation method and our new annotations for an existing dataset, as well as our new Citywalks dataset for egocentric trajectory forecasting. Section 3.2 introduces our machined-aided annotation framework for multi-camera trajectories and our new multi-camera trajectory forecasting dataset, Warwick-NTU Multi-Camera Forecasting (WNMF). In Section 3.3 we discuss future research directions made possible by our new datasets and annotation strategies along with details of other works that have used our datasets. The chapter is concluded in Section 3.4 with a summary.

3.1 Single-Camera Machine-Annotation Framework

Training trajectory forecasting models in a supervised learning setting requires dense (per-frame) bounding box annotation of pedestrians, which are time-consuming to obtain by hand. For this reason, the number and size of datasets with densely annotated pedestrian bounding boxes is limited. As discussed in Section 2.2.4, the size of existing datasets is prohibitive for the training of high-capacity deep learning models which rely on large quantities of data to learn an effective feature representation. To overcome this issue, we propose to learn a trajectory forecasting model from unlabeled data by using an automated pedestrian detection and tracking algorithm to generate bounding boxes without human labour.

3.1.1 Proposed Method for Machine-Annotation

Given a video sequence as input, our machine-annotation method uses a pedestrian detection algorithms obtain an estimate of the location L_t for each pedestrian, and a tracking method then links these estimated locations across each timestep t for each identity. Given a set of such detections, we adopt the self-supervision learning paradigm by training our model to predict future pedestrian locations, $L_{t+1:m}$, given only the current and past locations, viz., $L_{t-n:t}$.

Once frame-wise detections are obtained, detections are associated across frames using a tracking algorithm. We use the Deepsort [122] tracking-by-detection algorithm resulting in tracking identifiers associated with series of detections. The tracking-by-detection algorithm results in some false positives. To reduce the number of false positives, we apply a track filtering method, discarding detections with a height fewer than 50 pixels and tracks shorter than

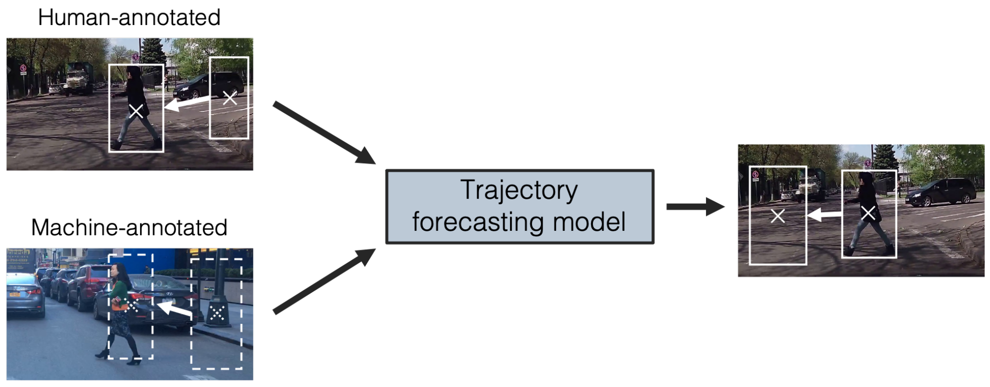


Figure 3.1: **Using human-annotated and machine-annotated training data.** We propose using both human-annotated and machine-annotated data to train a trajectory forecasting model.

25 frames. In practice, we find these constraints filter detections aggressively, removing 90% to 97% of initial detections. This strategy filters most false positives (non-pedestrians mistaken for pedestrians), as many false positives are less than 50 pixels in height or shorter than 25 frames in length. The number of tracks removed by this process is shown in more detail in Section 3.1.2. The result is a machine-annotated dataset, which can be used to supplement training on a smaller human-labelled dataset as shown in Figure 3.1. The resulting dataset contains pedestrian locations labelled and tracked over time, which can be used for trajectory forecasting. The location of a pedestrian in the early part of a track is used to predict the location of the pedestrian in the later part of the track.

Automated detectors do not perform on par with human annotators and make different errors to humans, such as false-positive detections of vertical structures [156]. Due to this, it is not self-evident that models trained on machine-annotated data will generalise across datasets and to human-annotated data. To verify our proposed machine-annotation system, we validate the performance of our proposed trajectory forecasting model and existing approaches on a human-annotated dataset. These findings are discussed in Chapter 4. In this chapter, we focus on the data collection process.

3.1.2 Vehicle-Mounted Camera Data Collection

We utilise our machine-annotation method introduced in the previous subsection to generate new annotations for an existing dataset of vehicle-mounted footage, where an outward-facing camera is mounted on a vehicle dashboard.

Motivation

Interacting with humans in complex urban environments remains a challenging problem for Autonomous Vehicles (AVs). Unlike highways with well-defined rules for traffic, urban environments necessitate that vehicles interact with other road users, such as pedestrians and cyclists, in a more nuanced manner. For an AV to navigate effectively in such environments, the vehicle must be able to locate and react to pedestrians in order to avoid collisions. The first component of such a navigation system, detecting pedestrians, has seen a tremendous amount of research effort in the past decade [157]. If current trends continue, performance will soon match and even surpass human-level performance on standard evaluation benchmarks [156]. The rapid advancements in this area have led to real-world implementations of ADASs to aid drivers in critical situations. Such systems can provide warnings or initiate braking if a pedestrian is detected in front of the vehicle but are less reliable at anticipating potentially dangerous events before a pedestrian steps into the roadway.

As vehicles move towards greater autonomy, the need for accurate pedestrian trajectory forecasting grows. ADASs may be designed conservatively with a human driver in the loop as false negatives can be tolerated. For an AV, however, the reliable anticipation of pedestrian *intent* is a critical safety feature but a complex challenge. Although driven by long-term motion goals such as reaching a specific destination [158], pedestrian motion is highly dynamic and may change at a moment's notice, such as a child running rapidly into the street. To deal with this uncertainty, human drivers use heuristics such as pedestrian head pose, gait, and scene dynamics to reason about intent [19]. Without these cues, human drivers find it more challenging to predict if a pedestrian is about to cross the road [21].

Modern vehicles equipped with sensors such as LiDAR and Radar can build an accurate representation of the surrounding environment [159]. Both LiDAR and Radar, however, lack the capability for extracting high-resolution features and are, thus, commonly supplemented with visible spectrum cameras. Manually annotating features such as pedestrian head pose and body language cues such as pose and posture from camera data is challenging and time-consuming, and automated state-of-the-art methods make more errors than human annotators [30]. Furthermore, pedestrian behaviour varies across different cultures and driving environments [160]. A model trained to anticipate pedestrian behaviour in California, USA, is unlikely to perform well on the streets of Mumbai, India. Without a practical method for learning from unlabelled data, it is likely that large quantities of data must be manually annotated for deployment in each environment.

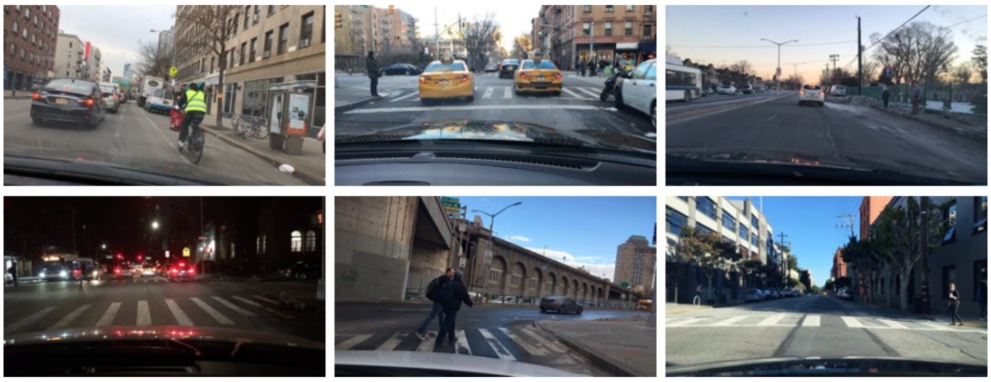


Figure 3.2: **Example video frames from BDD-10K.** The footage consists of both day and night drives in the US. As the data is crowd-sourced, there is some variety in camera placement.

BDD-10K

BDD-100K [105] is a large-scale dataset consisting of 100,000 videos shot from front-facing cameras mounted on car dashboards. The videos are crowd-sourced from over 50,000 rides around densely populated environment in the United States by 10,000 different drivers. The original dataset release [161] contained image classification, object detection, lane marking, drivable area, and semantic segmentation annotations. The latest release in 2020 [105] added Multi-Object Tracking and Multi-Object Tracking and Segmentation annotations for a subset of 2,000 and 90 videos, respectively.

The size and diversity of BDD-100K make it an ideal candidate to apply our machine-annotation method. Our method does not rely on human labour. However, due to the size of the dataset, we limit our annotations to the first 10,000 videos to reduce the computational requirements. We will henceforth refer to this subset as BDD-10K. We annotate pedestrian bounding boxes in the BDD-10K dataset using two popular off-the-shelf object detectors, YOLOv3 [162] and Faster-RCNN [163]. Although the detectors are capable of detecting a variety of objects, we use the pedestrian class only. Our aim here is to evaluate the robustness of our proposed machine-annotation system to different pedestrian detectors, rather than to compare detector performance directly. Nonetheless, for consistency, we train both detectors on the same dataset (MS-COCO [164]) and threshold confidence scores at 0.6. Using our proposed annotation scheme, we find a total of 16,900 valid non-overlapping pedestrian tracks using YOLOv3 and 13,200 using Faster-RCNN. Figure 3.3 shows typical pedestrian detection results from BDD-10K, and Figure 3.4 shows examples of false-positive detections examples that were successfully removed by our machine-annotation method. Additional information on the dataset is shown in Table 3.1. Our dataset is available for the research community to download

[1].

Table 3.1: **BDD-10K metadata.** We apply our machine-annotation method to the first 10,000 videos of BDD-100K to generate new annotations suitable for trajectory forecasting. Trajectories are filtered by restricting the minimum height of a bounding box to 50 pixels and minimum length of a track to 25 frames.

Video clips	10,000
Resolution	1280×720
Framerate	30hz
Fixed clip length	40 seconds
Time of day	Day/Night
Number of trajectories before filtering (YOLOv3 [162])	160,183
Number of trajectories after filtering (YOLOv3 [162])	16,908
Number of trajectories before filtering (Faster-RCNN [165])	456,845
Number of trajectories after filtering (Faster-RCNN [165])	13,173

3.1.3 Pedestrian-view Data Collection

Using videos from YouTube, we create a new dataset for pedestrian-view trajectory forecasting using our proposed machine-annotation method.

Motivation

As discussed in Chapter 2, many existing trajectory forecasting methods use a BEV. Forecasting from this perspective has two main shortcomings. Firstly, it is challenging to extract visual features from within a target pedestrian's bounding box from a BEV. Previous work has shown that visual features are useful for trajectory forecasting [21, 31]. Secondly, for several applications, a BEV is not available. For example, several methods have been proposed for assisting people with visual impairment navigate using wearable cameras [31, 166, 167]. BEVs are also not available for social robot navigation, where assistive robots interact with an environment share with humans [67].

Also as discussed in Chapter 2, there are few publicly-available pedestrian-view trajectory forecasting datasets. The few available datasets are limited in

[1]`https://github.com/olly-styles/Multiple-Object-Forecasting`

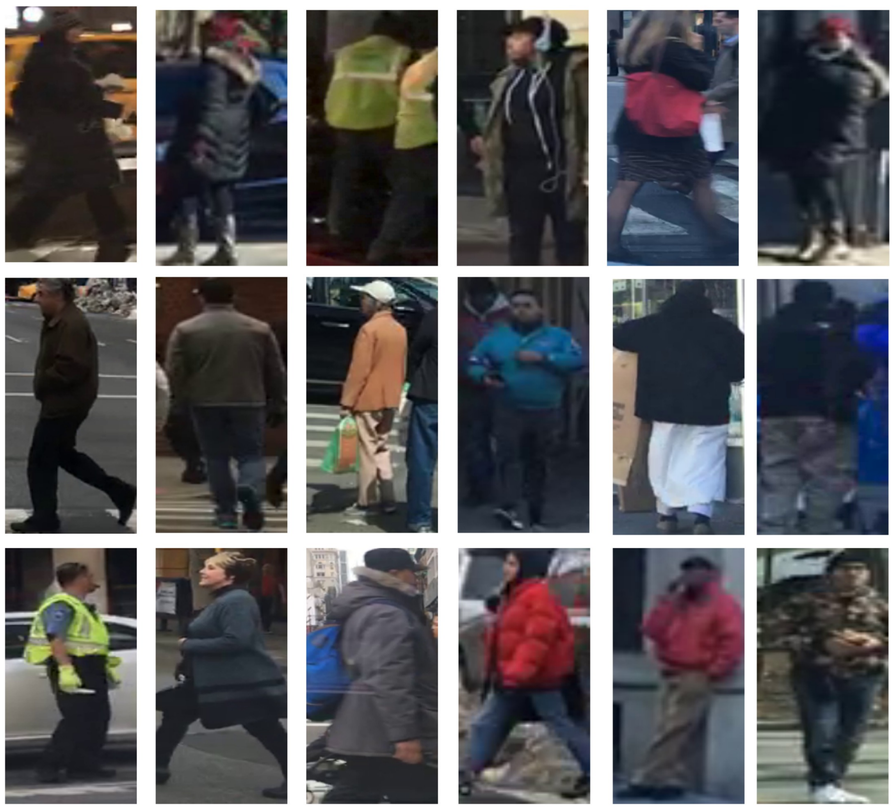


Figure 3.3: **Example pedestrians in BDD-10K.** Each is detected using our proposed machine-annotation method. False positives (non-pedestrians mistaken as pedestrians) are uncommon, although some bounding boxes are not as tight as human-annotated ground truth bounding boxes.

scale and diversity, thereby limiting the assessment of model generalisability. For example, datasets collected in laboratory conditions such as the Daimler dataset [88] contain just a few actors, which may have different movement patterns compared to classes of pedestrian which are not represented in the dataset, such as children and the elderly. This lack of representation can introduce algorithmic bias, as models will not be trained or evaluated on classes underrepresented or entirely absent in the dataset. We aim to address this limitation by collecting a large and diverse pedestrian-view trajectory forecasting dataset.

Citywalks

Our newly-constructed Citywalks dataset comprises 358 video sequences containing footage from 21 different cities in 10 European countries.

Data collection. We extract footage from the online video-sharing site YouTube[2]. Each original video consists of first-person footage recorded using an Osmo Pocket camera with a gimbal stabiliser held by a pedestrian walking

[2]Videos are obtained from `https://www.youtube.com/c/poptravelorg`

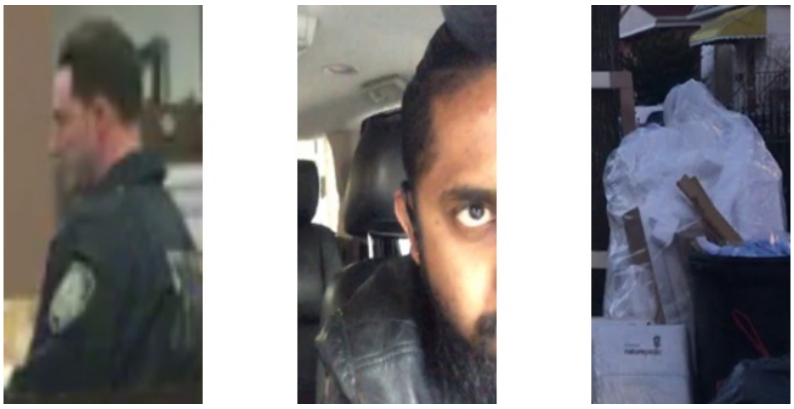


Figure 3.4: **False positive pedestrian detections.** False positive detections include partial detections, crowd sourcing errors, and objects classified as humans. The three examples shown here were all initially detected by the pedestrian detection but successfully removed during the filtering stage. The examples were chosen by hand.

in one of the many environments for between 50 and 100 minutes. Videos are recorded in a variety of weather conditions, as well as both indoor and outdoor scenes. Example frames showcasing the variety of the dataset are shown in Figure 3.6.

Video clip filtering. One of the fundamental challenges of egocentric forecasting is the bounding box motion caused by both ego-motion and object motion. Sudden ego-motion, such as the camera operator turning around a corner, will impact future bounding box location more than object motion. We aim for the Citywalks dataset to be used for modelling both motion sources; however, large displacements resulting from significant ego-motion may overwhelm the training process for trajectory forecasting models on Citywalks. To mitigate the impact of large ego-motions, we filter the dataset by removing high motion segments by estimating global motion each frame and removing frames above a manually tuned threshold. Global motion is estimated by extracting dense optical flow and selecting short video clips from windows with a mean optical flow magnitude below a threshold. Specifically, we downsample video frames to 128×64 pixels for faster computation and extract dense optical flow using FlowNet2-S [150]. We then select 20-second clips from longer videos using segments containing frames that do not exceed a mean optical flow magnitude threshold of 1.5. An example of the clip selection process is shown in Figure 3.5. This process restricts the distributions of motions in the dataset, which we find in practise removes large ego-motions.

Annotations. Once clips are selected, pedestrians are detected using an object detection algorithm. We generate annotations for two object detectors: YOLOv3 [162] and Mask-RCNN [168]. Both detectors are trained using the

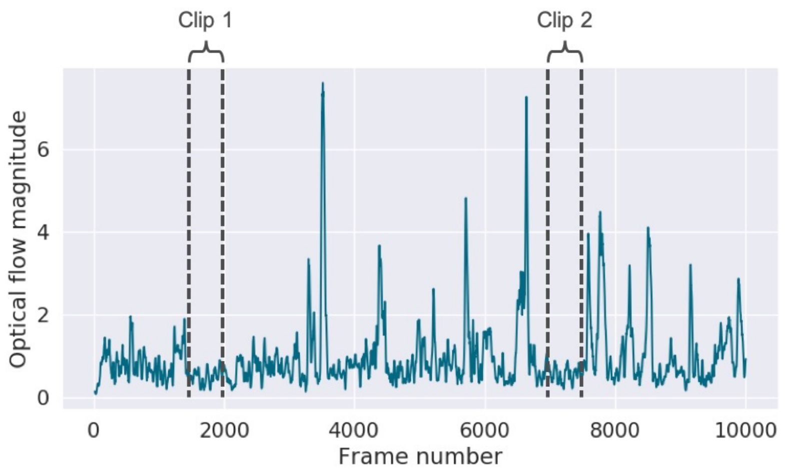


Figure 3.5: **Citywalks clip selection.** 20-frame clips are selected from segments with an average optical flow magnitude (pixel displacement per frame) below 1.5 for each frame.

MS-COCO [164] dataset and generalise well to Citywalks. For the YOLOv3 annotations, images are downsampled to 416 × 416 pixels before detection to simulate detection quality under low processing time requirements. We use a resolution of 1024 × 1024 for detection using Mask-RCNN to obtain the best detection performance. Note that we do not combine the two sets (such as in the work of Sanchez-Matilla et al. [169]) of annotations and treat the two separately. Following the detection phase, pedestrians are tracked using DeepSORT [122], which uses a Kalman filter and person Re-ID model is pre-trained on the MARS dataset [170]. We then discard tracks shorter than 3 seconds (90 frames at 30 FPS) as the previous 1 second of bounding box data is used to predict the next 2 seconds. Our goal is to have high-quality annotations with minimal false positives, and dropping short tracks reduces the number of false positives in the annotation set as we observe that erroneous tracks typically do not last longer than 3 seconds. Each video clip is also manually annotated with the recording city, time of day, and weather condition. We consider a clip to be shot at night time if street lights are illuminated, and in day time otherwise. Evaluating the quality of annotations is challenging, as the large unlabelled dataset has no ground truth with which to compare. However, by comparing the statistics in Figure 3.7, we can see some differences between the two detectors. Mask-RCNN detects more pedestrians and, in particular, detects more pedestrians smaller in height. This difference is likely due to the higher resolution of input images to Mask-RCNN compared to YOLOv3. We will further test the annotation quality in Chapter 4 by training trajectory forecasting models on Citywalks and then evaluating them using a human-annotated dataset. Citywalks metadata are shown in Table 3.2. Our

Figure 3.6: **Example frames from the Citywalks dataset.** Citywalks is markedly larger and more diverse than existing datasets.

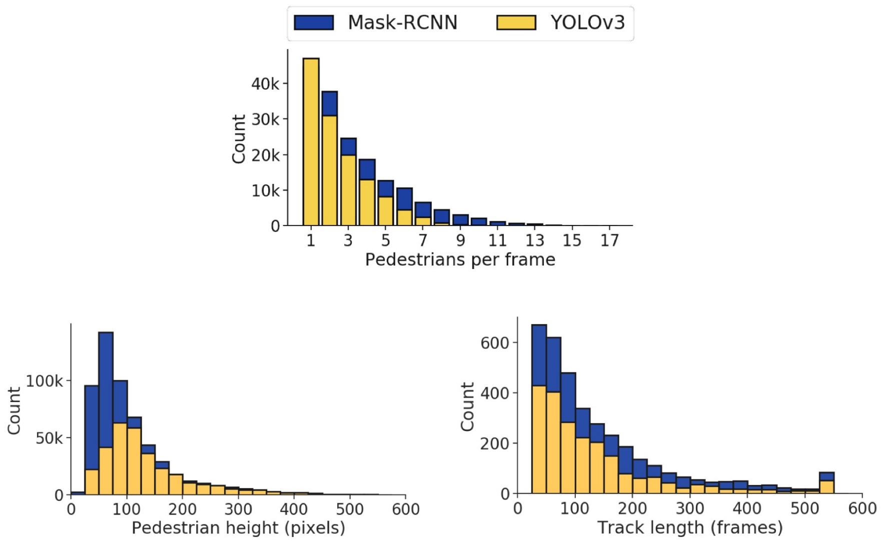


Figure 3.7: **Citywalks annotation statistics.** Mask-RCNN detects more pedestrians than YOLOv3, particularly small-scale pedestrians. However, unlike Mask-RCNN YOLOv3 runs in real-time.

dataset is available for the research community to download [3].

3.2 Multi-camera machine-aided annotation framework

In this section, we introduce our proposed machine-aided annotation method for Multi-Camera Trajectory Forecasting (MCTF) data. MCTF is a new task within trajectory prediction where an object's future trajectory is predicted across multiple camera views. For example, as a target leaves the field of view from one camera, its future trajectory is predicted with respect to other cameras, such that the object can be re-identified in another camera view. This

[3]`https://github.com/olly-styles/Multiple-Object-Forecasting`

Table 3.2: **Citywalks metadata.** Our annotation strategy enables us to collect a large and diverse dataset

Video clips	358
Resolution	1280×720
Framerate	30hz
Clip length	20 seconds
Unique cities	21
Weather conditions	Sun/Rain/Snow/Overcast
Time of day	Day/Night
Labelled objects per frame	0 - 17
Unique tracks (YOLOv3)	2201
Unique tracks (Mask-RCNN)	3623

approach is in contrast to traditional SCTF methods and datasets which rely on a single camera view only. Similarly to our machine-annotation method for SCTF datasets, we use algorithms for automated detection and tracking of pedestrians in a scene. This method is extended with additional steps for annotating tracks across multiple cameras views.

3.2.1 Motivation

Multi-camera systems are widespread today. For example, many buildings are fitted with CCTV systems for security. Multi-camera systems are also used in sports analysis [171] and road traffic monitoring [172]. The applications of a MCTF system are fourfold: (i) Long-term forecasting. This is possible by removing the constraint of a single camera viewpoint. (ii) Intelligent camera monitoring. When tracking a particular object-of-interest across a camera network, a subset of cameras may be monitored intelligently using the predictions from an MCTF model rather than continually monitoring all cameras. (iii) Enhanced tracking. Location predictions may be used in conjunction with a RE-ID model for more robust multi-camera tracking. (iv) Robustness to camera failure. Predicting an individual's location in multiple camera views adds redundancy; i.e., a target may still be identified if one or more cameras on the network are no longer operational.

We identify three significant challenges when collecting multi-camera datasets for trajectory forecasting. (i) Data annotation. Annotating data is a time-consuming and tedious process. Not only must trajectories in each camera be labelled, but the trajectories must also be associated across camera views.

The time required to associate trajectories manually increases significantly as more cameras are added to the camera network. (ii) Privacy limitations. The data from existing multi-camera networks, such as CCTV systems, are often subject to privacy restrictions and not accessible for academic research. (iii) Setup and synchronisation. Operators must temporally synchronise each camera in a multi-camera dataset to be suitable for trajectory forecasting. Internet Protocol (IP)-based camera systems can use a time syncing protocol such as Network Time Protocol (NTP) [173] to synchronize cameras accurately. If using regular (i.e. non-IP-based) video cameras, cameras need to be synchronised using alternative methods. A human annotator may achieve frame-level accuracy in overlapping cameras by manually identifying the same moment in a pair of cameras. Alternatively, audio tracks can be synchronised to sub-frame-level latency [174], assuming the captured audio is sufficiently similar.

Due to these challenges associated with MCTF data collection, few public datasets are available. The complexity of the annotation task makes manual data annotation infeasible for creating a dataset large enough to train and evaluate MCTF models. To address this problem, we propose a semi-automated method that uses existing object detection, tracking, and re-identification methods to aid the process. The process uses human annotation to assure data quality but reduces the work required by a considerable amount.

3.2.2 Multi-camera view data collection

Video collection. We collect a new database of 600 hours of video footage from 15 overhead mounted cameras set up indoors on the *Nanyang Technological University* campus. The Internet Protocol (IP)-based cameras are temporally synchronized and accessible through online monitoring software. Each camera is placed with a view of either a corridor or a junction. We describe our semi-automated data labelling method and collected dataset below. The footage is recorded for 20 days in 20-minute long segments collected evenly during the daytime. Example frames and the camera network topology is shown in Figure 3.8.

Implementation details. We use Mask-RCNN [168] pre-trained on the MS-COCO dataset [164] as our pedestrian detection model, with a detection confidence threshold of 0.5. We use DeepSORT [122] as our tracking model, for which the person Re-ID component is pre-trained on the MARS dataset [170] as with our SCTF machine-annotation method. Our annotation method produces a large set (approximately 2000) of verified departure-entrance pairs. A single individual is labelled for each cross-camera trajectory, and interactions between individuals are limited due to low person density (1.41 individuals

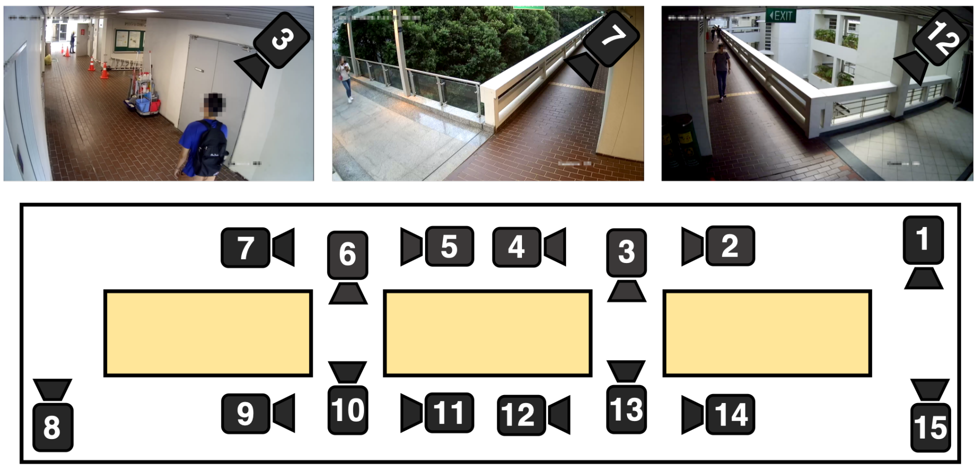


Figure 3.8: **Example frames and camera network topology.** We use a 15 camera setup, each of which is synchronized and recording simultaneously.

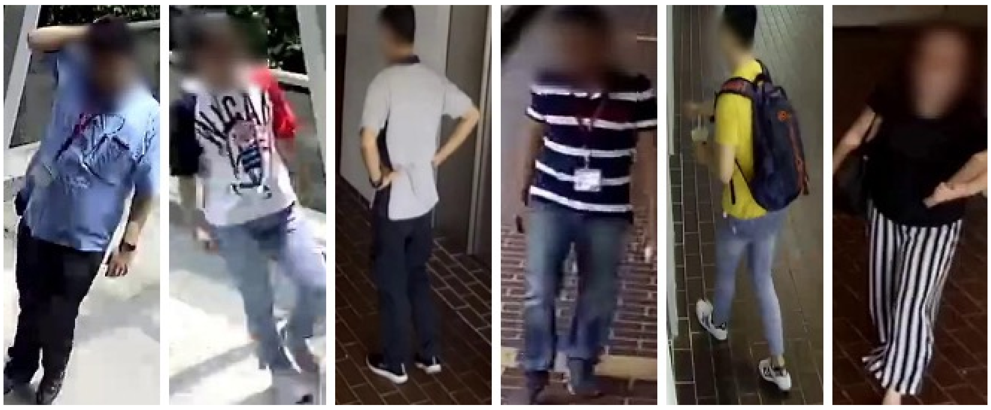


Figure 3.9: **Face blurring examples.** We apply a Gaussian blur filter to the detected face region to preserve the privacy of individuals in the dataset.

per camera per frame on average). A human annotator verifies that each track consists of bounding boxes judged to have an IOU of ≥0.5 with the ground truth box. We do not collect identity labels; however, in practice, we observe that many cross-camera trajectories are unique identities due to the long data collection period. To keep private the identity of the individuals in the WNMF database, we use the RetinaFace [175] algorithm to detect faces in each video frame and then apply a Gaussian blur to the detected region. This step was completed immediately prior to the data release, after the person Re-ID stage, so image processing were not impacted. RetinaFace [175] is a state-of-the-art algorithm for face detection and can identify faces from any orientation, including partially occluded faces. The algorithm detects the face region only, so we double the detected region's size to include the region directly surrounding the face. Example blurred images are shown in Figure 3.9.

3.2.3 Proposed Method for Machine-Aided Annotation

To minimise the need for manual annotation, we propose a semi-automated method that uses a combination of off-the-shelf models for detection, tracking, and person RE-ID. These results are then manually verified to ensure that proposed tracks are accurate and correct cross-camera correspondences for pedestrians are found. An overview of this annotation method is shown in Figure 3.10, which consists of the following three steps:

(i) We run pre-trained object detection [168] and tracking [122] models to locate and track pedestrians in each of the k cameras. The first 20 bounding boxes of a track form an entrance tracklet, $E^i_t = \{b^i_t, \cdots, b^i_{t+20}\}$, where b^i_t is a the bounding box of object i at timestep t. Similarly, the last 20 frames of a track form a departure tracklet $D^i_t = \{b^i_{t-20}, \cdots, b^i_t\}$. We define the camera numbers of entrance and departure tracklets as c_E and c_D, respectively. We also define the first timestep of the entrance and departure tracklets as t_E and t_D, respectively. These tracklets represent the moments a pedestrian enters and exits a scene. 20 frames was chosen as the minimum, as at 5 FPS, this results in 4 seconds of data which is sufficient to complete the following steps.

(ii) We find cross-camera identity matches between all the departure and entrance tracklets. We use a person RE-ID model [128] to compute RE-ID features for each image and store the mean feature vector for the tracklet. We then compute the visual similarity between the entrance and departure tracklets that appear in different cameras (i.e. $c_E \neq c_D$) by computing the squared difference in their RE-ID features, $(R(E^t_c) - R(D^t_c))^2$, for all entrance and departure tracklets found in step (i), where $R(x)$ denotes the RE-ID feature vector of tracklet x. We use only this distance metric, and leave experimentation with other metrics, such as cosine distance, to future work. We retain those with a squared difference below a manually specified threshold, $\delta = 0.0015$. Through manual inspection of data samples, we find that at this threshold most true positive matches are retrieved. This threshold is set deliberately high as we wish to have a high recall of cross-camera matches to maximise the number of matches for our dataset. We are less concerned about precision, as false-positives are discarded during step (iii). In addition, we constrain candidate tracklets within a manually specified time window γ to cut down the search space of possible matches, i.e., we compare only those tracklets which satisfy $t_E - t_D < \gamma$. This parameter is specific to the camera network, as a larger network with non-overlapping cameras that are more spread apart will need a larger value of γ to give targets time to reach the view of the next camera. As we set $\gamma = 12$ seconds, the matches are generally from neighbouring cameras in the network. γ was tuned manually, and is dependent on the distance between cameras in the network. We confirmed this

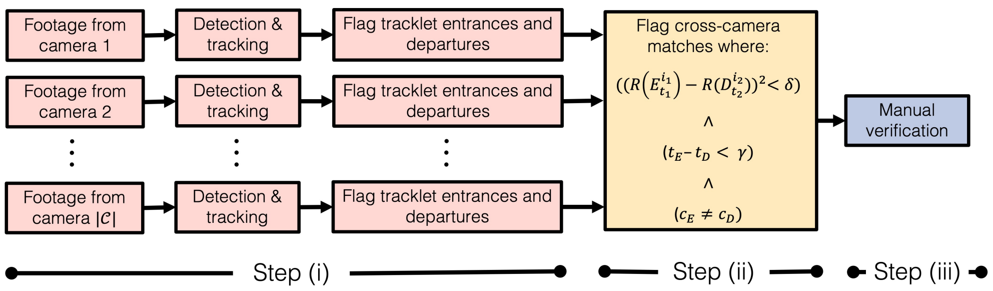


Figure 3.10: **Annotation method.** The proposed method generates the labeled data required for MCTF with minimal human labor by using automated methods for detection, tracking, and person RE-ID before a final manual verification step. See the corresponding steps in Section 3.2.3 for a full list of parameters.

Table 3.3: **WNMF metadata.** Our annotation method extract around 2000 multi-camera trajectories from 600 hours of raw footage. Trajectories are verified manually by a human annotator to confirm that the each cross-camera match corresponds to the same individual.

Hours of footage	600
Number of cameras	15
Collection period	20 days
Time period	8:30am – 7:30pm
Video Resolution	1920 × 1080
Frames Per Second (FPS)	5
Cross-camera matches	13.2K
Cross-camera matches after verification	2.0K
Mean cross-camera RE-IDs per track	2.08

by comparing the camera transitions with respect to the network topology in Figure 3.11. Our annotation method results in a set of cross-camera transition pairs $\mathcal{P} = \{(E^t, D^t)\}$.

(iii) Finally, we manually verify whether every match proposed by the algorithm is a true positive. We do this by comparing entrance and departure tracklet pairs proposed by the algorithm and confirming if the same individual is indeed present in both by hand. The manual verification step assures annotation quality, as false matches and bad detections are discarded. Table 3.3 tabulates WNMF metadata, showing 11.2K such bad matches were discarded.

As the human annotator only needs to verify the cross-camera matches rather than finding them from raw videos, the manual overhead is considerably lower than fully manual data annotation.

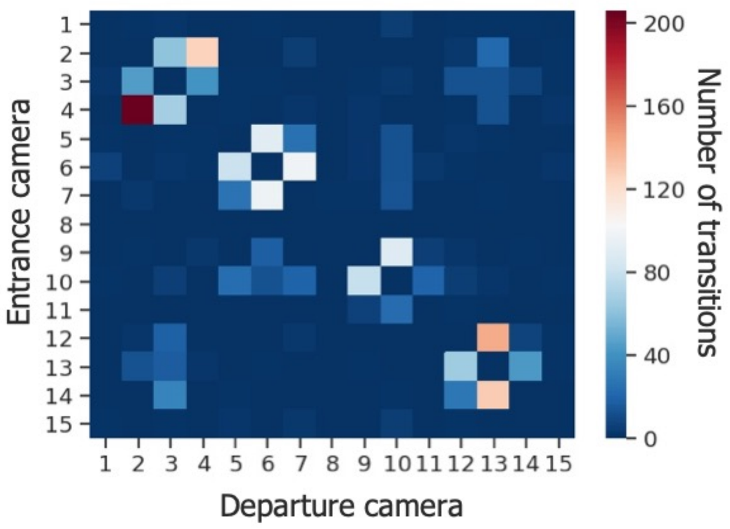


Figure 3.11: **Transition frequency between cameras.** Nearby cameras, such as 2 and 4, have a high number of transitions, whereas cameras far apart have few (often 0) transitions.

3.2.4 Summary

We have collected the WNMF database for MCTF using a new machine-aided annotation strategy. The WNMF database is available online for the research community[4]. We provide tracking data, pre-computed Re-ID features, full multi-camera trajectories. At the time of writing, we have granted 17 requests to access the WNMF dataset.

3.3 Discussion

Our proposed machine-annotation framework for SCTF datasets enables the collection of large-scale datasets with considerably less human labour than a fully-manual approach. The human annotation step is often a limiting factor in dataset collection, as obtaining unlabelled data is often available thanks to online resources. Therefore, our machine-annotation framework means that the size of future datasets will be limited by the amount of unlabelled data and computation resources available. However, it remains unknown how well models trained using machine-annotated data will generalise to human-annotated datasets. Will models generalise, despite differences in the data collection process? These questions are explored in Chapter 4, where we train trajectory forecasting models using our machine-annotated data and evaluate the performance using human-annotated data.

[4]`https://github.com/olly-styles/Multi-Camera-Trajectory-Forecasting`

Our proposed machine-aided annotation framework for MCTF has facilitated the collection of the WNMF dataset, which, to the best of our knowledge, is the first multi-camera dataset collected using a standard CCTV setup for trajectory forecasting. The dataset enables several problems to be studied, such as how to predict trajectories across multiple camera views, how to model a targets location when visible in multiple camera views, automatic camera topology learning, and how to use trajectory information for person Re-ID. Our machine-aided annotation framework may be used with other multi-camera networks, enabling new datasets to be collected more rapidly than manual data annotation. In Chapter 5, we develop a framework for forecasting pedestrian trajectories in a multi-camera environment using the WNMF dataset.

3.4 Summary

Our proposed single-camera machine annotation strategy enabled us to collect larger and more diverse datasets for trajectory forecasting than those previously available to the research community. By using state-of-the-art methods for pedestrian detection and tracking followed by an automated filtering strategy to remove false positive object tracks, we created a high-quality dataset without manual labelling. For our single-camera annotation method, our annotations have been generated algorithmically; therefore, it is not given that models trained on our machine-annotated datasets will generalise to human-annotated data. This issue is explored in the following chapter, where we train models on our collected datasets and evaluate them using existing human-annotated datasets.

Our proposed multi-camera machine-aided annotation strategy enabled the collection of the WNMF dataset. The dataset is the first dedicated to Multi-Camera Trajectory Forecasting (MCTF), opening new possibilities for future research in multi-camera environments. After anonymising the dataset by blurring the faces of individuals in the dataset, we release WNMF to the research community to further facilitate future research on MCTF. MCTF is a new problem in the trajectory forecasting literature, and will be explored in detail using the WNMF dataset in Chapter 5.

Chapter 4

Egocentric Single-Camera Trajectory Forecasting

Video-based trajectory forecasting models are often grouped into two categories: Bird's Eye-View (BEV) approaches and egocentric approaches. BEV trajectory forecasting models use a top-down perspective to predict future object trajectories using video captured from overhead, such as drone footage. Egocentric approaches, in contrast, use video footage captured at an object-level perspective. Many existing trajectory forecasting works use footage shot from a BEV, which is effective for modelling interactions between targets and the surrounding environment. However, extracting visual features from targets, such as pedestrian gait, is challenging from a BEV perspective. Egocentric trajectory forecasting video sequences often contain higher-resolution footage of targets from which visual features can be extracted more easily. Furthermore, egocentric viewpoints are commonplace for practical robotics applications, such as autonomous vehicles. An egocentric viewpoint, however, introduces new challenges, such as changes in object scale and egomotion.

In this chapter, we formally introduce the SCTF task and propose two models for egocentric SCTF. The first model, Dynamic Trajectory Predictor (DTP), uses footage captured from on board a moving vehicle to predict the future trajectory of pedestrians. We test the model on several scenarios, including many potentially dangerous scenarios where a pedestrian steps out into the vehicle's path. We train DTP using a dataset annotated with our proposed machine-annotation framework introduced in Chapter 3, and evaluate on a human-annotated dataset. The second model, Spatio-Temporal Encoder-Decoder (STED), predicts both object scale and trajectory using a novel two-stream encoder-decoder architecture. We evaluate STED on the Citywalks dataset, also introduced in Chapter 3, as well as another human-annotated dataset.

The rest of this chapter is organized as follows. In Section 4.1, we introduce the traditional SCTF problem formulation and our new Multiple Object Forecasting formulation. In Section 4.2, we describe the datasets used in this chapter. In Section 4.3, we introduce our two proposed approaches and present experimental results. In Section 4.4 we discuss the implications of the contributions of the chapter. The chapter is concluded with a summary in Section 4.5.

4.1 Problem Formulations and Baselines

In this section, we formalize the existing egocentric SCTF problem and introduce the new task of Multiple Object Forecasting (MOF). We also introduce baselines for each task.

4.1.1 Egocentric Single-Camera Trajectory Forecasting

Problem formulation

We defined trajectory forecasting more generally in Chapter 2 as the task of predicting a sequence of object locations $L_{t+1:m}$ given a sequence of past locations $L_{t-n:t}$. Each location is comprised of a coordinate $(x, y)_t$. From an egocentric perspective, we define the location of an object as the centre point of its bounding box, b_t^i. This formulation enables us to use an object detection model to predict an object bounding box and then convert this to a location by removing the height and width components of the predicted bounding box. Note that the visual information contained in video frames $f_{t-n:t}$ may be used in addition to the sequence of locations $L_{t-n:t}$ for learning features useful for the egocentric trajectory forecasting task.

Our focus here is on predicting the centroid in the 2D coordinate space obtained by a camera rather than the 3D world coordinates. For certain applications, 2D object detections may be associated with 3D world coordinates using a depth estimation method such as the one proposed by Hirschmuller [176].

Baselines

We use Constant Velocity (CV) and a Linear Kalman Filter (LKF) as baselines. These two baselines are used widely for comparison in both egocentric [31, 61, 89, 90, 92] and BEV [32, 33, 76, 80, 83, 84] trajectory forecasting literature.

Constant Velocity. We define the horizontal and vertical components of velocity, v_t^x and v_t^y, at time t of a pedestrian relative to the camera in the 2D projection obtained by a camera can be estimated by taking the first order derivative of the past n locations:

$$v_t^x = \frac{x_t - x_{t-n}}{n} \quad , \quad v_t^y = \frac{y_t - y_{t-n}}{n} \tag{4.1}$$

We use only the past n locations as the trajectory may be long, over which time the velocity can change considerably. As a baseline, we consider that the pedestrian maintains their average velocity of the previous n timesteps in the future m timesteps:

$$\widetilde{x}_{t+m} = x_t + v_t^x \cdot m \quad , \quad \widetilde{y}_{t+m} = y_t + v_t^y \cdot m \tag{4.2}$$

Note that for this problem formulation, we only consider the object centroid. We consider scale estimates in Section 4.1.2. We denote velocity at time t as v_t, comprised of vertical and lateral velocities v_t^x and v_t^y. The predicted location of the centroid at time $t + m$ following the constant velocity assumption is denoted as:

$$\widetilde{L}_{t+m} = L_t + v_t \cdot m \tag{4.3}$$

Linear Kalman Filter (LKF). The LKF [74] is a recursive Bayesian filtering technique for estimating the value of some variable given a sequence of noisy measurements. By estimating a probability distribution for the variable over time, the estimated value of the variable is usually more accurate than any of the single measurements in isolation. The LKF filters noise in the measurements, which has the effect of smoothing predictions for the trajectory forecasting task. In this case, past object locations are the measurements.

4.1.2 Multiple Object Forecasting

A shortcoming of egocentric SCTF methods is that object scale is not considered as part of the prediction, as object location is described only in terms of its centroid. We introduce the task of MOF to address this, which is an extension of the widely-studied Multi-Object Tracking (MOT) problem.

Problem formulation

Consider a sequence of n video frames $f_1, f_2, \ldots, f_n$. Given the t^{th} frame f_t, the task of object detection is to associate each identifiable object $i \in \mathcal{I}$ in the frame with an object bounding box $b_t^i = (x_t, y_t, w_t, h_t)$ which represents the centroid (x_t, y_t), width, and height of the object bounding box, and $\mathcal{I}$ is the set of all identifiable objects in the video. Given the framewise detections $\{b_1^i\}, \{b_2^i\}, \ldots, \{b_n^i\}$ for all $i \in \mathcal{I}$, the task of MOT is to associate each detection

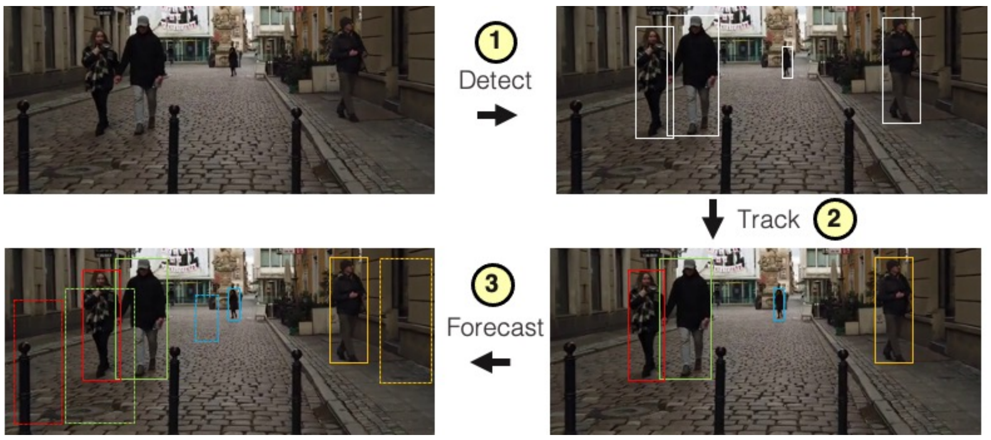


Figure 4.1: **Multiple Object Forecasting.** Given detection (1) and tracking (2) results, Multiple Object Forecasting involves the forecasting (3) of future object bounding boxes. Target object trajectory, scale, and camera egomotion must all be accounted for to accurately forecast future object bounding boxes.

b_t^i with a unique object identifier $j \in 1, 2 \ldots |\mathcal{I}|$, where $|\mathcal{I}|$ is the total number of unique objects across all frames, such that each object is tracked across the set of frames.

We extend the MOT task to MOF, as shown in Figure 4.1. Given $f_{t-n}, f_{t-n+1}, \ldots, f_t$ with associated object detections $\{b_{t-n}^i\}, \{b_{t-n+1}^i\} \cdots \{b_t^i\}$ and tracks, we define MOF as the joint problem of predicting the future bounding boxes $\{b_{t+1}^i\}, \{b_{t+2}^i\}, \ldots, \{b_{t+m}^i\}$ and associated object tracks of the upcoming $f_{t+1}, f_{t+2}, \ldots, f_{t+m}$ video frames for each object present in frame f_t, where n is the number of past frames used as input and m is the number of future frames to be predicted. The MOF task is similar to the egocentric trajectory forecasting task, with the inclusion of object scale in addition to trajectory.

Baselines

Constant velocity & constant scale. Similarly to the constant velocity baseline, the constant velocity & constant scale baseline linearly extrapolates the object location, i.e., the bounding box centroid. A constant value is predicted for the object scale, i.e., the predicted bounding box height and width is the last observed height and width.

Linear Kalman Filter. The LKF baseline can be applied to MOF in the same way as with egocentric SCTF. The future height and width values are predicted in the same way as the centroid.

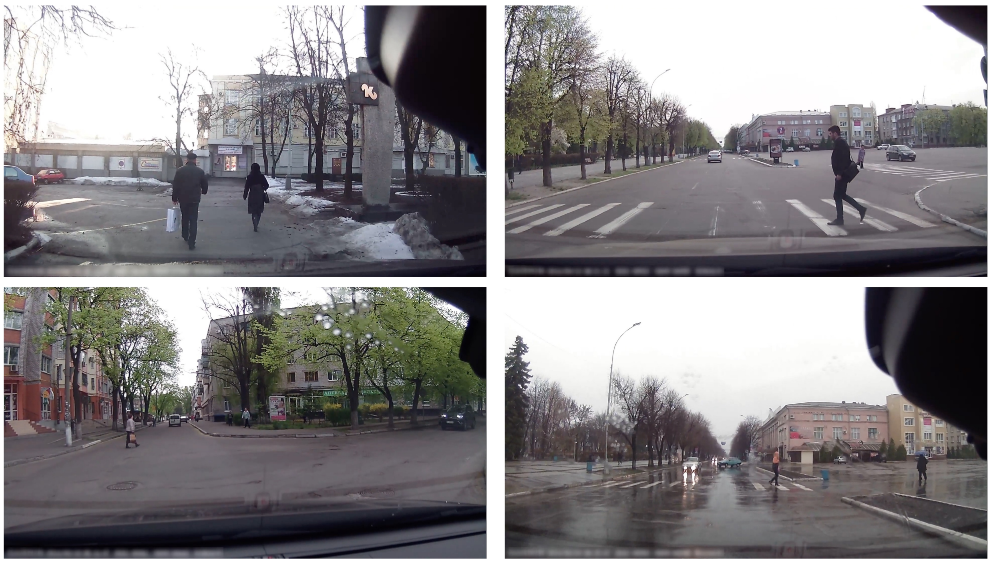


Figure 4.2: **Example frames from JAAD.** Different cameras and vehicles are used for data collection, so there is some difference in camera placement.

4.2 Datasets

We use two datasets for model evaluation in addition to our BDD-10K and Citywalks datasets introduced in Chapter 3.

4.2.1 Joint Attention in Autonomous Driving

The JAAD dataset introduced by Kotseruba et at. [64] consists of 346 short video clips of footage shot using a front-facing camera mounted on a vehicle dashboard. The dataset features many interactions between drivers and pedestrians, such as individuals crossing the road or yielding to the vehicle. Each video is accompanied by full annotation of pedestrian bounding boxes, behavioural data, weather conditions, time of day, and other information. We use only the pedestrian bounding boxes in our analysis. Example frames from the dataset are shown in Figure 4.2, and dataset metadata are listed in Table 4.1. We make a few modifications to the JAAD dataset for evaluating our models. Namely, pedestrians smaller than 50 pixels in height, occluded pedestrians, and tracks shorter than 25 frames are discarded. These values were chosen manually, as pedestrians that do not this criteria are more challenging to extract visual features from.

4.2.2 MOT-17

The MOT-17 dataset [46] was originally created for studying the MOT task. However, as full object bounding boxes and tracking information is provided, the dataset is also suitable for the MOF task. The dataset features BEV

Table 4.1: **JAAD metadata.** JAAD is a large annotated dataset for studying driver-pedestrian interaction.

Video clips	346
Resolution	1280 × 720 & 1920 × 1080
Framerate	30hz
Clip length	5-15 seconds
Unique cities	5
Weather conditions	Clear/Snow/Rain/Cloudy
Time of day	Day/Night

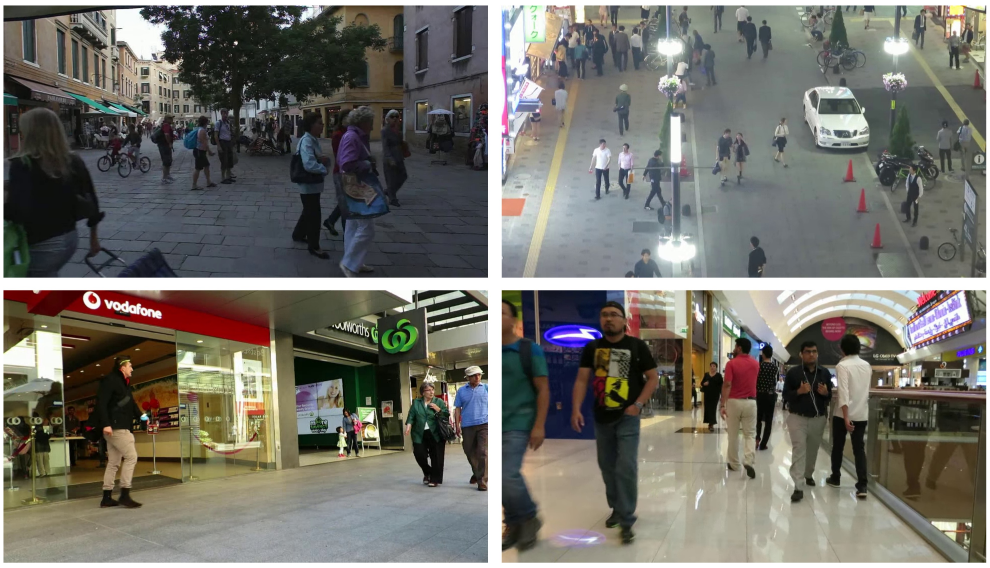


Figure 4.3: **Example frames from MOT-17.** As we are interested in forecasting from an egocentric perspective, we do not use video clips filmed from overhead, such as the top right frame.

and egocentric footage and full annotations of object bounding boxes and tracking information. Example video frames are shown in Figure 4.3 and dataset metadata are shown in Table 4.2.

4.3 Proposed Approaches

In this section we introduce our proposed models: DTP and STED.

Table 4.2: **MOT-17 metadata.**

Video clips	14
Egocentric video clips	10
Resolution	640×480 & 1920×1080
Framerate	30hz & 14hz
Clip length	17-85 seconds

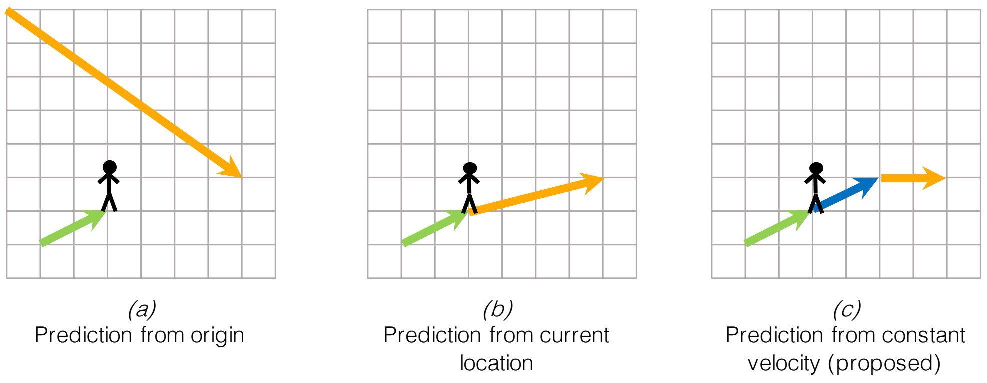

(a)
Prediction from origin
(b)
Prediction from current location
(c)
Prediction from constant velocity (proposed)

Figure 4.4: **Constant velocity correction.** Given an observed trajectory (**Green**), SCTF aims at predicting the future trajectory of an object (**Yellow**). Existing works have shown that a prediction vector relative to the current object location *((b))* is more effective that predicting from the origin *((a))*. We propose to predict the object's deviation from constant velocity (**Blue**) *((c))*.

4.3.1 Dynamic Trajectory Predictor

Method

Inspired by the effectiveness of action recognition models that use optical flow as an input modality [178, 179], DTP uses a stack of optical flow frames as input to a CNN that extracts a compact representation of human and camera motion. In many scenarios, such as when a pedestrian is stationary or walking at a constant speed, the constant velocity baseline reasonably predicts their future location. Challenging situations deviate significantly from this assumption, such as when a pedestrian starts walking or abruptly changes direction. An effective model must anticipate a change in velocity and adjusts predictions accordingly. The error resulting from the constant velocity baseline is denoted by:

$$\widetilde{e}_{t+m} = |L_{t+m} - \widetilde{L}_{t+m}| \tag{4.4}$$

Rather than predicting a location $\hat{L}_{t+m}$ directly, existing works [31, 92] output the location relative to the last observed timestep, $\Delta L_{t+m} = L_{t+m} - L_t$.

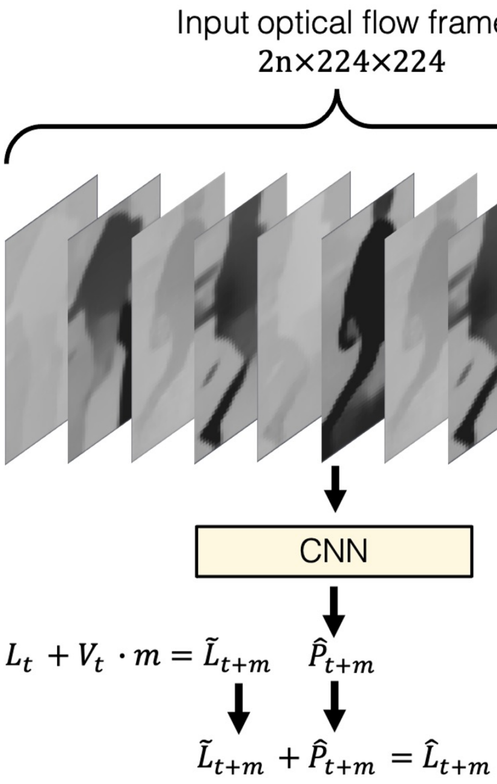

Input optical flow frames
2n×224×224
CNN
$L_t + V_t \cdot m = \tilde{L}_{t+m}$
$\hat{P}_{t+m}$
$\tilde{L}_{t+m} + \hat{P}_{t+m} = \hat{L}_{t+m}$

Figure 4.5: **Dynamic Trajectory Predictor.** DTP forecasts pedestrian trajectory relative to a constant velocity baseline. We use ResNet [177] with modified input and output layers to compute features from past optical flow. The optical flow magnitude is dependent on both object and ego motion.

Instead, we propose to output a compensation term, $P_t = -\widetilde{e}_t$, which corrects for errors in the constant velocity baseline. The difference between these three representations is shown in Figure 4.4. By training our model to predict C_t, the model is first initialised to a strong baseline (in the case where $C_t = 0$, the model's predictions equal constant velocity) and then fine-tunes predictions on training examples for which the constant velocity assumption results in errors. Training a model to instead predict displacement, ΔL_t, initialises the model to a the weaker constant position baseline, $\Delta L_t = 0$. The final predicted coordinates in the original 2D image projection, $\hat{L}_{t+m}$, are then recovered as follows:

$$\hat{L}_{t+m} = \tilde{L}_{t+m} + \hat{P}_{t+m} \tag{4.5}$$

From the feature vector extracted from optical flow, DTP uses a fully connected layer to predict the pedestrian's future location, $\hat{P}_{t+m}$, representing the estimated correction factor. A vector of large magnitude indicates that the pedestrian velocity will increase or decrease, whereas a vector of magnitude close to 0 indicates that the pedestrian will maintain their current velocity.

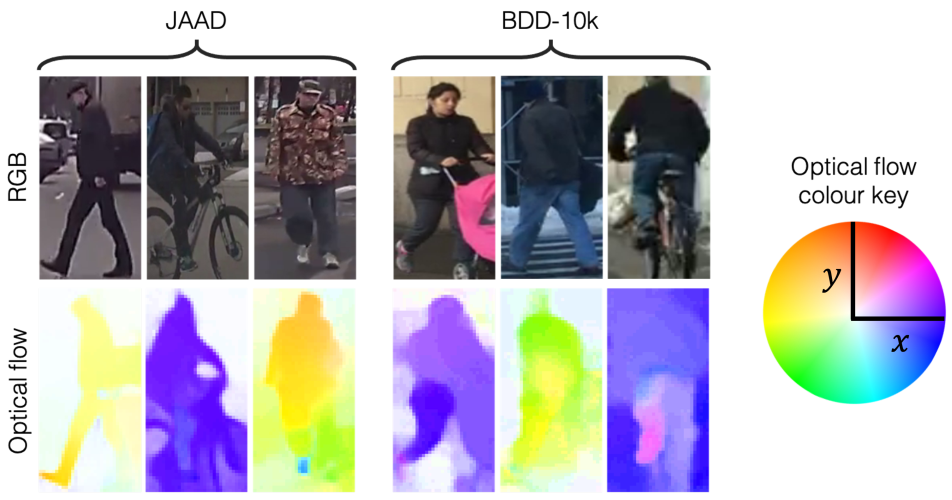


Figure 4.6: **Example pedestrians with associated optical flow.** The left 3 images are human-annotated pedestrians from the JAAD dataset, right 3 images are pedestrians detected on the BDD-100k dataset using YOLOv3. Optical flow captures motion resulting from both the camera and the pedestrian. In the lower row, we show the optical flow magnitude corresponding to the RGB image above. The colour key to the right of the figure shows the direction of pixel displacement.

We use ResNet [177] as our backbone network, owing to its consistently good performance on many vision tasks. A high-level diagram of our model is shown in Figure 4.5. Further details of the architecture modifications are outlined in the implementation section below.

Performance evaluation

We implement DTP using Pytorch [180] and evaluate results using the BDD-10k dataset introduced in Section 3.1.2 and the JAAD [64] dataset. We adopt the conventional methodology of pre-training on a large dataset before fine-tuning on a smaller target dataset, intending to improve generalizability on the target dataset [181]. Code is available online at: `https://github.com/olly-styles/Dynamic-Trajectory-Predictor`.

Implementation. Optical flow is extracted from cropped pedestrians using the provided human-annotated bounding boxes with the Flownet2-CSS algorithm [150]. Pixel displacements are clipped at ± 50 and scaled to the range $[0, 1]$. Clipping displacements removes extreme values, and is common practice in action recognition (see, for example, the work of Feichtenhofer et al.[182]). Example pedestrian optical flow images are shown in Figure 4.6 (second row). We use a stack of n horizontal and n vertical optical flow frames at timesteps $t-n$ to t. Features are computed from the $2n$ input channels using the ResNet-18 CNN architecture [177]. We modify the first convolutional layer to use $2n$ input channels rather than 3, keeping other dimensions the same. We replace the 1000-D softmax output layer with a 30-D fully connected later, producing predictions for the x and y coordinates of the 15 future bounding box centroids, corresponding to one second into the future. We use cross-modality pre-training, and partial batch normalization [179] to initialize our CNN with ImageNet weights. The model is optimized to minimize the ℓ_2 loss between the true and predicted future locations, $L_{t+1} \ldots L_{t+m}$ and $\hat{L}_{t+1} \ldots \hat{L}_{t+m}$ as follows:

$$loss = ||L_{t+1 \ldots t+m} - \hat{L}_{t+1 \ldots t+m}|| \tag{4.6}$$

We perform 5 fold cross-validation on the JAAD training set to tune hyperparameters. DTP is trained until convergence using the Adam [183] optimizer with an initial learning rate of 10^{-5}, which is reduced to 10^{-6} once performance saturates. We use a batch size of 64 and a weight decay of 10^{-2}. Each pedestrian is resized to 256×256 pixels. For data augmentation, a randomly cropped sub-image of size 224×224 is taken.

We split the JAAD dataset into training (videos 0-250) and testing (videos 251-346) sets. Once hyperparameters are fixed, we obtain an estimate of the model's generalizability by training on each of the 5 folds until performance on the respective validation set saturates. We then evaluate the model on the test set. We report the mean performance on the test set for the 5 folds.

Evaluation. We use two metrics to evaluate model performance, mean squared error (MSE) and displacement error (DE@t) at timesteps up to 15, following existing works [31, 92]. The MSE is the mean of the squared errors of the predicted centroid in pixels from all timesteps 1 to m and across all samples in the test set. The DE@t is the mean Euclidean distance in pixels of the predicted and ground truth centroid for timestep t only. Both metrics are relative to an image resolution of 1280×720.

We evaluate our proposed approach with 4 different inputs: a single RGB frame at time t, a single optical flow frame at time t, a stack of 5 optical flow frames at times $t-4$ to t, and a stack of 9 optical flow frames at times $t-8$ to t. We use 9 as our maximum value of m rather than the 10 frames commonly used for action recognition [178, 179]. As 2 consecutive frames are required to compute optical flow, using 10 RGB frames results in 9 optical flow frames, allowing us to compare fairly with Future Person Localization (FPL) [31], which uses 10 frames as input. As each optical flow frame requires two consecutive RGB frames to be computed, using 10 input frames results in 9 optical flow frames. Following prior works [31, 32] we adopt constant velocity (CV) and constant acceleration (CA) as baselines. We compute the average velocity in the image space using the previous locations and predict the future location assuming the pedestrian maintains a linear velocity for the CV baseline.

Table 4.3: Input modality comparison.

Input modality	**MSE**	**DE@5**	**DE@10**	**DE@15 (FDE)**
CA	1426	15.3	28.3	52.8
CV	1148	16.0	26.4	47.5
RGB frame	1042	11.6	24.9	45.2
Optical flow frame	873	11.1	23.0	41.2
5 optical flow frames	651	9.4	19.3	35.6
9 optical flow frames	**610**	**9.2**	**18.7**	**34.6**

Similarly, we compute the average acceleration using the change in previous velocities over the same time period as the CV baseline for the CA baseline and extrapolate these values into the future timesteps assuming linear acceleration. Using 4 previous locations to compute both velocity and acceleration resulted in the best cross-validation performance, using the cross-validation strategy as outlined in Section 4.3.1.

Results. Table 4.3 shows the performance of each model with different input modalities in comparison with the CV and CA baselines. Due to the relatively poor performance of the RGB input, we do not fuse RGB and optical flow models as in the two-stream action recognition model which inspired this work [178]. Example successful trajectory predictions of our model using a stack of 9 optical flow frames compared to baselines are shown in Figure 4.7. DTP performs particularly well in situations where a pedestrian first begins walking and when the ego-vehicle begins to turn sharply. Figure 4.8 shows failure cases. DTP performs less well under conditions of significant background motion and sudden upper body movements from the target pedestrian.

We compare our method using a stack of 9 optical flow frames with linear baselines and FPL [31] in Table 4.4. We modify FPL to output 15 timesteps, corresponding to 1 second in into the future at 15FPS rather than the 10 as in the original architecture, which was implemented for a dataset at 10 FPS. We use optical flow for ego-motion estimation as described in the original work. We also find that removing the batch normalization layers results in better performance for our dataset. Both DTP and FPL see a reduction in error with our proposed Constant Velocity (CV) correction term P_t (rather than directly predicting the location displacement ΔL_{t+m}). DTP attains the best performance.

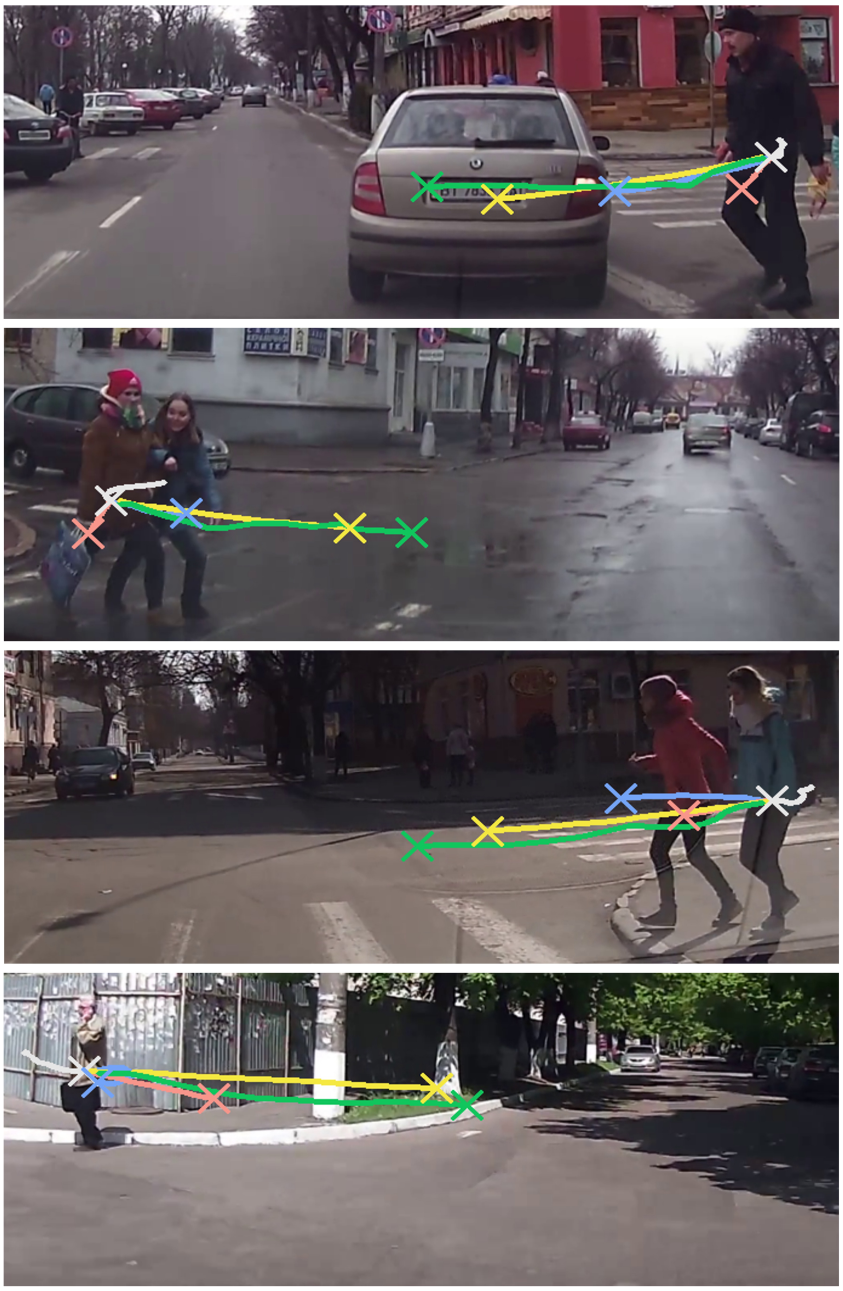


Figure 4.7: **Example successful trajectory forecasts on the JAAD test set.** Colours represent past trajectory **(White)**, ground truth future trajectory (**Green**), the constant velocity baseline (**Red**), FPL[31] (**Blue**), and DTP (Yellow). Crosses represent final location after one second. DTP is able to anticipate a pedestrian crossing before stepping into the road, and account for vehicle ego-motion. Data samples were chosen from the predictions with lowest FDE on the JAAD test set.

Machine-annotated pre-training

We pre-train DTP using our machine-annotated dataset, BDD-10K, introduced in Chapter 3. By using a large dataset for pre-training, our goal is to learn

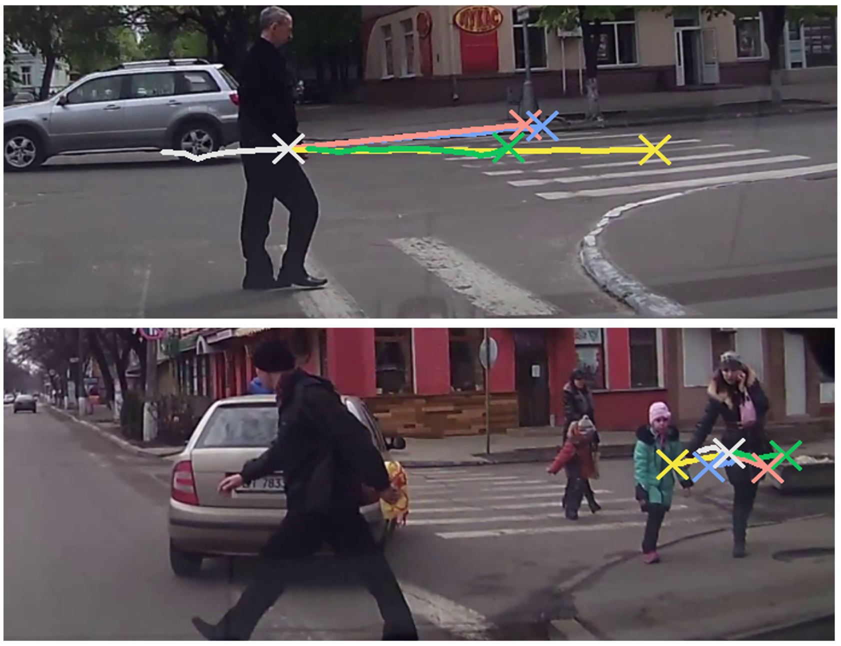


Figure 4.8: **Example unsuccessful trajectory forecasts on the JAAD test set.** Colours represent past trajectory **(White)**, ground truth future trajectory (**Green**), the constant velocity baseline (**Red**), FPL[31] (**Blue**), and DTP (**Yellow**). Crosses represent final location after one second. DTP performs poorly in situations with significant background motion, such as those due to other vehicles (upper example), or upper body motion in the counter walking direction (lower example). Data samples were chosen from the predictions with highest FDE on the JAAD test set.

model weights that will be transferable to the target dataset through transfer learning [181]. However, as BDD-10K is labelled using our machine-annotation method and JAAD is labelled using human annotators, it is not clear that features will generalize from one annotation method to the other. We evaluate this by pre-training both DTP and FPL [31] using the BDD-10K dataset and then fine-tuning models on the JAAD dataset.

Evaluation. We use an 80%-20% training-validation split for BDD-10K. We pre-train DTP and FPL on BDD-10K using the same hyperparameters as in other experiments outlined previously. Once performance on the validation set saturates, the models are then fine-tuned on the JAAD training set. We evaluate the trajectory forecasting performance with and without pre-training on BDD-10K rather than the pedestrian detection quality, owing to the lack of human-annotated bounding boxes.

Results. The impact of machine-annotated pre-training using the YOLOv3 detector before fine-tuning on the human-annotated JAAD dataset is shown in Table 4.5. We observe an improvement in MSE and FDE for both DTP and FPL when pre-trained on BDD-10K. The performance improvement may be

Table 4.4: Model comparison.

Model	**CV correction term**	**MSE**	**FDE**
FPL [31]	✗	1405	49.5
FPL [31]	✓	881	41.3
DTP	✗	1404	54.6
DTP	✓	**610**	**34.6**

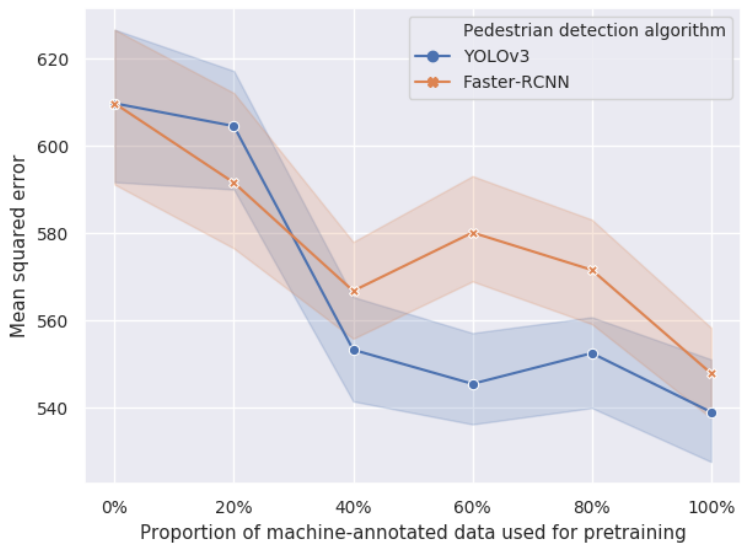


Figure 4.9: **Impact of machine-annotated dataset pre-training.** Impact of pre-training dataset size and pedestrian detection algorithm on the performance on JAAD (human-annotated) test set. Shaded areas show the 95% confidence interval. Performance does not appear saturated at 100% of data used for pre-training, suggesting that more data may improve performance further.

due to the model's ability to learn the motion patterns of under-represented classes, such as children or the elderly, from a larger dataset.

We evaluate the impact of pre-training dataset size and pedestrian detector by training DTP on subsets of BDD-10K ranging from 20% to 100% of the total dataset size. Figure 4.9 shows the MSE on the JAAD test set for both YOLOv3 and Faster-RCNN annotations using the DTP model. In general, the error on the JAAD test set reduces as larger subsets of our machine-annotated dataset is used for pre-training.

Table 4.5: **Impact of pre-training on BDD-10K with YOLOv3.** Both models perform better in terms of MSE and FDE on the human-annotated JAAD dataset after pre-training on the machine-annotated BDD-10K dataset.

Model	**Pre-training with machine annotation**	**MSE**	**FDE**
FPL [31]	✗	881	41.3
FPL [31]	✓	805	40.1
DTP	✗	610	34.6
DTP	✓	**539**	**32.7**

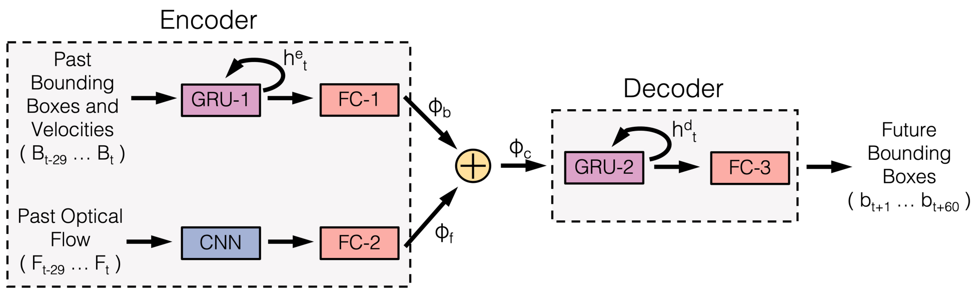

Encoder
Past Bounding Boxes and Velocities ($B_{t-29} \dots B_t$)
h^e_t
GRU-1
FC-1
ϕ_b
Past Optical Flow ($F_{t-29} \dots F_t$)
CNN
FC-2
ϕ_f
ϕ_c
Decoder
h^d_t
GRU-2
FC-3
Future Bounding Boxes ($b_{t+1} \dots b_{t+60}$)

Figure 4.10: **Spatio-Temporal Encoder-Decoder (STED).** STED consists of a GRU, a CNN, and two FC layers for feature encoding. Our decoder takes the encoded feature vector ϕ_c as input and outputs predicted object bounding boxes for the next 2 seconds using another GRU and FC layer. This architecture was chosen as recurrent units are commonly used for time series prediction problems, and the CNN also allows visual features to be extracted from within target bounding boxes.

4.3.2 Spatio-Temporal Encoder-Decoder

In this section, we present STED. STED is an encoder-decoder architecture for MOF that combines visual and temporal features. STED predicts full object bounding boxes, as opposed to trajectories alone. Similarly to DTP, STED predicts values of the centroid relative to the constant velocity baseline. Height and width are predicted relative to the current height and width.

Method

The proposed architecture has three components: (i) A bounding box feature encoder based on a Gated Recurrent Unit (GRU) [184] that extracts temporal features from past object bounding boxes (ii) A CNN-based encoder that extracts motion features directly from optical flow, and (iii) a decoder implemented as another GRU for generating future bounding box predictions given the learned features. An overview of STED is shown in Figure 4.10.

Bounding box feature encoder. Our bounding box encoder extracts features from past bounding box coordinates of each object i represented in terms of its centroid, width and height $b_t^i = (x_t, y_t, w_t, h_t)$. In addition, we compute the velocity in the x and y directions, (v_t^x, v_t^y), change in width, Δw_t, and change in height, Δh_t. This results in an 8-dimensional vector associated with each object bounding box at each timestep, $B_t^i = (x_t, y_t, w_t, h_t, v_t^x, v_t^y, \Delta w_t, \Delta h_t)$.

For each observed timestep, a GRU (GRU-1 in Figure 4.10) takes the vector B_t^i as input and outputs an updated hidden state vector h_t^e. This update is repeated for all timesteps, resulting in a single hidden state vector h_t^e at the final timestep which summarizes the entire sequence of bounding boxes. The 256-dimensional feature vector ϕ_b from a fully connected layer (FC-1 in Figure 4.10) is used as a compact representation of the history of bounding boxes.

Optical flow feature encoder. We adapt DTP to learn features directly from optical flow. Optical flow frames, F_t, are extracted from within object bounding boxes at each timestep. A stack of 10 frames is sampled uniformly from timesteps $t - 29$ to t inclusively, representing 1 second of motion history. The stack of 10 horizontal and 10 vertical frames are used as input to a CNN which takes the $20 \times 224 \times 224$ stack of frames as input and is trained to predict future object bounding boxes. The 2048-dimensional feature vector ϕ_f from the final fully connected layer (FC-2 in Figure 4.10) is used as a compact representation of optical flow features. As optical flow captures both object motion and ego-motion, the vector ϕ_f encodes information from these two motion sources. Using optical flow as the input of our encoder rather than features from a human pose estimation model [31] avoids the challenges relating to obtaining accurate pose estimations, such as body part occlusion, self occlusions, and clothing variations [47]. However, similarly to human pose estimation, optical flow models are often computationally demanding, although some faster models are available [150].

Decoder. Following the feature encoding stage, we use another GRU to generate the estimated sequence of future bounding boxes, enabling the model to generate predictions for an arbitrary number of timesteps into the future. The two feature vectors, ϕ_f and ϕ_b, are concatenated resulting in a single feature vector ϕ_c representing both optical flow and bounding box history. For each future timestep to be predicted, the decoder GRU (GRU-2 in Figure 4.10) receives two inputs: The concatenated feature vector ϕ_c, and the internal hidden state h_{t-1}^d. The GRU outputs a new value for h_t^d at each timestep. Given each generated hidden state, a final fully connected layer generates the predicted bounding box for each timestep. Rather than representing object bounding boxes by their absolute location [83], or relative displacement from the previous bounding box [31], we adopt the same formulation as DTP and

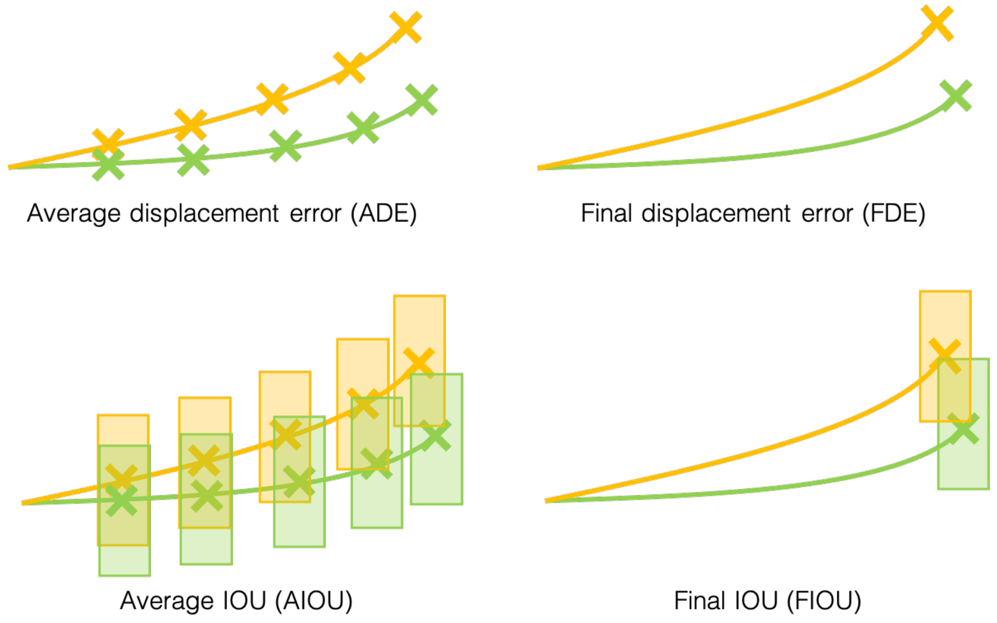

Average displacement error (ADE)
Final displacement error (FDE)
Average IOU (AIOU)
Final IOU (FIOU)

Figure 4.11: **MOF metrics**. We use the average and final displacement error metrics which are used throughout the trajectory forecasting literature. To evaluate the performance of the bounding box predictions (in addition to centroids) we also introduce the average and final IOU metrics.

represent the bounding box centroid as the relative change in velocity. The decoder generates a vector $(\Delta v^x, \Delta v^y, \Delta w, \Delta h)$, representing the change in velocity along the x and y-axes, and the change in bounding box width and height. The untrained model is initialized to the case where $\Delta v^x = \Delta v^y = 0$ (constant velocity) and $\Delta w = \Delta h = 0$ (constant scale). This formulation results in a better initialization than absolute or relative locations.

Experimental results

We use 4 metrics for evaluating models on the MOF task. We adopt the Average Displacement Error (ADE) and Final Displacement Error (FDE) metrics from the trajectory forecasting literature [32]. ADE is defined as the mean Euclidean distance between predicted and ground-truth bounding box centroids for all predicted bounding boxes, and FDE is defined similarly for the centroid at the final timestep only. We also use the Average Intersection-Over-Union (AIOU) and Final Intersection-Over-Union (FIOU). AIOU is defined as the mean IOU of the predicted and ground truth bounding boxes for all predicted boxes, and FIOU is the IOU for the box at the final timestep only. A visual representation of these 4 metrics is shown in Figure 4.11.

We adapt the following models for MOF, which were initially developed for trajectory forecasting. Each model is modified for full bounding box prediction assuming object scale is constant or by adding additional output channels representing bounding box height and width for the learning-based approaches.

Constant Velocity & Constant Scale (CV-CS): We adopt the simple constant velocity model, which is used widely as a baseline for trajectory forecasting models [31, 32, 84] and as a motion model for MOT [185–187]. We take the difference in start position and end position of the object averaged over the previous 5 frames to compute velocity. We find that using a constant scale performs better than linearly extrapolating a change in width and height, which may be due locomotion causing changes in bounding box width.

Linear Kalman Filter (LKF) [74]: The LKF is a widely-used method for tracking objects and predicting trajectories under noisy conditions. We use an LKF with initial parameters chosen using cross-validation and the last updated motion value for forecasting. The LKF is one of the most popular motion models for MOT [122, 188, 189].

Future Person Localization (FPL) [31]: We adapt FPL, which uses pedestrian pose extracted using OpenPose [29] and ego-motion estimation using optical flow extracted with FlowNet2 [150]. FPL predicts 10 output timesteps, but Citywalks has a 2-second prediction horizon, totalling 60 timesteps. Therefore, we train the model to predict a location for every 6th frame and linear interpolate between predictions to get predictions for 60 timesteps. We evaluate one version with just centroid predictions and another with additional output channels for the height and width of the target bounding box.

Dynamic Trajectory Predictor (DTP) [1]: We adapt DTP, which uses a CNN with past optical flow frames as input to predict future bounding boxes. Similarly to FPL, we evaluate one version with just centroid predictions and another with additional output channels for the height and width of the target bounding box.

Clips from Citywalks are split into 3 folds, and the test set is further divided 50% for validation and 50% for testing for each fold. We use inter-city cross-validation, i.e., footage from cities in the validation/testing sets *do not* appear in the training set. This challenging evaluation setup ensures that pedestrian identities from the training set do not appear at test time, and prevents models from overfitting to a particular environment.

Bounding box feature encoder. Bounding box vectors B_t^i are computed by taking the velocity of the object over the previous 5 timesteps, i.e., $v_t^x = x_t - x_{t-4}$ and $v_t^y = y_t - y_{t-4}$. Our feature encoder consists of a GRU with 512 hidden units which uses B_{t-1}^i and the previous hidden state vector h_{t-1}^e as input and outputs an updated hidden state vector h_t^e. We use GRUs rather than LSTMs as recurrent units in STED as we find the performance is similar while GRUs is less computationally demanding.

Table 4.6: Results averaged over 3 train-test splits on Citywalks with our two annotation sets using YOLOv3. DTP and FPL predict object centroids only, so IOU metrics are not applicable.

Model	ADE (↓)	FDE (↓)	AIOU (↑)	FIOU (↑)
CV-CS	32.9	60.5	51.4	26.7
LKF [74]	34.3	62.1	49.1	25.5
DTP [1]	28.7	52.4	–	–
FPL [31]	30.2	53.4	–	–
DTP-MOF	29.0	52.2	54.6	30.8
FPL-MOF	31.6	55.7	53.0	30.9
STED	**27.4**	**49.8**	**56.8**	**32.9**

Optical flow feature encoder. We compute optical flow for each video frame using FlowNet2 [150]. The flow from within each pedestrian bounding box is then cropped, clipped to a range of -50 to 50, scaled to a fixed size of 256×256, and normalized to a range of 0 to 1. We perform standard data augmentation, taking a random crop of size 224×224 from the stack of optical flow frames. We train the optical flow feature encoder using ResNet50 [177] as the backbone CNN architecture for 10k iterations with a batch size of 64 and learning rate of 1×10^{-5} to predict future object locations and then freeze the weights to use our optical flow encoder as a fixed feature extractor.

Decoder. Our decoder takes the concatenated feature vector ϕ_c as input. The decoder consists of another GRU with 512 hidden units. For each of the 60 timesteps to be predicted, the decoder takes ϕ_c and previous hidden state h^d_{t-1} and outputs a new hidden state h^d_t. A linear layer takes the hidden state and generates a predicted bounding box for the respective timestep. The optical flow feature encoder is used as a fixed feature extractor, while the bounding box encoder and decoder are trained jointly end-to-end using an initial learning rate of 1×10^{-3}, which is halved every 5 epochs. We use a batch size of 1024 and train the model for 20 epochs. The model is optimized using the smooth ℓ_1 loss, defined as follows:

$$\text{loss}(y, \hat{y}) = \begin{cases} 0.5(y-\hat{y})^2, & \text{if } |y-\hat{y}| < 1 \\ |y-\hat{y}| - 0.5, & \text{otherwise} \end{cases} \tag{4.7}$$

We find the smooth ℓ_1 loss to be more robust to outliers in the training data than the ℓ_2 loss.

Table 4.7: Results averaged over 3 train-test splits on Citywalks with our two annotation sets using Mask-RCNN. DTP and FPL predict object centroids only, so IOU metrics are not applicable.

Model	ADE (↓)	FDE (↓)	AIOU (↑)	FIOU (↑)
CV-CS	31.6	57.6	46.0	21.3
LKF [74]	32.9	59.0	43.9	20.1
DTP [1]	26.7	48.5	–	–
FPL [31]	28.6	49.8	–	–
DTP-MOF	27.3	49.2	49.6	25.1
FPL-MOF	29.3	51.0	44.9	22.6
STED	**26.0**	**46.9**	**51.8**	**27.5**

Table 4.8: Ablation study evaluating the bounding box (BB), optical flow (OF) encoders separately. Results are the mean of both annotation sets.

Model	ADE / FDE (↓)	AIOU / FIOU (↑)
BB-encoder	29.6 / 53.2	51.5 / 27.9
OF-encoder	27.5 / 50.0	53.2 / 28.8
Both encoders	**26.7 / 48.4**	**54.3 / 30.2**

4.3.3 Results

We evaluate each model on the Citywalks dataset using both the YOLOv3 and Mask-RCNN annotation sets and evaluate each component of STED separately. Finally, we evaluate the cross-dataset generalizability of each model on the MOT-17 dataset [46].

Results on Citywalks. Table 4.6 shows the ADE / FDE[1] and AIOU / FIOU of all methods on Citywalks using the YOLOv3 annotation set, and Table 4.7 shows the same results using the Mask-RCNN annotation set. We evaluate the original DTP and FPL models for trajectory forecasting, as well as the versions modified for MOF. STED consistently performs better than existing approaches across all metrics, resulting in more precise bounding box forecasts. Figure 4.12 shows example bounding box predictions. STED implicitly anticipates both object and ego-motion in a diverse range of environments and situations. Figure 4.13 shows failure cases. The model performs poorly in challenging conditions such as large ego-motions and when the pedestrian scale is small.

[1]A displacement error of 50 pixels corresponds to 2.5% of the total frame size at a resolution of 1280 × 720.

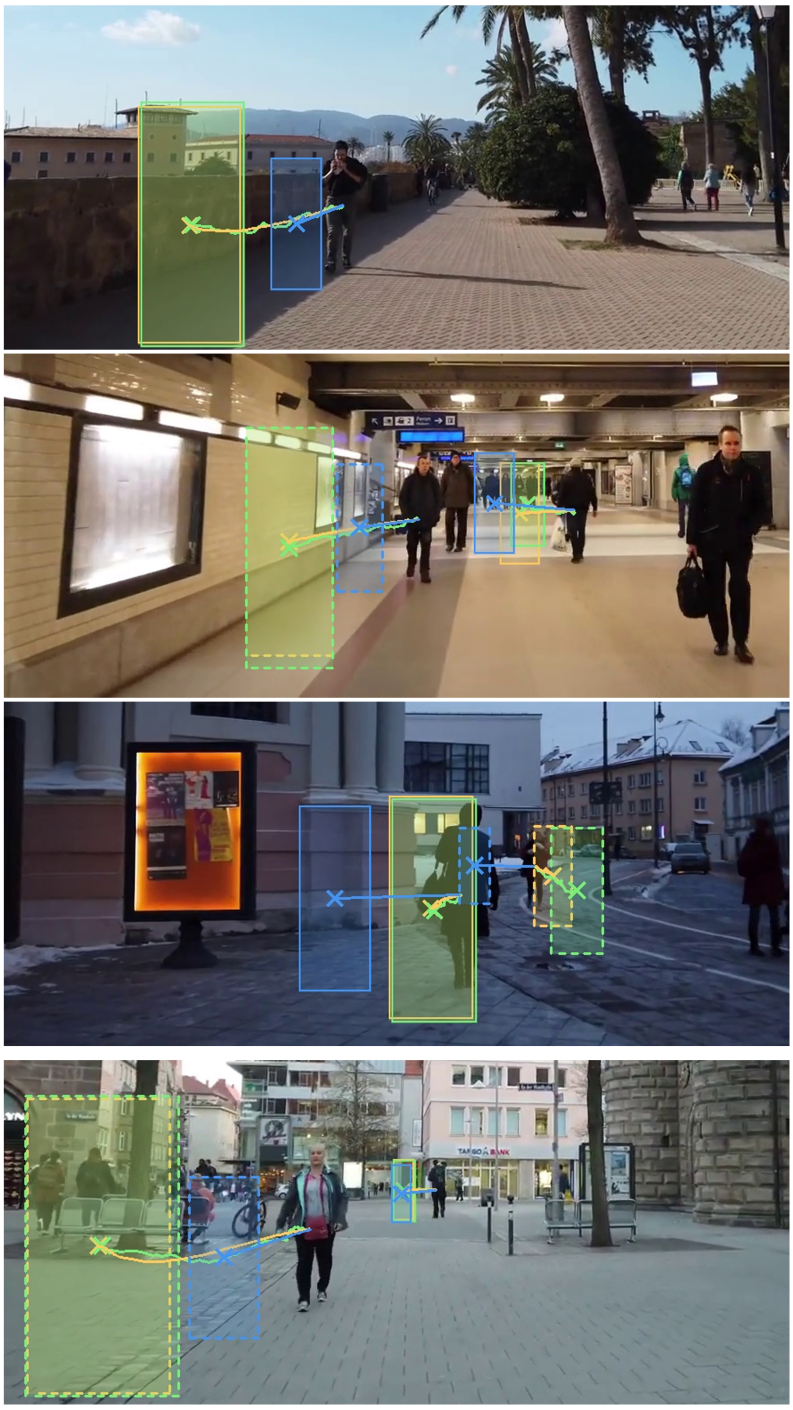


Figure 4.12: **Example successful object forecasts using STED.** Colours represent ground truth (**Green**), constant velocity and scale (**Blue**), and STED (**Yellow**). Forecasts are made for each of 60 timesteps in the future for all pedestrians in the scene, but here we visualize the predicted bounding box at $t = 60$ only and at most two pedestrians per frame for clarity. Line type (dashed/solid) denotes unique pedestrians. Data samples were chosen from the predictions with highest FIOU on the Citywalks test set. More examples are available at: `https://youtu.be/GPdNKE6fq6U`

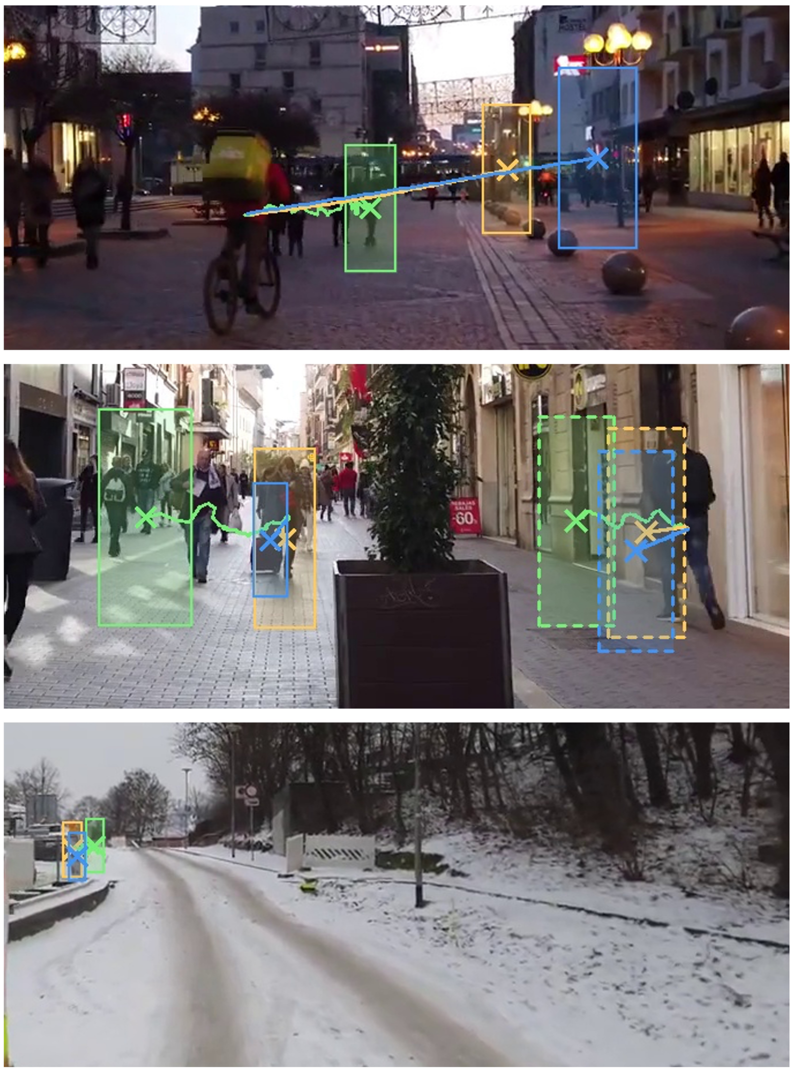
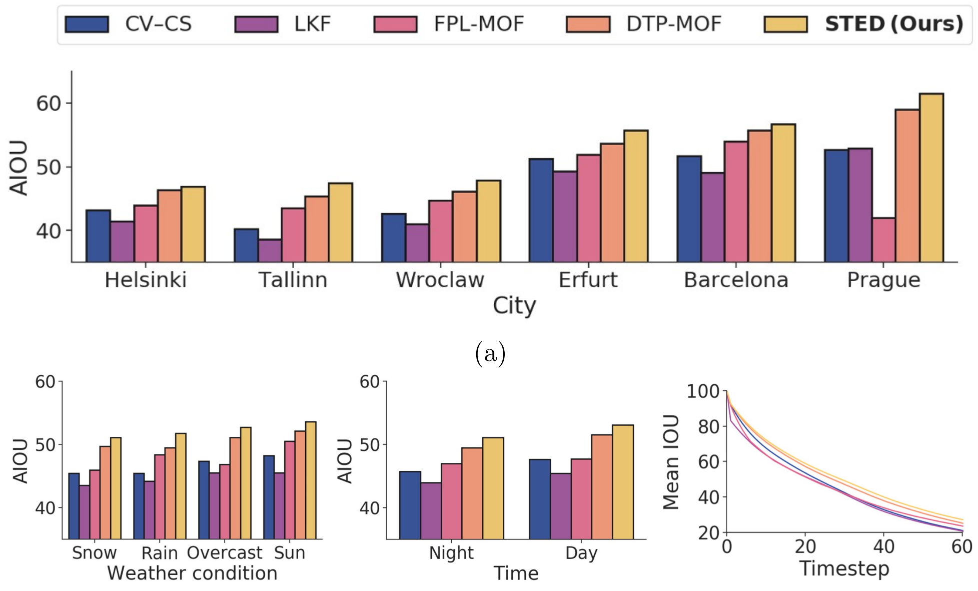


Figure 4.13: **Example unsuccessful object forecasts using STED**. Colours represent ground truth (**Green**), constant velocity and scale (**Blue**), and STED (**Yellow**). The examples highlight the difficulty of the Citywalks dataset, which contains several distant pedestrians and motions that are not well predicted by constant velocity. Data samples were chosen from the predictions with lowest FIOU on the Citywalks test set

We further break down performance on Citywalks in Figure 4.14. We find that most models perform better for sequences recorded in cities with clear weather conditions (e.g., Barcelona, Prague) than, in particular, snow (e.g., Tallinn, Helsinki). To confirm this intuition, we further plot the performance in different weather conditions and at different times of the day. Larger prediction errors in rainy and snowy conditions may be due to the additional challenge of extracting features from optical flow under poor weather conditions or less predictable movements from the targets negotiating snow and puddles. Finally, we plot the mean IOU at all predicted timesteps 1 to 60. The IOU of



(b) (c) (d)

Figure 4.14: **Performance analysis on Citywalks.** Here, we report performance on both validation and test sets for all 3 folds to cover the entire dataset. Performance is broken down by (a) top 3 and bottom 3 cities by AIOU, (b) weather condition, (c) time of day and (d) future timestep.

the predicted and ground-truth bounding boxes predictably declines quickly, particularly for earlier timesteps. STED maintains the best IOU throughout the entire prediction horizon.

Ablation study. We evaluate the benefits of each feature stream of our proposed model by evaluating them separately. Specifically, we use the bounding box encoder feature vector ϕ_b as input to the decoder, rather than the concatenated feature vector ϕ_c. We repeat this for the optical flow encoder feature vector ϕ_f. By not concatenating the two vectors, the output feature vector from the encoder is halved in size. Therefore, we also half the size of the decoder GRU-2 hidden state. Table 4.8 shows the results of our ablation study on Citywalks. Both the bounding box and optical flow encoders contribute to the overall performance and therefore work in a complementary fashion. The optical flow encoder performs better than the bounding box encoder in isolation. The optical flow is considerably higher dimensional ($20 \times 224 \times 224$) than the bounding box (10×8) inputs.

Computational complexity. The most computationally expensive component of STED is computing optical flow. Our implementation uses FlowNet2, which requires 123ms to compute on an Nvidia GTX 1080 GPU [150]. This model may be replaced by more efficient methods, although we found the quality of optical flow to impact overall performance. Additional components, such as the CNN architecture or the number of hidden units in the GRUs may

Table 4.9: Results on MOT-17 after training on fold 3 of Citywalks. Models are not fine-tuned on MOT-17.

Model	ADE / FDE (↓)	AIOU / FIOU (↑)
CV-CS	58.9 / 104.7	43.8 / 21.5
LKF [74]	62.0 / 110.2	41.6 / 20.1
FPL [31]	56.9 / 96.3	–
DTP [1]	55.2 / 99.0	–
FPL-MOF	58.0 / 98.4	41.4 / 20.4
DTP-MOF	52.2 / 92.4	**47.7** / **26.1**
STED	**51.8** / **91.6**	46.7 / 24.4

be modified if real-time performance is required, at some cost in forecasting accuracy.

Cross-dataset evaluation. In order to evaluate the generalizability of models trained on Citywalks, we use the popular MOT-17 dataset [46]. We use sequences 2, 9, 10, and 11 from the MOT-17 train set and discard sequences 4 and 13 as these sequences are filmed from an overhead perspective. We also discard sequence 5 due to the low image resolution and frame rate. We follow a similar pre-processing setup to Citywalks, discarding tracks shorter than 3 seconds. We also ensure pedestrians are occluded no more than 50% of their total bounding box size using the annotations provided, resulting in 83 unique pedestrian tracks. We take each model trained on Citywalks and evaluate using each of the four sequences. Note that we do not modify the models, and crucially we *do not* fine-tune on MOT-17. Table 4.9 shows encouraging results suggesting that models trained on Citywalks generalize cross-dataset and to human-annotated bounding boxes. However, due to the small size of the MOT-17 dataset, these results should be treated with caution.

4.4 Discussion

The methods introduced in this chapter have demonstrated the benefits of using dense optical flow as an input modality for egocentric trajectory forecasting models. Modelling motion information in this way facilitates the learning of cues that indicate a person's intentions. For example, several of the examples shown in Figure 4.7 demonstrate that DTP can anticipate a pedestrian crossing the street before they step into the road. Optical flow is also a useful input modality for our other model, STED, and our ablation study demonstrates that both optical flow and bounding box encoders are complementary.

Training models to predict the target's deviation from constant velocity baseline using our proposed constant velocity correction term rather than directly predicting their trajectory relative to their current location improves the performance of the models we tested. As several existing works such as FPL [31] predict trajectories relative to an object's current location, we suggest that future works consider using the deviation from constant velocity as a training target instead. Our promising results suggest that initializing to a stronger baseline in this way improves trajectory forecasting accuracy.

The performance of trajectory forecasting models improves as more machine-annotated data is used for pre-training. Furthermore, models trained using the machine-annotated Citywalks dataset outperform the constant-velocity constant-scale baseline on the MOT-17 dataset *without fine-tuning*. This result is encouraging, particularly in settings where human-labelled training data is scarce, as is the case for egocentric trajectory forecasting and MOF. The benefits of pre-training models on machine-annotated datasets to improve their performance on a human-annotated dataset is also studied in the follow-up work of Ansari et al. [155], which will be discussed in more detail in Chapter 6.

4.5 Summary

In this chapter, we have formally defined the egocentric SCTF problem and MOF problems and proposed a model for each problem formulation. DTP outperforms the existing state-of-the-art using the JAAD dataset. Our experiments results demonstrate that our proposed machine-annotation pre-training strategy improves the performance of both DTP and an existing state-of-the-art model, FPL [31]. Furthermore, we show that our proposed CV correction term also improves the performance of both DTP and FPL.

We also introduced STED, a two-stream encoder-decoder model for MOF. STED predicts object scale in addition to trajectory and outperforms existing state-of-the-art using the Citywalks dataset. Our ablation study demonstrates that the bounding box and optical flow streams are complementary and perform better than either stream in isolation. Further performance breakdown also reveals that STED and other comparable models generally perform better in fair weather conditions (sun or overcast) compared to poor weather conditions (snow or rain) and perform better for video sequences shot during the day than those shot after dark.

Our proposed models predict the trajectory of an object as viewed from a single camera view, which may be mounted on an intelligent vehicle or other kinds of robotic systems that share space with pedestrians. Using an egocentric perspective facilitates extracting features from optical flow in addition to trajectory features from the sequence of historical locations. Both models are trained using datasets generated using our machine-annotation procedure proposed in Chapter 3 and generalize to human-annotated datasets.

In the next chapter, we extend these methods by asking the following question: How can we forecast trajectories in a multi-camera environment?

Chapter 5

Multi-Camera Trajectory Forecasting

The methods introduced in Chapter 4 tackle the task of single-camera trajectory forecasting. A critical drawback of the single-camera settings is that models cannot anticipate when new objects will enter the scene [17] as they are limited to the data from a single camera. Furthermore, SCTF methods are only suitable for short-term trajectory forecasting, typically 1 to 5 seconds [2, 31–33, 84, 190], due to the limited field-of-view. The constraint of a single camera viewpoint must be removed to overcome these issues. To this end, we introduce Multi-Camera Trajectory Forecasting (MCTF) - a new framework within trajectory forecasting. Given the information about an object's location in one or more camera views, we want to predict its future location across a camera network in all possible camera views. This idea is shown visually in Figure 5.1. The MCTF formulation introduces new challenges when compared to SCTF, such as identifying which camera a person will re-appear and when they will re-appear. Throughout this chapter, we consider predicting the trajectory of humans, although MCTF can easily be generalised to any moving object, such as vehicles or animals.

Given an object tracklet in one or more camera view(s), our MCTF framework comprises the following three tasks: (i) In *which* cameras will the object appear next? (ii) *When* will the object appear in those cameras? (iii) *Where* will the object appear in the identified camera views? Owing to the wide body of complementary literature on pedestrian detection [23] and person Re-ID [124], we focus on pedestrians for our MCTF task. Nevertheless, our proposed data representation and model can be easily generalized to any moving object.

We present a deep encoder-decoder approach to MCTF that introduces the idea of *trajectory tensors* - a new technique to encode trajectories across multiple camera views and the associated uncertainty in future locations. Trajectory tensors are an attractive alternative to the coordinate-based trajectory, which

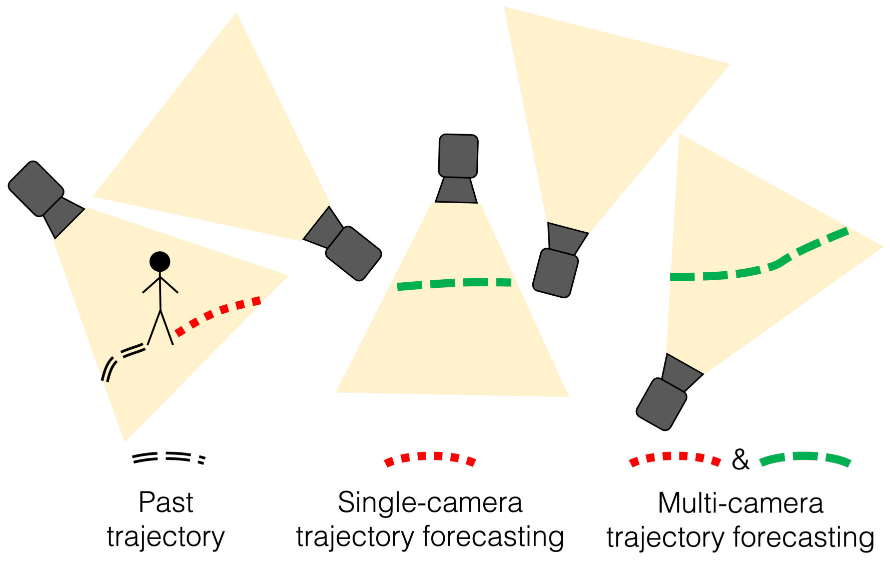

Past trajectory
Single-camera trajectory forecasting
&
Multi-camera trajectory forecasting

Figure 5.1: **Multi-Camera Trajectory Forecasting**. We introduce a novel formulation of the trajectory forecasting task which considers multiple camera views.

is the de facto representation in existing trajectory forecasting works [2, 31–33, 84, 190], with only a few recent exceptions [87]. Coordinate trajectories represent the historical and predicted future locations of an object as a sequence of coordinates. The representation considers coordinates in the image space in a single-camera view or world coordinates if a projection is available. In contrast to the coordinate approach, proposed trajectory tensors divide viewpoints from several cameras into grid cells with values indicating an object's presence or absence. This representation enables us to intuitively model diverse future locations, associated uncertainty, and object locations in an arbitrary number of viewpoints. Trajectory tensors also offer easy to interpret results that can be visualised easily.

The rest of this chapter is organised as follows. In Section 5.1, we introduce Multi-Camera Trajectory Forecasting and the motivations behind studying the problem. In Section 5.2, we formally introduce the problem formulations within our proposed MCTF framework. We introduce our proposed approach in Section 5.3, and evaluate model performance in Section 5.4. The implications of our work are discussed in Section 5.5. The chapter is summarised in Section 5.6.

5.1 Motivation

A typical automated multi-camera surveillance system consists of detection, tracking, and RE-ID components to monitor objects-of-interest [130, 132, 133]. Scaling such systems to large camera networks can be challenging due to computational demands, as each component must run on each camera. Jain et al. [130] study the impact of scaling multi-camera tracking to large networks and show that filtering the search space to high traffic areas can considerably reduce the search space at only a small cost in person Re-ID recall. In addition to using the traffic level in each area, trajectory information has also been used for both RE-ID [129] and multi-camera tracking [131]. Using trajectories for these tasks can supplement existing appearance-based models by providing a second source of information for matching objects across camera views. However, previous works [129–133] reactively use trajectory information, i.e., the object must have already been observed in multiple camera views. Processing videos at a lower image resolution or frame rate may reduce the computational demands, but this often results in missed detections [191]. Alternatively, we may choose to monitor only a subset of the cameras in a network, resulting in missed detections. A successful MCTF model can mitigate this issue by preempting an object's location in a distributed camera network, allowing the system to monitor fewer cameras through an intelligent selection technique. This is distinct from previous works that use trajectory information for person Re-ID [129] or vehicle tracking [131] which are *reactive* to observations *after* an object has been observed in multiple camera views. Our proposed MCTF framework is *proactive* - it predicts the future location of an object *before* it enters the camera view.

Multiple overlapping camera views provide an additional information source to machine learning models. This is beneficial in tasks such as human activity recognition [192], where multiple camera viewpoints make recognising a particular activity easier. This result is intuitive, as different perspectives may provide different information that helps to recognise the action easier. We hypothesise that multiple overlapping camera viewpoints may also be helpful for trajectory forecasting, as additional viewpoints act as a source of redundancy. This idea is explored later in the chapter.

5.2 The Multi-Camera Trajectory Forecasting Framework

Consider a typical multi-camera surveillance setup where a set of cameras $\mathcal{C} = \{c_i\}_{i=1}^{|\mathcal{C}|}$ are mounted overhead monitoring objects of interest. Given an object's bounding boxes from the previous n timesteps up to the current timestep t, $b_{(t-n:t)} = \{b_{(t-n)}, \cdots b_{(t-1)}, b_{(t)}\}$ in one or more camera views, MCTF is the task of predicting the objects future location within the area covered by the camera network. We propose a hierarchy of MCTF problem formulations at three levels.

In *which* camera(s) will the object appear in the future? Given $b_{(t-n:t)}$, our task is to identify a subset of $\mathcal{C}$ in which the object may appear at any future timestep, up to a maximum of m timesteps. We cast this as a multi-class multi-label classification problem. Our goal is to estimate the probability of appearance of the object $P_a(c_i|b_{(t-n:t)})$ for each camera $c_i \in \mathcal{C}$ where a positive class is a camera in which the object re-appears. The output is a vector $[P_a(c_1|b_{(t-n:t)}), \dots, P_a(c_k|b_{(t-n:t)})]$ of length $|\mathcal{C}|$.

***When* will the object appear?** The task here is to predict when the object will appear within the next m timesteps in a given camera. Similar to the *Which* problem, we also formulate this as a multi-class multi-label problem, where we compute the joint probability of appearance of the object $P_a(c_i, t_j|b_{(t-n:t)})$ for each camera $c_i \in \mathcal{C}$ at each timestep t_j with $j = 1, \cdots, m$. The output is a matrix $\left[\begin{smallmatrix} P_a(c_1,t_1|b_{(t-n:t)}) & \cdots \\ \cdots & P_a(c_{|\mathcal{C}|},t_m|b_{(t-n:t)}) \end{smallmatrix}\right]$ of dimension $k \times m$.

***Where* will the object appear?** This task aims at spatially localizing the object within a camera view in addition to *which* and *when*. To this end, we divide each camera view into a $w \times h$ grid and predict a probability of appearance score $P_a(c_i, t_j, g_{xy}|b_{(t-n:t)})$ for each grid cell g_{xy}, where $x = 1, \cdots, w$ and $y = 1, \cdots, h$. The output is a tensor $\mathbf{Z}$ of dimension $|\mathcal{C}| \times m \times w \times h$ containing the probability of appearance scores $P_a(c_i, t_j, g_{xy}|b_{(t-n:t)})$ for each camera $c_i \in \mathcal{C}$, timestep t_j with $j = 1, \cdots, m$, and grid cell g_{xy} with $x = 1, \cdots, w$ and $y = 1, \cdots, h$.

A visual representation of these three problem definitions is shown in Figure 5.2. It is important to note that we focus on modelling target locations in a multi-camera setting where accurate geometry information is not available. We predict the location of a single target unless otherwise specified.

5.3 Proposed Approach

In this section, we introduce our proposed data representation technique, models, and evaluation strategy for MCTF.

5.3.1 Trajectory Tensors

Existing works represent trajectories using a coordinate approach [32, 33, 84] for SCTF. Coordinate trajectories are $(x, y)_t$ vectors in the image or world space, representing the location of an object at a particular timestep t. Sometimes, the width (w) and height (h) of the object may be also included in the representation, i.e., $(x, y, w, h)_t$ [2, 61]. The representation considers coordinates in the image space in a single-camera view or world coordinates if a projection is available.

There are three critical drawbacks to a coordinate trajectory representation: (i) Standard coordinate-based approaches generally do not define a null tra-

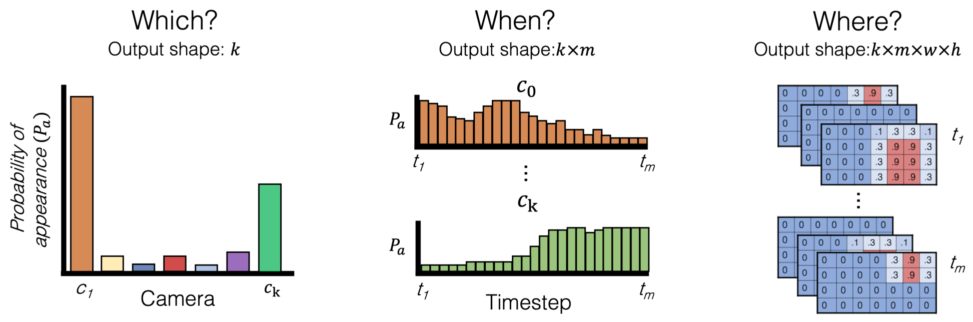

Which?
Output shape: k
Probability of appearance (P_a)
c_1
Camera
c_k
When?
Output shape: $k \times m$
c_0
P_a
t_1
t_m
c_k
Timestep
Where?
Output shape: $k \times m \times w \times h$
t_1
t_m

Figure 5.2: **MCTF problem formulations.** We formulate the MCTF framework at three levels of granularity, where each level builds upon the previous. The first level of the hierarchy predicts in *which* camera an object will appear. At the second level, in addition to *which* camera, we predict *when* the object will appear. At the final level, we predict *where* within each camera view the object will be located. We predict the probability of appearance (P_a) at each level. Owing to the impracticality of forecasting trajectories across multiple camera views with a long time horizon at a pixel level, we predict trajectories in a grid cell representation, where a grid cell represents multiple pixels.

jectory. This representation can be problematic in real-world scenarios where object coordinates can become unavailable, such as when the object is occluded, or the detection algorithm fails. It is particularly problematic in a multi-camera scenario where objects are not visible in all camera views simultaneously.
(ii) Coordinate trajectories can only represent a trajectory as viewed from a single camera. Due to the lack of representation for null trajectories, coordinates cannot be easily generalised to multiple cameras unless all objects are simultaneously visible in all cameras in $\mathcal{C}$, or separate models are created for each camera. Trajectories can instead be mapped to the world coordinate space to overcome this issue. However, this mapping requires accurate measurements of the camera locations and intrinsic parameters, which are not always available. This work assumes that such information is not available.
(iii) Finally, coordinate trajectories do not inherently represent uncertainty, which is intrinsic to the trajectory forecasting task. The space of future trajectories is multi-modal, e.g., an object can travel either left or right at a junction. Existing works address this issue by using generative models to simulate multiple futures [84, 87], from which a probability distribution can be created. Our approach, in contrast, intuitively models uncertainty in one shot.

To overcome the shortcomings of coordinate trajectory representation, we introduce the idea of *trajectory tensors* - a novel technique for compact representation of multi-camera trajectories. As shown in Figure 5.3, trajectory tensors are constructed in four steps:
(i) We consider a set of cameras $\mathcal{C} = \{c_i\}_{i=1}$, where an object of interest may appear in any number of cameras in $\mathcal{C}$.

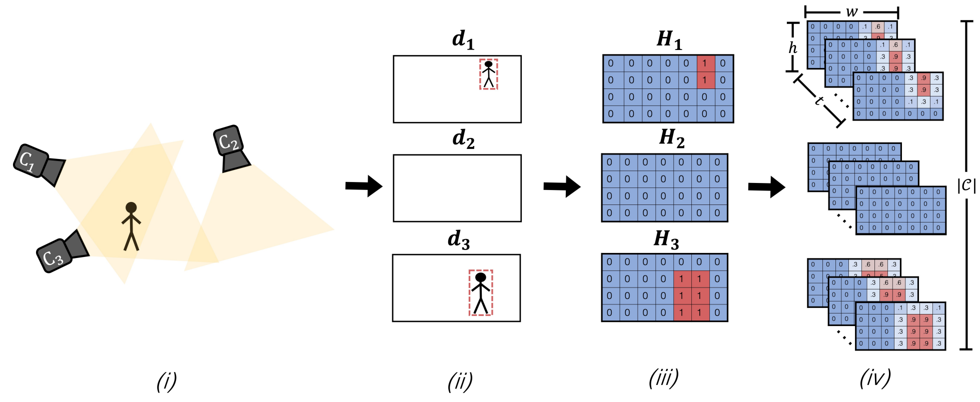


Figure 5.3: **Trajectory tensors.** Our proposed trajectory tensors are an intuitive data representation capable of representing object trajectories in multiple camera views, null trajectories, and associated uncertainty.

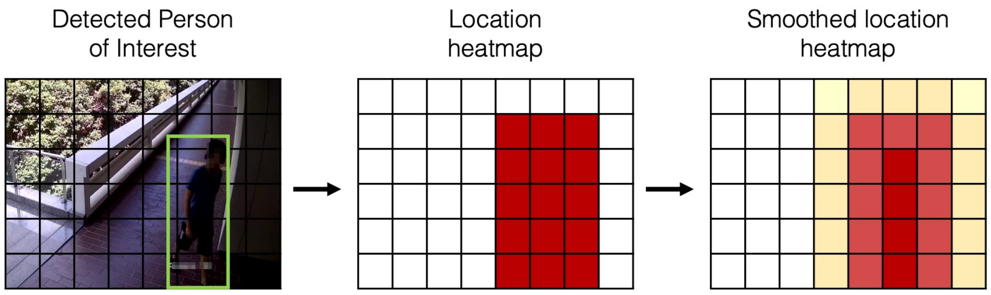


Figure 5.4: **Single frame heatmap.** We convert a bounding box detection to a heatmap. Grids cells containing the target bounding box are set to 1, and other cells to 0. The heatmap is then smoothed using a Gaussian kernel with a σ value between 0 and 4.

(ii) Each camera c_i has an associated detection $\mathbf{d_i}$ representing the object bounding box, if present.
(iii) We convert each $\boldsymbol{d_i}$ into a heatmap $\boldsymbol{H_i}$ which is an alternative representation of the object's location. This representation for a single frame is shown in Figure 5.4. A heatmap is a matrix of size $w \times h$, where each entry is a binary value indicating the presence or absence of the object in this grid cell. A single object may span any number of grid cells, depending on its size.
(iv) Finally, the heatmaps $\boldsymbol{H_i}$ for each of the cameras in $\mathcal{C}$ are stacked along the camera dimension, and computed for t timesteps. We also smooth each heatmap using a Gaussian kernel, which we find has a regularising effect. The result is a trajectory tensor, $\mathbf{Z}$, of shape $|\mathcal{C}| \times t \times w \times h$, where each entry is a value between 0 and 1. $\mathbf{Z}$ represents the trajectory of an object in multiple camera views simultaneously and can be used to represent both past (inputs) and future (predicted) object locations. Encoding object locations as trajectory tensors allows us to represent objects in multiple cameras elegantly. Trajectory tensors enable the representation of null trajectories rather than discarding this

data or using placeholder values. If a trajectory in not present in a particular camera in a particular timestep, the values in the corridponding grid will consist

Trajectory tensors share similarities with the grid-cell representation proposed recently by Liang et al. [87]; however, our proposed encoding includes multiple camera views. Besides, objects represented using trajectory tensors may span multiple grid cells, which allows us to account for variability in object scale. Object scale is not considered in most SCTF works [32, 33, 84, 87], but is a critical component in our framework.

5.3.2 Models

Existing trajectory forecasting methods such as Social-LSTM [32], Social-GAN [84], and SoPhie [33] are designed for SCTF using datasets with birds-eye view cameras [76, 96] that do not include object scale. Although some previous methods proposed [2, 31] forecast object scale in addition to location, they are also designed for SCTF; hence direct comparison for MCTF is not possible. To compare with these existing works, we adapt the methods to our MCTF framework. These models, along with new approaches based on trajectory tensors, are summarised in Figure 5.5. Each neural network-based model is comprised of a combination of fully-connected, recurrent, and convolutional layers. Inspired by fully-convolutional networks for semantic segmentation [193], we use transposed convolutional layers when predicting trajectory tensors as targets. Due to the large number of neural network models introduced in this section, we introduce each briefly and provide full details of the architectures needed to reproduce our results in Appendix B.

Coordinate trajectory approaches

Here we present our coordinate trajectory approaches which are shown in Figure 5.5 (a) and (b). Due to the lack of data representations for coordinate trajectories in multiple camera views, our coordinate trajectory approaches use a separate model for each camera, resulting in k models.

GRU. Recurrent networks have been a prevalent approach for trajectory forecasting in a single camera. We use an adapted version of STED, introduced in Chapter 4. The model proposed for SCTF uses two encoders, one for bounding box coordinates, and another which uses a Convolutional Neural Network (CNN) to extract motion information from optical flow. We use only the bounding box encoder for a fair comparison as other methods do not use visual features. For the *which* task, we use a fully-connected classification layer. For the *when* task, we use another GRU as a decoder with 128 hidden units followed by a fully connected classification layer. For the *where* task, the

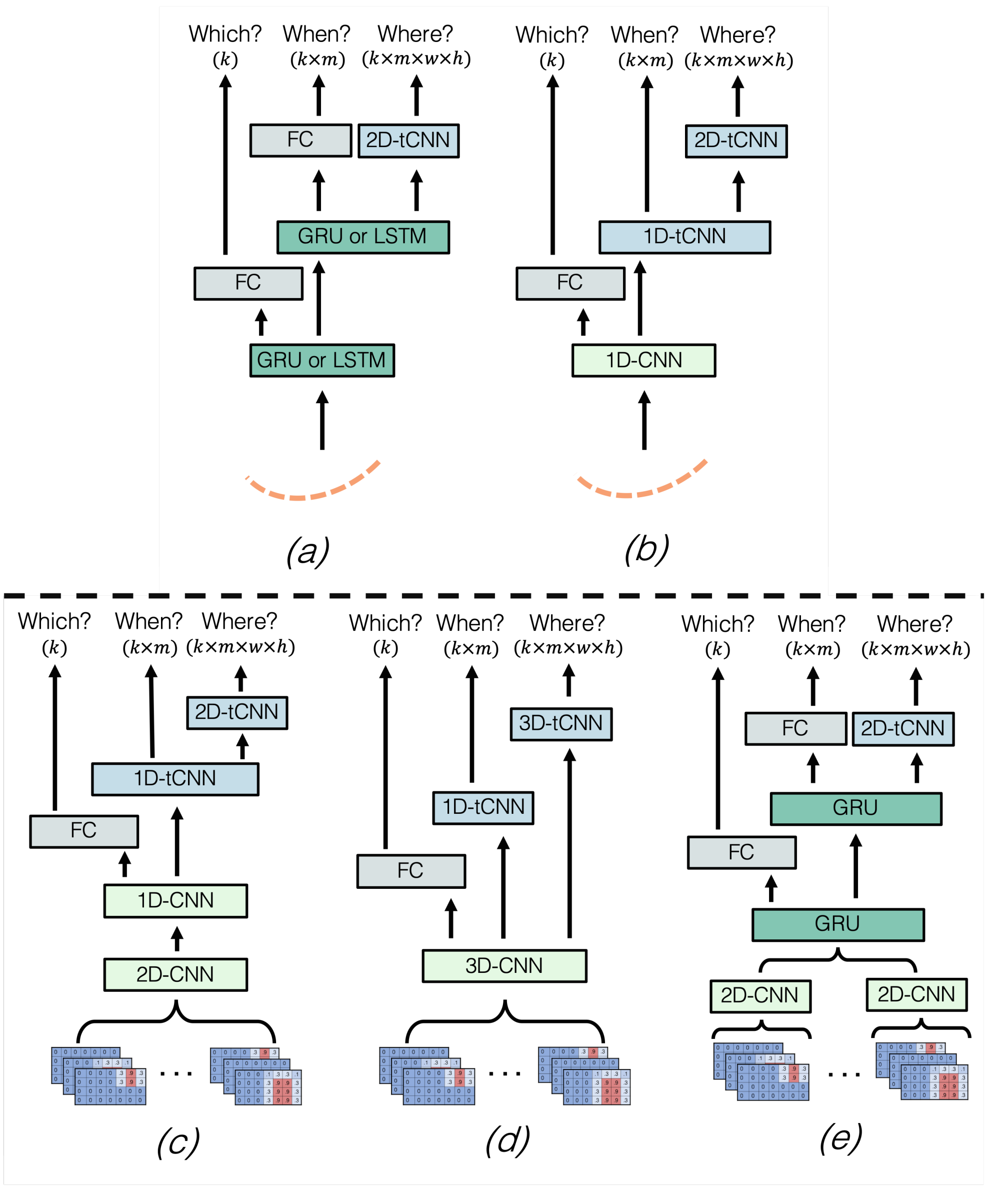


Figure 5.5: **MCTF models.** We introduce 2 coordinate-trajectory based (top) and 3 trajectory-tensor based (bottom) approaches for MCTF. *(a)* A recurrent encoder-decoder adapted from STED [2]. *(b)* A 1D-CNN adapted from FPL [31]. *(c)* A CNN approach with separated layers for spatial and temporal feature extraction. *(d)* A CNN approach with 3D convolutions for extracting spatial and temporal features simultaneously. *(e)* A hybrid CNN-GRU approach which uses a CNN to extract spatial features which are passed to an GRU for extracting temporal features. Note that for coordinate trajectory models, a separate model is created for each camera. In contrast, trajectory tensor models use a single unified model for all cameras. Each model is trained with either the *which*, *when*, or *where* MCTF formulation.

decoder GRU is followed by 2D Transposed Convolutional Neural Network (tCNN) layers for spatial upsampling.

LSTM. Our LSTM model is the same as our GRU, with an alternative recurrent unit for both encoder and decoder.

1D-CNN. 1D-CNNs have received some attention as an alternative to the de facto recurrent models for tasks involving time series [194], including trajectory forecasting [31]. We use the encoder architecture proposed by Yagi et al. [31], with modified decoders adapted for the MCTF formulation. For the *which* task, we use a fully-connected classification layer. For the *when* task, we use 1D transposed convolutional layers as a decoder with 4 layers. For the *where* task, the 1D transposed convolutional layers are followed by 2D transposed convolutional layers for spatial upsampling. Similarly to STED, we use only the trajectory feature extractor for a fair comparison with other methods.

Trajectory tensor approaches

Here we present our trajectory tensor approaches which are shown in Figure 5.5 (c), (d), and (e). Our proposed representation enables efficient modelling of object trajectories across multiple camera views. Therefore, each approach consists of a single unified model for all cameras, which is less cumbersome and more scalable than the multi-model approach for coordinate trajectories. Each model uses an encoder-decoder architecture with the same encoder for each task and a task-dependent decoder.

2D-1D-CNN. We use a CNN consisting of three 2D convolutional and pooling layers for spatial feature extraction followed by three 1D convolutional and pooling layers for temporal feature extraction. Using separate 2D and 1D convolutional layers rather than 3D layers reduce the number of network parameters and is inspired by existing models for video classification [195]. We use a fully-connected layer as decoder for the *which* formulation, 5 1D transposed convolution layers for the *when* formulation, and 3 1D transposed convolution followed by 3 2D transposed convolution layers for the *where* formulation.

3D-CNN. We use a 3D-CNN with four layers for spatio-temporal feature extraction. 3D convolutions can simultaneously extract spatial and temporal features and have seen some success for tasks such as action recognition [196]. We use a fully-connected layer as decoder for the *which* formulation, 5 1D transposed convolution layers for the *when* formulation, and 4 3D transposed convolution layers for the *where* formulation.

CNN-GRU. We train a convolutional auto-encoder to reduce a trajectory tensor at a single timestep (dimension $|\mathcal{C}| \times w \times h$) to a feature vector of dimension 512. This encoding is then used as the input to a GRU which predicts future feature vectors, which are decoded as shown in Figure 5.5 (e). The auto-encoder is first pre-trained until convergence and then trained end-to-end with the GRU. This architecture is inspired by the model proposed by

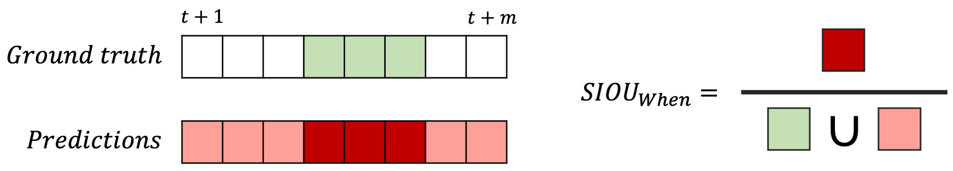


Figure 5.6: **Visual representation of the $\mathbf{SIOU_{when}}$ metric.** The sum of the predictions that overlap with the ground truth is divided by the sum of the predictions that do not overlap with the ground truth plus the sum of the ground truth values. Shown is an example of computing the $SIOU_{when}$ for a single camera. If a target appears in more than one camera, we take the mean $SIOU_{when}$ value for all cameras in which the target appears.

Luc et al. [45], which uses a similar strategy of forecasting future convolutional features for predicting future instance segmentation maps. We use a fully-connected layer as decoder for the *which* formulation, a GRU followed by a fully-connected layer for the *when* formulation, and a GRU followed by 3 2D transposed convolution layers for the *where* formulation.

5.3.3 Evaluation Strategy

Due to the high levels of uncertainty and the multi-modal nature of the MCTF, traditional SCTF metrics such as average and final displacement errors are not well-suited to MCTF. Alternatively, some works generate multiple trajectories and select the one most similar to the ground truth [84]; however, this evaluation method is optimistic as performance comparable to sophisticated methods can be obtained using a simple constant velocity model that generates trajectories with high variance [86]. Owing to shortcomings of displacement error metrics, we compute the Average Precision (AP) for all problem formulations and plot precision-recall curves in addition to computing displacement error metrics. We choose precision-recall curves over Receiver Operating Characteristic (ROC) curves. We find ROC to be an overly-optimistic metric due to the considerable class imbalance between the positive (object presence) and negative (no object presence) classes. We define AP_{which}, AP_{when}, and AP_{where} for the three problem definitions respectively. For each, we take the entire set of predictions and compute the Average Precision with respect to the entire set of targets.

The AP metrics provide a holistic interpretation of model performance but are not easy to interpret. Therefore, we propose two new, more interpretable metrics for MCTF evaluation, the soft-intersection-over-union ($SIOU$) for the when and where problem formulations. The $SIOU$ for evaluating the *when* formulation is as follows:

$$SIOU_{when} = \frac{1}{|\mathcal{C}^+|} \sum_{c}^{\mathcal{C}^+} \frac{\sum_t^{\mathcal{T}^+} \hat{y}_t^c}{\sum_t^{\mathcal{T}} \hat{y}_t^c} \tag{5.1}$$

where $\mathcal{C}^+$ and $\mathcal{T}^+$ are sets of true positive cameras and timesteps, respectively. $\hat{y}_t^c \in (0, 1)$ is the predicted value for the presence the object in camera c at timestep t. Intuitively, $SIOU_{when}$ is a value between 0 and 1, representing the temporal overlap between the predicted and ground truth timesteps for all cameras where the individual appears. Figure 5.6 depicts the $SIOU_{when}$ for a single camera view.

We compute the $SIOU$ for evaluating the *where* problem as follows:

$$SIOU_{where} = \frac{1}{|\mathcal{C}^+|}\sum_{c}^{\mathcal{C}^+}\frac{1}{|\mathcal{T}^+|}\sum_{t}^{\mathcal{T}^+}\frac{\sum_i^{\mathcal{I}^+}{}_i\hat{y}_t^c}{\sum_i^{\mathcal{I}}{}_i\hat{y}_t^c} \tag{5.2}$$

where $\mathcal{I}^+$ is the set of grid cells where the individual is present. $SIOU_{where}$ is similarly a value between 0 and 1 representing the spatial overlap between predicted and ground truth grid cell locations for all true positive cameras and timesteps. Figure 5.7 depicts the $SIOU_{where}$ for a single timestep and camera view. We use these new $SIOU$ metrics because the metrics are more easy to interpret, unlike the AP metric. Both $SIOU$ metrics are a value between 0 and 1 representing the overlap between targets and ground truth, where a value of 0 mean no overlap, 1 mean perfect overlap, and everything else in between. We use the $SIOU$ metrics jointly with AP metrics as the $SIOU$ metrics only consider True Positives, whereas AP metrics consider both True and False Positives. We therefore suggest both metrics are considered when evaluating MCTF models.

In addition to our new metrics, we also adapt the standard Average Displacement Error (ADE) and Final Displacement Error (FDE) metrics used widely in the SCTF literature [32, 33, 83, 84]. We refer to these metrics as ADE_{where} and FDE_{where} hereinafter to distinguish them from their SCTF counterparts. We compute a single coordinate value for a heatmap $\boldsymbol{H_i}$ by computing the center of mass, $R_{x,y}$, as follows:

$$R_{x,y} = ((\frac{1}{M}\sum_{m\in\boldsymbol{H_i}} x_m \cdot r_x^m), ((\frac{1}{M}\sum_{m\in\boldsymbol{H_i}} y_m \cdot r_y^m))), \tag{5.3}$$

where M is the sum of all elements in $\boldsymbol{H_i}$. Note that $\boldsymbol{H_i}$ denotes a heatmap for camera i at a single timestep. r_x^m and r_y^m respectively denote the x and y positions of the m^{th} element in $\boldsymbol{H_i}$, and x_m and y_m denote heatmap values at position m. This value corresponds to the centroid of the target bounding box in the image space, weighted by heatmap value. As a target can appear in more than one camera view, we compute ADE_{where} and FDE_{where} separately for each camera the target appears and take the mean. ADE_{where} and FDE_{where} are valid for the *where* problem formulation.

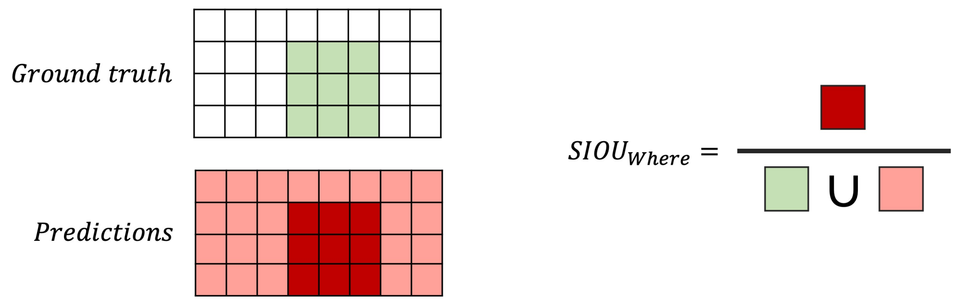


Figure 5.7: **Visual representation of the $\mathbf{SIOU_{where}}$ metric.** The sum of the predictions that overlap with the ground truth is divided by the sum of the predictions that do not overlap with the ground truth plus the sum of the ground truth values. Shown is an example of computing the $SIOU_{where}$ for a single timestep in a single camera. We then compute the mean $SIOU_{where}$ for all timesteps and in all camera views in which the target appears.

5.4 Performance Evaluation

In this section, we introduce the evaluation dataset, baseline approaches, and assess the performance of each model for the *which*, *when*, and *where* tasks.

5.4.1 Warwick-NTU Multi-Camera Forecasting Dataset

We introduced the Warwick-NTU Multi-Camera Forecasting (WNMF) dataset in Chapter 3. WNMF was collected specifically for MCTF using a set of 15 cameras in a building on the Nanyang Technological University campus and contains both overlapping and non-overlapping views recorded over 20 days. The dataset consists of cross-camera trajectories where an individual departs from one camera view and then re-appears in another after no more than 12 seconds. An individual may also be visible in any number of other camera views during this tracking period. We use a robust 5-fold cross-validation setup in all experiments that follow, where training, validation, and testing sets all contain footage recorded on different days. The WNMF dataset is available to download at `https://github.com/olly-styles/Multi-Camera-Trajectory-Forecasting`.

5.4.2 Baseline Approaches

In addition to the models introduced in Section 5.3.2, we also evaluate several baselines.

Shortest real-world distance. We use the physical distance between cameras in the real world and predict the camera closest to the current camera. This baseline applies to the *which* formulation only.

Training set mean. We consider all training set observations for a particular camera and take the mean of all ground truth labels in the training set.

Most similar trajectory. We find the most similar trajectory in terms of ℓ_2 distance in the same camera from the training set to the observed trajectory and predict the same label.

Hand-crafted features. We extract some features from the bounding boxes and classify them with a fully-connected network. Our 10-dimensional hand-crafted feature vector contains velocity in x and y direction, acceleration in x and y direction, last observed bounding box height and width, and four coordinates. We compute all features with respect to the 2D coordinate system as captured by the camera.

5.4.3 *Which* Camera Will the Target Appear?

Experimental setup. We use our proposed network architectures (Section 5.3.2) and baselines (Section 5.4.2). For the coordinate trajectory approaches, we train a separate model for each camera. To adapt the SCTF models for MCTF, we leave encoders unchanged and change decoders to fully-connected output layers of size 15, the number of cameras in the WNMF dataset. We use a binary cross-entropy loss function with a sigmoid activation function at the output layer. The activation maps each output to a value between 0 and 1, representing the predicted probability of appearance for the individual in each camera. Coordinate trajectory and trajectory tensor models are trained with the Adam optimizer using learning rates of 1×10^{-3} and 1×10^{-4}, respectively, chosen using cross-validation. Coordinate trajectory encoders extract a feature vector of size 128, whereas trajectory tensor approaches extract a feature vector of size 512 as a single encoder is shared across all cameras and therefore requires a larger representation capacity. All models are trained using a batch size of 64. We use a heatmap $\boldsymbol{H}$ of size either 16×9, 32×18, or 48×27 and a Gaussian smoothing kernel size between 0 and 4 as input, chosen using cross-validation. The impact of changing these parameters is investigated in Section 5.4.7.

Results. The AP_{which} for each model is shown in Table 5.1, and precision-recall plots are shown in Figure 5.15a. Coordinate trajectory approaches outperform our 4 baselines, and trajectory tensor approaches perform the best.

5.4.4 *When* Will the Target Appear?

Experimental setup. We use the same encoders for evaluating the *when* task, with either 1D-transposed convolutional layers or recurrent unit for decoding as shown in Figure 5.5. Each model is trained with the binary cross-entropy loss, and the trajectory tensor models are trained with the same hyperparameters as the *which* problem formulation.

Table 5.1: ***Which* results.** Given observations from one camera, each model predicts which camera(s) the person will re-appear.

	Model	AP_{which} ($\uparrow$)
Baselines	Shortest real-world distance	45.4
	Training set mean	68.9
	Most similar trajectory	65.3
	Hand-crafted features	76.2
Coordinate trajectories	LSTM	84.1
	GRU	84.0
	1D-CNN	83.3
Trajectory tensors	2D-1D-CNN	87.1
	3D-CNN	**87.5**
	CNN-GRU	86.1

Results. The AP_{when} and $SIOU_{when}$ for each model are shown in Table 5.2, and precision-recall plots are shown in Figure 5.15b. We observe a similar trend to the results of the *which* task, with the 3D-CNN trajectory tensor model performing best. The most similar trajectory baseline method performs well in terms of $SIOU_{when}$, but poorly in terms of AP_{when}. This result suggests that this approach effectively predicts the correct time window an individual will be visible in the target camera but often predict the wrong camera.

5.4.5 *Where* Will the Target Appear?

Experimental setup. We use the same target heatmap size of 16×9 regardless of the input heatmap size for a fair comparison. We use this small heatmap size to reduce computational complexity, and 16×9 is the smallest possible while maintaining the original aspect ratio of the video frames. The output trajectory tensor is, therefore, of dimension $15 \times 60 \times 16 \times 9$. Unlike when using trajectory tensors as inputs, we do not apply Gaussian smoothing to the ground truth trajectory tensor targets. We use 2D or 3D transposed convolutional layers to upsample extracted feature vectors to trajectory tensor outputs, representing an individual's future location. The trajectory tensor models are trained with the same hyperparameters as in Section 5.4.3, again using the binary cross-entropy loss.

Results. The AP_{where}, $SIOU_{where}$, ADE_{where}, and FDE_{where} for each model is shown in Table 5.3 and Table 5.4, and precision-recall plots are shown in Figure 5.15c. Note that due to the considerably increased complexity of the

Table 5.2: ***When* results.** Given observations from one camera, each model predicts in which camera(s) the person will re-appear, and when (i.e., in which timesteps) they will be present.

	Model	AP_{when} (↑)	$SIOU_{when}$ (↑)
Baselines	Training set mean	61.7	40.0
	Most similar trajectory	46.6	**56.7**
	Hand-crafted features	67.1	48.7
Coordinate trajectories	LSTM	76.8	51.8
	GRU	77.5	53.7
	1D-CNN	77.1	52.8
Trajectory tensors	2D-1D-CNN	77.4	54.4
	3D-CNN	**79.0**	53.9
	CNN-GRU	69.1	42.0

Where problem formulation compared to *Which* and *When*, the AP is much lower. Results for this problem formulation are more mixed than for the *Which* and *When*, and some methods perform well on one metric but poorly on others. For example, the hand-crafted feature baseline performs the best in terms of ADE_{where} and FDE_{where}, but poorly in terms of AP_{where}. Unlike the AP_{where} metric, the displacement error and $SIOU_{where}$ metrics do not take into account erroneous predictions in camera views that the target does not appear. Therefore, this result suggests that the hand-crafted features baseline accurately forecasts the target trajectory but incorrectly assigns a high likelihood to the target appearing in other camera views.

5.4.6 Qualitative Results

Here, we visualise successful and unsuccessful model predictions for each problem formulation. For reference, we reproduce the WNMF dataset camera network topology in Figure 5.8.

***Which* problem formulation.** Two successful predictions of the next camera the individual will re-appear are shown in Figure 5.9. In the upper example, an individual departs from Camera 5 and continues straight, re-appearing in Camera 7. Our trajectory-tensor based model also assigns some probability that the individual will re-appear in Camera 4, positioned further down the corridor. In the lower example, an individual departs from Camera 11. Our model assigns Camera 6 and Camera 10 as the most likely next cameras, which both have a view of the corridor. An unsuccessful prediction is shown in Figure 5.10. All models incorrectly assign Camera 6 as the most likely next

Table 5.3: ***Where*** **results.** Given observations from one camera, each model predicts which camera(s) the person will re-appear, in which timesteps they will be present, and where in the camera view they will appear. Best results are highlighted in **bold** typeface, second best are <u>underlined</u>.

	Model	AP_{where} (↑)	$SIOU_{where}$ (↑)
Baselines	Training set mean	28.4	14.4
	Most similar trajectory	11.8	**29.8**
	Hand-crafted features	25.5	20.2
Coordinate trajectories	LSTM	16.3	8.7
	GRU	16.5	8.8
	1D-CNN	16.4	8.6
Trajectory tensors	2D-1D-CNN	<u>34.5</u>	22.6
	3D-CNN	**37.4**	<u>22.9</u>
	CNN-GRU	22.4	12.8

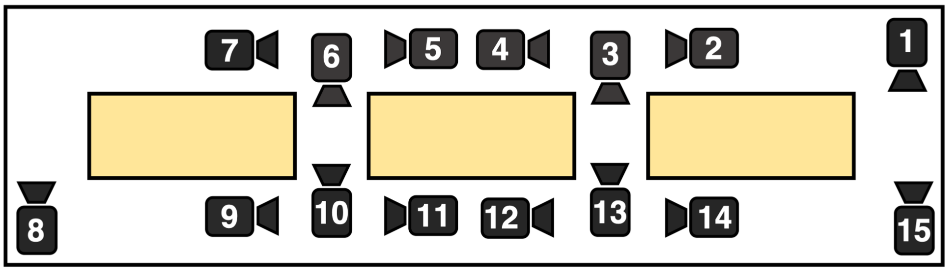


Figure 5.8: **WNMF camera network topology.** The dataset contains 15 cameras in an indoor environment.

camera rather than Camera 10. These two camera views are overlapping, so either may detect the target after departing from Camera 11.

When **problem formulation.** Two successful predictions of when the individual will re-appear in the target camera views are shown in Figure 5.11. In the upper example, an individual departs from Camera 5, turning the corner and re-appearing in Camera 7 after a short delay. Some models incorrectly assign a high probability to the individual appearing in Camera 6, which would be the case if they continued walking straight rather than turning at the junction. Our trajectory tensor-based model correctly assigns a high probability to Camera 7 after a short delay. In the lower example, an individual departs from Camera 14 and turns left at the junction, re-appearing in Camera 3 after 16 timesteps. Most models assign high probabilities to Camera 12, where the target would appear if they did not turn. Our proposed model correctly

Table 5.4: ***Where* results using traditional SCTF metrics.** Given observations from one camera, each model predicts which camera(s) the person will re-appear, in which timesteps they will be present, and where in the camera view they will appear. Best results are highlighted in **bold** typeface, second best are <u>underlined</u>. SCTF metrics do not penalize models for predicting a trajectory in the wrong camera view, and therefore less appropriate than our proposed metrics for evaluating MCTF performance.

	Model	ADE_{where} (↓)	FDE_{where} (↓)
Baselines	Training set mean	226.1	<u>260.4</u>
	Most similar trajectory	312.9	375.4
	Hand-crafted features	**216.2**	**255.2**
Coordinate trajectories	LSTM	285.9	365.5
	GRU	286.0	366.0
	1D-CNN	287.2	368.1
Trajectory tensors	2D-1D-CNN	225.9	265.8
	3D-CNN	<u>223.3</u>	279.2
	CNN-GRU	280.1	351.7

predicts the re-appearance in Camera 3 with steadily increasing probability over time. However, the model prediction does not cross a prediction threshold of 0.5 until after the target has been visible in camera 3 for several timesteps. An unsuccessful prediction is shown in Figure 5.12. All models incorrectly predict the target will appear in Camera 6 after a short delay rather than Camera 10. Camera 6 does not pick up the individual, possibly due to low lighting or occlusion. The individual is instead captured in Camera 10 after a long delay.

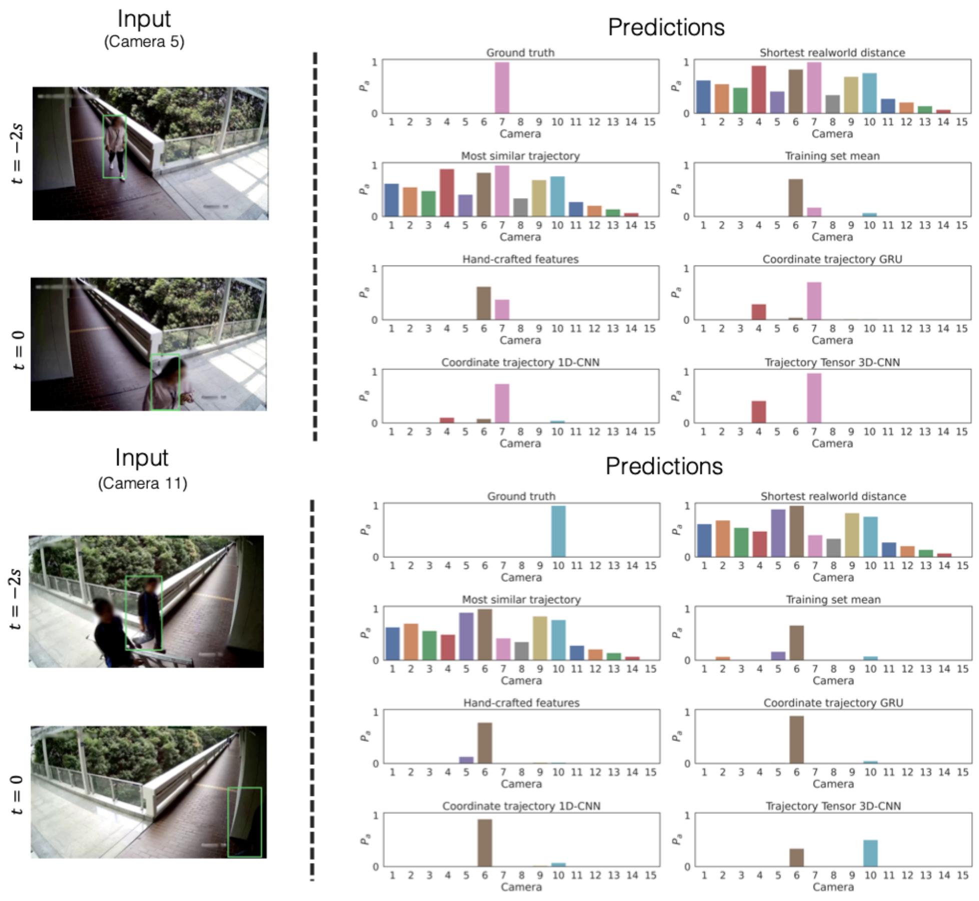


Figure 5.9: **Example successful *which* predictions.** Our model successfully anticipates the next camera an individual will re-appear. As the maximum time between target observations is 12 seconds, individuals generally re-appear in a nearby camera.

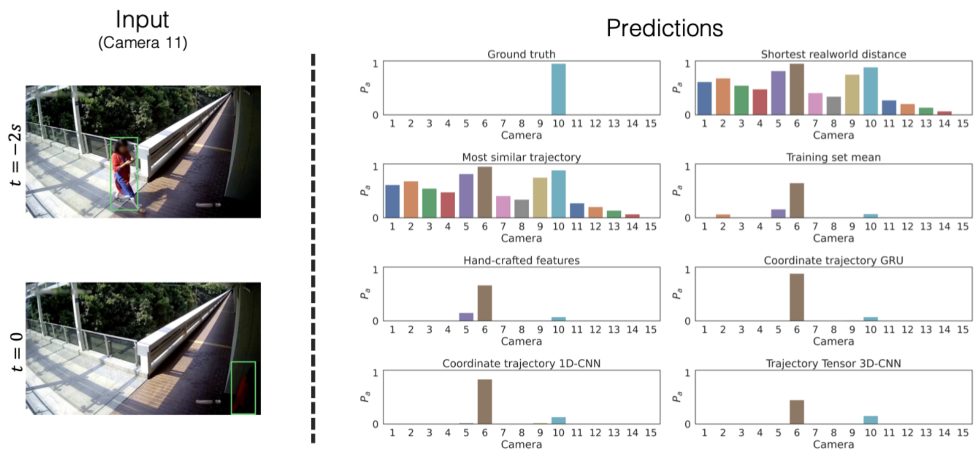
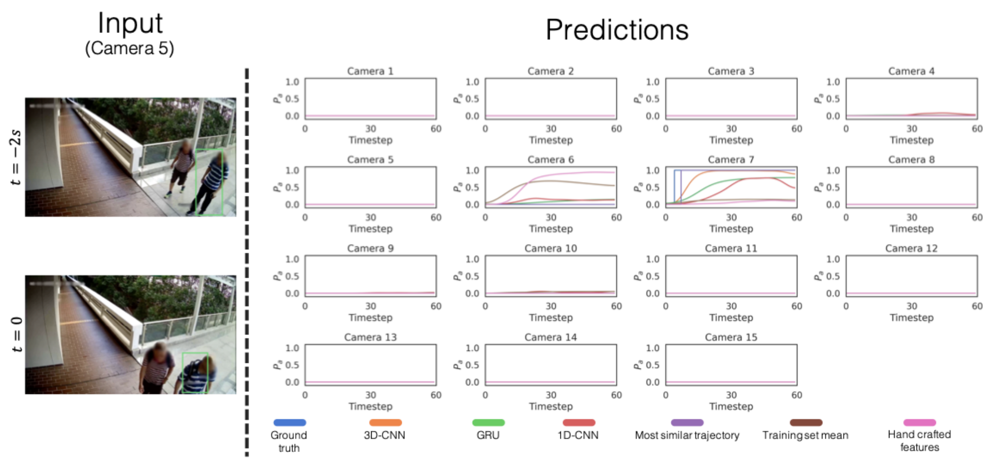


Figure 5.10: **Example unsuccessful *which* predictions.** All models incorrectly assign camera 6 as the most likely next camera.



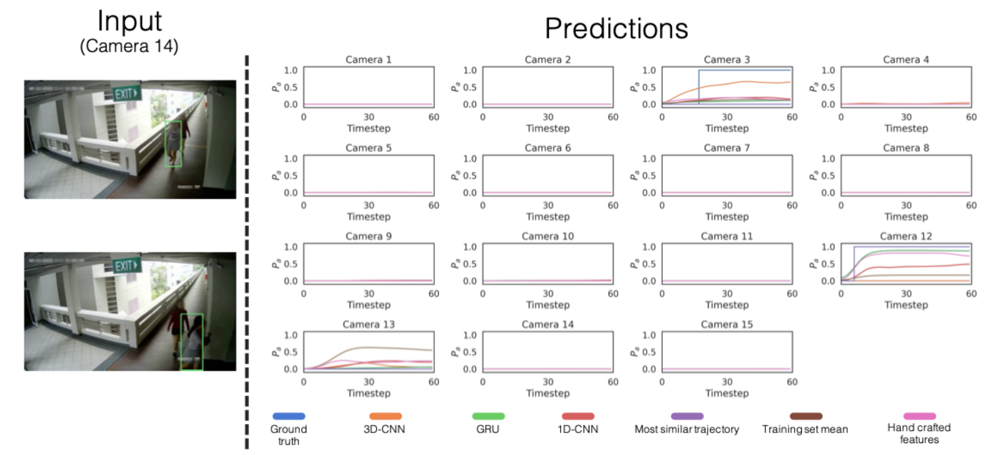


Figure 5.11: **Example successful *when* predictions.** Our model correctly anticipates individuals turning at intersections and approximates the time of re-appearance in another camera in the network.

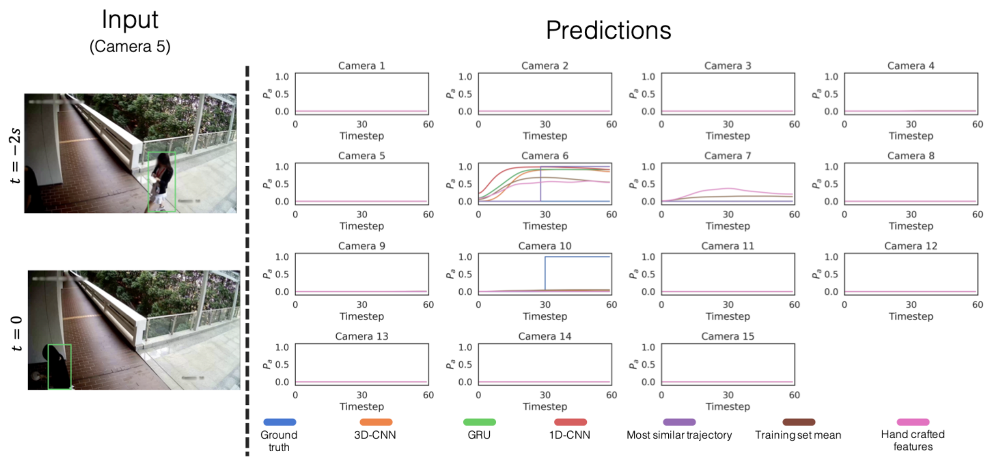


Figure 5.12: **Example unsuccessful *when* predictions.** This is a similar scenario as the *which* prediction in Figure 5.10 where the individual is predicted to appear in the wrong camera out of two overlapping views.

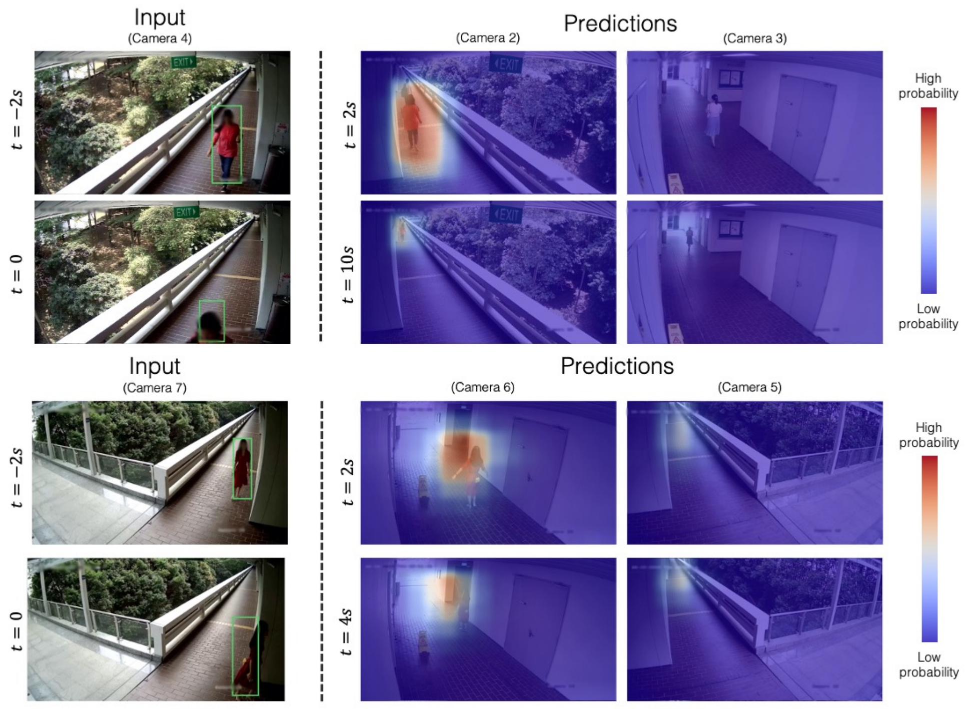


Figure 5.13: **Example successful *where* predictions.** Here we show example *where* predictions for our trajectory tensor model with 3D-CNN architecture. The predictions are made in a 16×9 grid which are smoothed with a Gaussian kernel for cleaner visualization.

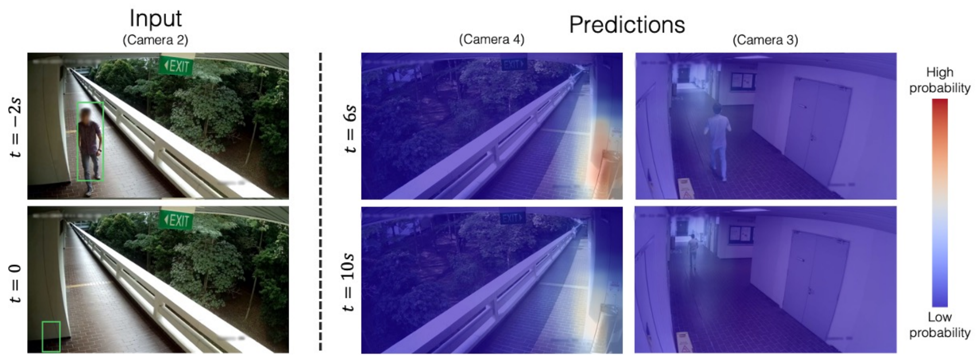


Figure 5.14: **Example unsuccessful *where* prediction.** The predictions are made in a 16×9 grid which are smoothed with a Gaussian kernel for cleaner visualization.

***Where* problem formulation.** Two successful predictions of where the individual will re-appear are shown in Figure 5.13. In the upper example, the individual departs from Camera 4 and continues to walk down the corridor. Our model correctly identifies where in the view of Camera 2 the individual will be visible. Note how the size of the predicted region decreases in later timesteps. In the lower example, we show a multi-modal prediction. Our model assigns some probability of the individual turning at the junction and appearing in Camera 6, and some probability of the individual continuing straight and appearing in Camera 5. A higher probability is assigned to Camera 6. An unsuccessful prediction is shown in Figure 5.14. Here the model predicts that the individual will stop at the water dispenser (visible in Camera 4) rather than turning right (visible in Camera 3). This result highlights the reliance on training data and the importance of gathering sufficient data to cover various scenarios.

5.4.7 Ablation Studies

Multi-view trajectory tensors. Our proposed trajectory tensor data representation enables us to model the same individual captured in multiple camera views. To study the impact of multiple camera views, we train our trajectory tensor-based models using only one of the available views, i.e., we set each heatmap at each timestep to the zero matrix for all but one of the camera channels $c \in \mathbf{Z}$. The results in Table 5.5 show a comparison of our trajectory tensor models with a single-view trajectory (each heatmap is 0 for all but one of the cameras $c \in \mathbf{Z}$) and a multi-view trajectory (where more than one of the camera channels $c \in \mathbf{Z}$ may be non-zero). The results show that the models can make use of the location information available in multiple camera views.

Heatmap size and smoothing. We investigate the impact of heatmap size

Table 5.5: **Multi-view trajectory tensor results.** Comparison of single-view and multi-view trajectories. Observing a trajectory in multiple camera views improves model performance in most cases.

Model	Multi-view trajectories	AP_{which}	AP_{when}	AP_{where}
2D-1D-CNN	No	81.8	76.6	**37.4**
	Yes	**87.1**	**77.4**	34.5
3D-CNN	No	83.0	77.1	**38.8**
	Yes	**87.5**	**79.0**	37.4
CNN-GRU	No	83.4	68.4	**23.6**
	Yes	**86.1**	**69.1**	22.4

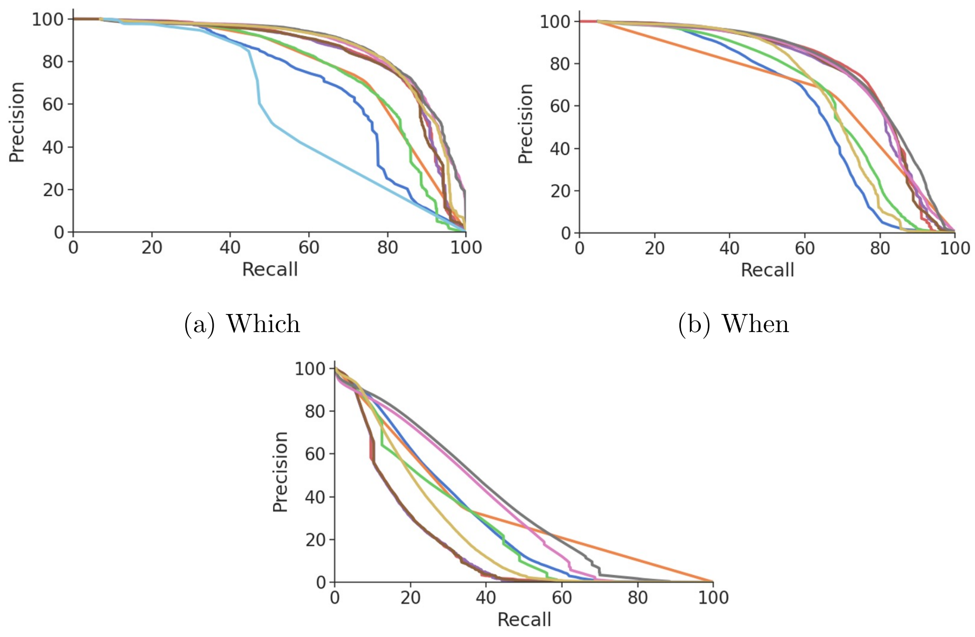


Figure 5.15: **Precision-recall plots.** Precision and recall of each model for all three problem formulations. Colours represent Training Set Mean (**Dark Blue**), Most Similar Trajectory (**Orange**), Shortest Real-world Distance (**Light Blue**), Hand-crafted Features (**Green**), GRU (**Red**), LSTM (**Purple**), 1D-CNN (**Brown**), 2D-1D-CNN (**Pink**), 3D-CNN (**Grey**), and CNN-GRU (**Yellow**). The performance of all models for the *Where* problem formulation is lower than the *Which* and *When* formulations owing to the considerably increased difficulty.

and standard deviation of the Gaussian filter for smoothing. Larger heatmap sizes afford models more representation power at the cost of more parameters and a tendency to overfit the training data. On the other hand, smoothing has a regularising effect by reducing the impact of errors during the detection and tracking stages. We suggest, therefore, that both heatmap size and smoothing sigma should be tuned in tandem. Figure 5.16 shows the impact of heatmap size and smoothing sigma.

5.4.8 Multi-Target Multi-Camera Trajectory Forecasting

Thus far, we have focused on the problem of Single-Target MCTF, where models predict the trajectory of a single target. We extend this to Multi-Target Multi-Camera Trajectory Forecasting (MT-MCTF), allowing us to predict multiple target trajectories simultaneously. The WNMF dataset, as first introduced in Chapter 3, does not contain MT-MCTF labels, so we first generate a new set of annotations amenable to MT-MCTF.

Experimental setup. We start our trajectory predictions when a target

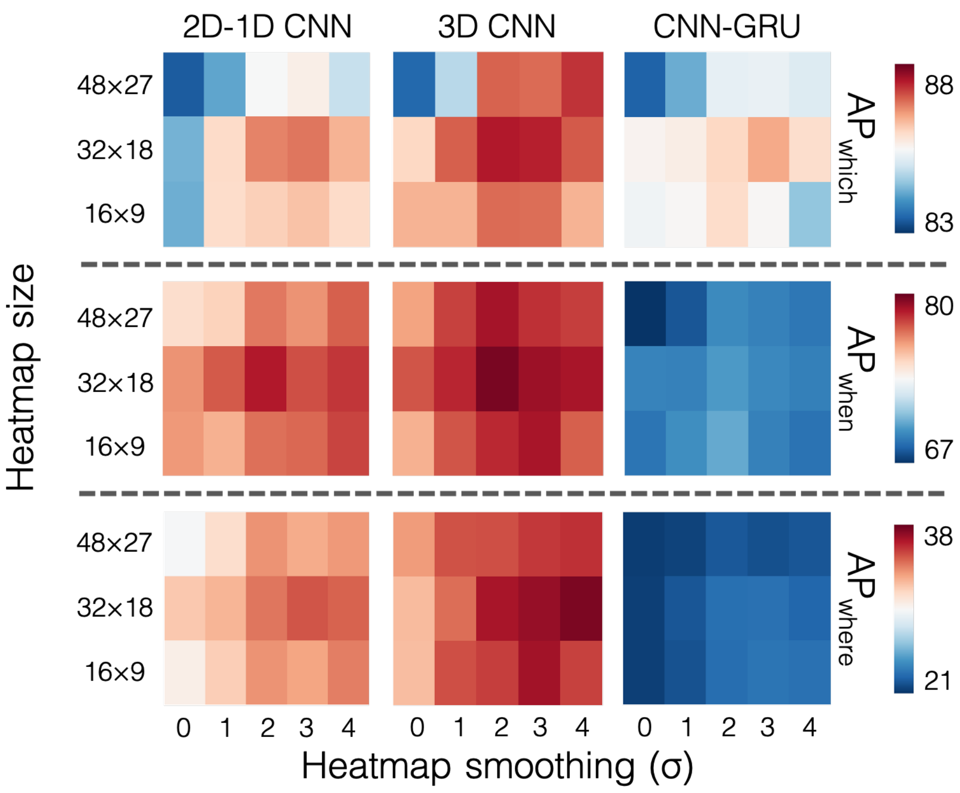


Figure 5.16: **Impact of heatmap size and smoothing.** Here we show the impact of heatmap size and smoothing sigma using 5-fold cross validation. We observe that larger heatmap sizes are most beneficial for the *Where* problem formulation where more fine-grained predictions are required.

departs from a particular camera view. Therefore, for multi-target MCTF, we group trajectories into multi-target groups where each target departs at a similar time. Specifically, we group trajectories into bins of 2 seconds (10 timesteps), such that targets with trajectories ending within 2 seconds of each other are considered multi-target trajectories. As we predict the future trajectory using 2 seconds of past data, grouping trajectories in this way ensures that trajectories overlap temporally for at least one timestep. Trajectories may be visible in the same or different camera views to other trajectories in the same multi-target group. To focus on the multi-target problem, we discard trajectories where only one target is visible, resulting in 58 data samples, containing a total of 119 trajectories, an average of 2.05 trajectories per sample. The maximum number of trajectories in a single data sample is 4. Due to the small size of the multi-target subset (119 trajectories compared to $1,967$ in the full dataset), results should be treated with caution.

We evaluate our trained models and baselines on the multi-target subset of WNMF, which is available to download alongside the full dataset. We use 5-fold cross-validation, using a model trained on data collected on different days to

Table 5.6: **Multi-target multi-camera trajectory forecasting results.** Comparison of each model on a multi-target subset of the WNMF dataset. Trajectory tensor models stack multi-targets across an additional tensor dimension. Other models process each trajectory sequentially.

	Model	AP_{which} ($\uparrow$)	AP_{when} ($\uparrow$)	AP_{where} ($\uparrow$)
Baselines	Shortest real-world distance	39.5	N/A	N/A
	Training set mean	63.9	56.8	28.6
	Most similar trajectory	58.6	40.7	12.2
	Hand-crafted features	48.0	36.6	10.0
Coordinate trajectories	LSTM	77.8	**73.1**	19.5
	GRU	73.5	72.3	20.7
	1D-CNN	75.3	71.6	20.2
Trajectory tensors	2D-1D-CNN	70.8	70.6	**33.4**
	3D-CNN	**80.0**	63.3	30.2
	CNN-GRU	46.9	36.3	18.0

the test set. Each model is trained until the performance on the validation set saturates for each fold, and is then evaluated on the test set. For our baselines and coordinate trajectory approaches, we process each trajectory sequentially. As our trajectory tensor-based approaches use a single unified model for all camera views, we process the multi-target trajectories in parallel by stacking trajectories along the batch dimension. Therefore, the input trajectory tensors are of dimension $b \times |\mathcal{C}| \times t \times w \times h$, where b is the number of targets, and other dimensions are the same as described in Section 5.3.1. Stacking inputs along the batch dimension enables us to simultaneously predict trajectories for any arbitrary number of targets that fit into system memory.

Results. Results for multi-target MCTF are shown in Table 5.6. Learned approaches outperform the baselines, although there is variation between the performance of the three tasks across models. This variation may be due to the small size of the multi-target subset compared to the full dataset. Our 2D-1D-CNN and 3D-CNN trajectory tensor models outperform coordinate trajectory approaches for the *which* and *where* tasks and process all targets and camera views using a single unified model rather than predicting trajectories for each target sequentially using a separate model for each camera.

5.5 Discussion

Our results show that using trajectory tensors has several advantages over traditional coordinate trajectories for MCTF. Representing trajectories as tensors allows us to model relative object locations in multiple camera views simultaneously, which improves MCTF performance when used as inputs and provides an intuitive way to represent multi-modal futures when used as targets. Trajectory tensors are also considerably more efficient than using coordinate trajectories in an MCTF setting, as a single model can be used rather than creating separate models for each camera. Note that our approach and other baselines require that the camera network is the same at training and testing time.

Trajectory Tensors are robust to camera failure. As objects are potentially visible in multiple camera views, the trajectory may still be modelled if the object is not visible in a particular camera view or if the camera is not operational. Trajectory tensors also facilitate the representation of trajectories in multiple cameras simultaneously. Furthermore, trajectory tensors are an intuitive way to represent multi-modal future trajectories. Existing work has explored other approaches to representing multi-modal futures, such as using generative models to generate many possible futures. Trajectory Tensors model multiple futures in one shot.

We find the 3D-CNN architecture consistently performs the best across MCTF tasks. 3D-CNNs have traditionally only seen major success in settings where vast amounts of data are available for training [196]. However, in our setting, we use resolutions of up to 48×27 rather than the 224×224 or higher commonly used in activity recognition, which considerably reduces the number of parameters and facilitates training on smaller datasets.

5.6 Summary

We have developed a complete framework for MCTF formulated in a hierarchy of three spatio-temporal localization tasks: In *which* camera, *when*, and *where* will the object(s) appear? Our work is the first to address the challenges of trajectory forecasting in a multi-camera environment. We introduced the idea of the *trajectory tensor* - a new trajectory representation that facilitates the encoding of multi-camera trajectories and associated uncertainty. Trajectory tensors are an attractive alternative to the traditional coordinate trajectory representation used in previous works. Our experiments demonstrate the accurate performance of trajectory tensor models on the WNMF dataset. Furthermore, we found that using data from multiple camera views proves to be beneficial for MCTF using trajectory tensors. We envision our MCTF framework and model will enhance multi-camera surveillance systems, complementing existing models for person Re-ID and multi-camera tracking.

Chapter 6

Conclusions and Future Work

In this thesis, we have proposed a set of strategies for pedestrian trajectory forecasting in video. In this chapter, we summarise our contributions, discuss applications, and consider future work.

6.1 Summary of Contributions

In Chapter 3, we presented a method for generating machine-annotated datasets for egocentric SCTF, and a machine-aided annotation method for generating MCTF datasets. Our methods address the issue of prohibitively small datasets in the trajectory forecasting field by providing a framework for collecting data with minimal human labour. We collected new annotation for the BDD-100k [105] dataset, and collected the new Citywalks and WNMF datasets, which are made available to download for the research community. In Chapter 4, we presented DTP, a method for vehicle-view trajectory forecasting, and STED, a method for our newly-defined MOF problem formulation. DTP used optical flow as an input modality to predict trajectories, and performance improved on a human-annotated dataset after training the model on a machine-annotated dataset, annotated using our method introduced in Chapter 3. STED used past object bounding boxes in addition to optical flow to predict full future object bounding boxes. STED outperformed other methods on the Citywalks dataset, and also outperformed other methods on the human-annotated MOT-17 dataset after training on the machine-annotated Citywalks dataset *without fine-tuning.* In Chapter 5, we introduced the MCTF framework. MCTF extends the existing trajectory forecasting problem formulation to a multi-camera environment, opening new applications for trajectory forecasting. We introduced the trajectory tensor, which enables us to model multi-camera trajectories with a single model elegantly. Our proposed model uses a 3D-CNN backbone and outperforms existing SCTF works adapted for MCTF.

6.2 Applications

The methods proposed in this thesis have a range of applications. We identify four main application domains, shown in Figure 6.1.

6.2.1 Intelligent Vehicles

The rapid improvements in pedestrian detection systems have led many car manufacturers, such as Toyota, to implement ADAS to aid drivers in critical situations [197]. Such systems provide warnings or even apply automatic braking if a pedestrian is detected in front of the vehicle but work only in a limited capacity for potentially dangerous events that require the anticipation of future intentions of pedestrians. Forecasting future pedestrian trajectories is vital for an AV to be safe, particularly in light of the recent pedestrian fatality as a result of an AV malfunction [198].

6.2.2 Mobile Robotics

Service robots in environments shared with humans are beginning to become more common in both domestic and industrial settings and generally travel at low speeds in order to minimise the frequency and impact of collisions [199]. Trajectory prediction is a crucial component for such robots to travel more quickly in environments shared with humans or to assist humans based on their likely future movements [67].

6.2.3 Intelligent Traffic Surveillance

A MCTF system could be deployed for vehicle prediction to improve traffic flow and transport efficiency. Existing methods use infrastructure-to-vehicle communication in order to enable more effective planning for intelligent vehicles [200]. With an accurate MCTF system, overhead CCTV cameras could relay information about future vehicle trajectories, helping to improve traffic flow at signalised intersections and reduce congestion without the need for other sensors.

6.2.4 Object Tracking

Tracking objects in video is a widely studied problem in computer vision. Current systems often use a Kalman filter to model short term motion of objects [122, 201]. However, our results in Chapter 4 have demonstrated that our proposed methods outperform a Kalman filter for short term motion prediction up to 2 seconds. Using a model such as DTP or STED capable of modelling non-linear motion for motion prediction rather than a Kalman filter

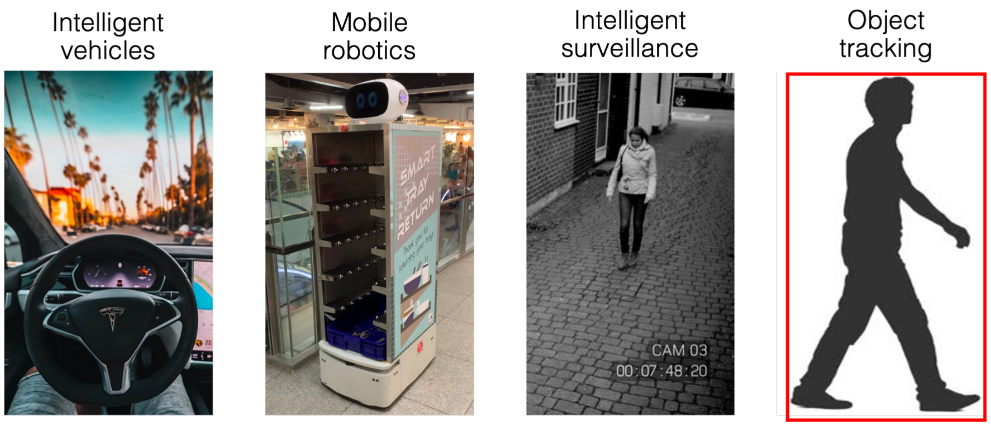


Figure 6.1: **Applications of our contributions.** Predicting the future location of an object in video has several application domains.

is likely to result in better object tracking, especially in scenarios where the object track is lost due to occlusions.

6.3 Future Work

In this section, we discuss works that have used our proposed methods and consider future research directions.

6.3.1 Works Using our Machine-Annotation Methods

Our single-camera machine annotation framework was extended by Le et al. [202]. The authors adopt our proposed machine-annotation method to generate additional training data for an object detection model rather than a trajectory prediction model. As part of the machine-annotation schedule involves using an object detector, the proposed method is repeated iteratively, using the re-trained detector with each pass of the unlabelled data. The authors also extend our machine-annotation framework with an additional step for recovering missed objects. After removing low-confidence detections and tracks shorter than a minimum length l, the remaining tracks are extended in both directions where the neighbouring frames contain detections with an IOU above a threshold. The work demonstrates the applicability of our proposed machine-annotation framework to other computer vision tasks.

Ansari et al. [155] use the Citywalks dataset to study transfer learning across egocentric trajectory forecasting datasets and propose a real-time egocentric forecasting model. The proposed model trained on Citywalks performs comparability to state-of-the-art methods on the FPL [31] dataset without any fine-tuning. The model performs poorly when evaluated on the JAAD [64] dataset without fine-tuning, as the JAAD dataset contains footage from a vehicle-mounted camera, which is different to the Citywalks dataset. However, fine-tuning with 15% of the JAAD dataset results on state-of-the-art performance. These encouraging results show promise for using our machine-annotated data from pre-training egocentric trajectory forecasting models.

Bouhsain et al. [203] use the Citywalks dataset to predict future object bounding boxes. The authors propose a model which is evaluated on both the human-annotated JAAD [64] datasets. The proposed model is discussed in more detail in the next subsection.

6.3.2 Works Using our Trajectory Forecasting Methods

Makansi et al. [154] propose a model for MOF and compare predictions with DTP and STED. Their model combines predicted vehicle egomotion with a novel reachability prior to predict the future locations of both vehicles and pedestrians using a vehicle-mounted camera. The reachability prior learns the most likely future locations of objects based on the scene's semantic segmentation. For example, vehicles are more likely to travel on roads than pavements. The proposed model outperforms the existing state-of-the-art. Although computing semantic segmentation can be computationally expensive, this is a common processing step in intelligent vehicles and could therefore be reused for other tasks, such as drivable area detection [204].

Ansari et al. [155] propose a model for MOF that uses only object bounding boxes as the model input. The emphasis in this work is on creating a fast and lightweight network that can run in real-time with minimal hardware requirements. The proposed model uses a similar encoder-decoder architecture to STED without the optical flow feature stream. The model uses a separate auto-encoder loss in addition to the future trajectory loss. After encoding the past bounding boxes using an LSTM, a separate decoder LSTM reconstructs the past bounding boxes in reverse order. This approach ensures that the encoder can reproduce the input. The authors train the model using our Citywalks dataset and demonstrate that the model generalises to the FPL dataset [31] *without fine-tuning.*

STED was used by Bouhsain et al. [203] who showed that performance comparable to our model in terms of displacement error was attainable on the Citywalks dataset using an LSTM model at half the runtime. However, the proposed model, SV-LSTM, does not perform as well as STED in terms of average and final IOU. The authors also train their SV-LSTM model using the JAAD dataset in a multi-task setting. SV-LSTM predicts both the future state of the pedestrian (crossing or not crossing) and their future bounding boxes. The multi-task formulation improves the performance of SV-LSTM for both tasks either in isolation, suggesting that the future state and future bounding box prediction tasks are complementary.

Poibrenski et al. [205] propose M2P3, a model for egocentric trajectory forecasting. M2P3 is an Conditional Variational Auto-Encoder (CVAE)-based model, meaning multiple future trajectories can be sampled from a single input trajectory. The authors show that M2P3 performs slightly better than DTP on the JAAD dataset when a single trajectory is sampled and considerably better when the best of 3 sampled trajectories is used. M2P3 is one of the first multi-modal approaches for egocentric trajectory forecasting, clearly demonstrating the advantages of such an approach.

Yin et at. [206] propose a MTN, a model for MOF from on board a moving vehicle, and compare with DTP and STED. MTN uses vehicle odometry information (where available), past target bounding boxes, and optical flow from within the target bounding boxes. Optical flow from the centre area of the camera view is also used to model the motion of the ego vehicle. All optical flow frames are downsampled using spatial average pooling and used as inputs to a transformer-based network. The proposed model is marginally better than STED in terms of MSE, while using considerably fewer model parameters.

Kesa et al. [207][1] extend the MOF problem formulation to Multiple Object Tracking and Forecasting (MOTF), where objects are tracked have their locations forecast by the same model. The authors propose JLA, a joint learning architecture for simultaneous tracking and forecasting. JLA uses a shared image feature embedding, where the features extracted from images are shared for both tracking and forecasting. Experimental results demonstrate that the multi-task framework improves performance on both tasks, i.e., the information from the tracking model is helpful for forecasting, and the information from the forecasting model is helpful for tracking. The proposed model reduces the number of identity switches by 33% - 47% on the MOTChallange benchmark datasets [46, 103].

6.3.3 Future Research Directions

Collecting new machine-annotated datasets. Our proposed dataset annotation strategies have made it quicker and cheaper to gather large datasets for trajectory forecasting. While our methods have been evaluated on human trajectory forecasting, our proposed annotation strategies and models are easily generalisable to any class of moving object. However, the performance of our methods for other object classes is currently unknown. Collecting datasets of other objects, such as vehicles, would enable us to study the differences in trajectory forecasting for different objects and open the possibility to explore new applications. For example, vehicle trajectory forecasting could enable

[1]The author of this thesis also contributed to this work.

intelligent traffic monitoring systems that control the flow of traffic by monitoring vehicles' current location and using accurate predictions about each vehicle's future trajectory.

Re-using our datasets for other tasks. The BDD-10k, WNMF and Citywalks datasets could be used for other tasks. For example, additional annotations could be added using a pose estimation algorithm to forecast human poses and trajectories. This addition would be particularly notable using the WNMF dataset, as to the best of our knowledge, there are no prior works on multi-camera pose forecasting.

Using the IOU as an objective function for MOF models. We use the AIOU and FIOU metrics to evaluate our MOF methods, but do not use these metrics as the objective function during training. Our MOF method, STED, is trained using the smooth ℓ_1 loss and FPL[31] is trained using the ℓ_2 loss. Recent work [208] has shown that optimizing the IOU metric directly improves the performance of object detection methods. A similar approach may be used for MOF methods by directly optimising for AIOU/FIOU.

Single-camera object interactions. Our MOF problem formulation considers the future bounding boxes of all identifiable objects in the scene. However, our proposed method does not consider the interactions between these objects during prediction. Existing trajectory forecasting works from a BEV show that modelling object interactions can be useful in trajectory prediction. Modelling object interactions is considerably more challenging from an egocentric perspective, as the impact of scale and perspective must also be considered. Nonetheless, effective modelling of the location of other objects is likely to improve the performance of MOF methods.

Multi-camera object interactions. Our proposed method for multi-target MCTF processes all targets in parallel but does not consider the interactions between targets. Similarly to a single camera view, modelling target interactions is likely to improve performance.

Multi-camera anomaly detection. Our trajectory forecasting models and datasets may also be used for the task of anomaly detection. Yao et al. [40, 93] proposed methods for anomaly detection using forecasting algorithms. If the observed trajectory deviated substantially from the predicted trajectory, this might indicate an anomalous event. In particular, the WNMF dataset provides suitable data for evaluating anomalous trajectory detection in a multi-camera environment. For example, if an MCTF model predicts with high confidence that a target will appear in a camera they do not appear, this may indicate that the individual is following an anomalous route.

MCTF for multi-camera tracking. Tracking a target across multiple non-overlapping camera views involves matching disjoint object trajectories. Our

MCTF framework may be used to aid in tracking, as the observed and predicted object track should be similar if the MCTF model is accurate. Matching tracks based on this similarity score may complement similarity scores using visual metrics and help distinguish between targets with similar visual appearance. Promising preliminary results on MCTF for multi-camera tracking using the WNMF dataset are shown in Appendix A.

Camera network independent MCTF methods. Our MCTF approach implicitly learns the camera network topology from training data. If the method is deployed on a new camera network, the model must be re-trained in order to learn the topology of the new camera network. Our proposed machine-aided annotation method makes data collection considerably easier than fully manual annotation; however, some human labour is still required. A model may generalise to other camera networks if multi-camera trajectories are projected to the world coordinate space without re-training.

6.4 Final Remarks

This thesis set out to expand on the state-of-the-art in human trajectory forecasting in video. We have created trajectory forecasting models at all stages of the pipeline: From data collection and annotation to model creation and evaluation. All of the code, datasets, and methods discussed in this thesis are available online at `https://github.com/olly-styles`, which we hope will facilitate future research towards increasingly more sophisticated trajectory forecasting systems.

Appendix A

Multi-Camera Trajectory Forecasting for Multi-Camera Tracking

One of the core applications of an MCTF system is multi-camera object tracking. Although a detailed study of this task is outside the scope of this thesis, here we present preliminary results of how an MCTF model may be deployed to facilitate multi-camera object tracking[1].

Methodology

We investigate the impact of using both an appearance-based person re-identification model in conjunction with our proposed MCTF model on cross camera tracking. The goal is to link two disjoint tracks, t_1 and t_2, that share the same identity. The tracks are always taken from different cameras, i.e., $c_{t_1} \neq c_{t_2}$.

Appearance features. We define $M_{app}(t)$ as the appearance features computed by a person re-identification model for track t. The appearance distance between a pair of tracks t_1 and t_2 are compared by computing the difference between their appearance features, $D_{app}(t_1, t_2) = F(M_{app}(t_1), M_{app}(t_2))$, where F measures the cosine distance between the appearance features, $M_{app}(t_1)$ and $M_{app}(t_2)$, of tracks t_1 and t_2.

Trajectory features. We use an MCTF model to compute a trajectory distance score, D_{traj}, between a pair of tracks, t_1 and t_2. D_{traj} is computed by comparing the future path predicted by our MCTF model with the observed tracks, $D_{traj}(t_1, t_2) = G(M_{traj}(t_1), M_{det}(t_2))$ where G measures the euclidean distance between the predicted and detected track.

[1]The work presented in this appendix was created in collaboration with Xufeng Lin.

Track association. Tracks from different cameras are associated by applying a pairwise bipartite matching algorithm to a distance matrix. We introduce a threshold η to avoid merging two tracks with a distance greater than η, i.e. when $D_{app}(k_1, k_2) > \eta$. We use a weighting parameter $\lambda \in [0, 1]$ to control the relative importance of the appearance distance score, D_{app}, and trajectory distance score, D_{traj}, to compute an overall distance score, $D = \lambda D_{app} + (1 - \lambda) D_{traj}$.

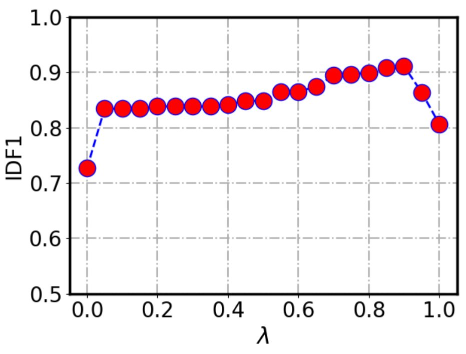
Performance evaluation

We annotate a subset of the WNMF dataset with identity labels and use an existing pre-trained person re-identification model to study the efficacy of joint track association using both appearance and trajectory features.

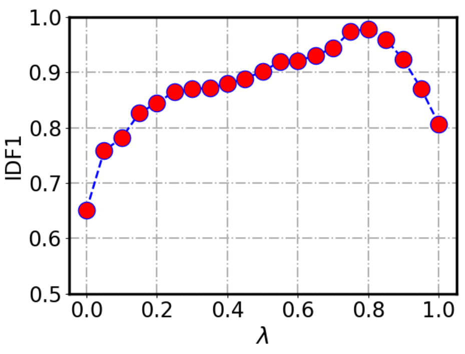
Dataset. The WNMF dataset consists of 1967 cross-camera trajectories. Identities are not annotated, and each cross-camera trajectory is not guaranteed to be a unique identity. We take trajectories from the first day of data collection and manually remove duplicate identities. Of the original 122 trajectories, 115 remain.

Experimental setup. We use Omni-Scale Network (OSNet) [209] to extract appearance features. The model is trained on the QMUL i-LIDS dataset (476 images, 119 identities and 4 non-overlapping cameras) [210] and MSMT17 dataset (126,441 images, 4101 identities and 15 cameras) [211]. We use our best-performing model (trajectory-tensors with a 3D-CNN backbone) as our MCTF model. We use the $IDF1$ metric to evaluate results, which is the ratio of correctly identified detections over the average number of ground-truth and computed detections [101].

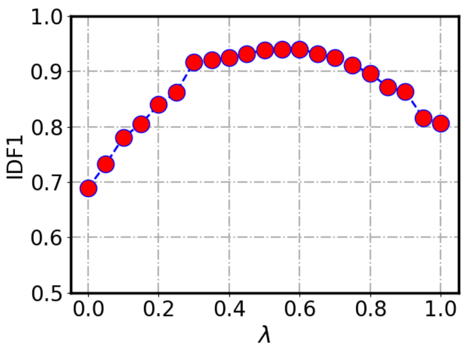
Results. The impact of varying λ on the tracking performance in terms of $IDF1$ scores is shown in Figure A.1. The highest $IDF1$ score is observed when $0 < \lambda < 1$, i.e., when both D_{app} and D_{tra} contribute to the overall distance score, D. In Table A.1, we show the $IDF1$ scores for different matching thresholds η and different weighting factors λ. For most cases, the highest $IDF1$ score is observed when $0 < \lambda < 1$. We see the best performance using the *when* problem formulation. This may be due to incorrect location predictions for the *where* formulation causing true matches to be unmatched.



(a) Using *which* model predictions to compute D_{traj}



(b) Using *when* model predictions to compute D_{traj}



(c) Using *where* model predictions to compute D_{traj}

Figure A.1: **Impact of varying λ on the $IDF1$ score.** We use a matching threshold of 0.40 and compare $IDF1$ scores using the 3 MCTF problem formulations to compute D_{traj}. We observe the highest scores using the *when* problem formulation.

Table A.1: Multi-camera tracking results with the *which*, *when*, and *where* MCTF models in terms of the $IDF1$ score on the WNMF dataset. The row with $\lambda^\star$ shows the highest achievable *IDF1* score and the value in parentheses is the value of $\lambda^\star$ when this highest $IDF1$ score is achieved. We highlight the highest $IDF1$ score in bold when a performance improvement is observed.

	λ \ η	0.15	0.20	0.25	0.30	0.35	0.40
Which	0	76.5	78.0	74.8	74.5	74.1	72.7
	1	59.9	68.9	75.8	80.3	80.9	80.7
	$\lambda^\star$	**78.4**(0.05)	**81.2**(0.05)	**83.7**(0.8)	**89.5**(0.85)	**92.3**(0.9)	**91.1**(0.9)
When	0	59.2	61.2	62.2	64.8	66.4	65.1
	1	59.9	68.9	75.8	80.3	80.9	80.7
	$\lambda^\star$	**66.2**(0.15)	**71.1**(0.35)	**83.6**(0.75)	**92.1**(0.85)	**97.0**(0.8)	**97.9**(0.8)
Where	0	57.6	59.4	63.0	65.3	66.3	68.9
	1	59.9	68.9	75.8	80.3	80.9	80.7
	$\lambda^\star$	59.9(1)	68.9(1)	**78.4**(0.85)	**85.7**(0.8)	**92.1**(0.65)	**94.0**(0.6)

Appendix B

Multi-Camera Trajectory Forecasting Model Architecture Details

Here, we provide details for the model architectures introduced in Chapter 5.

Table B.1: **RNN architecture.** Based on [2]

Block	Layer type	Kernel size	Stride	Output dimension
Encoder	Input	-	-	4
	FC + ReLU	-	-	128
	RNN + ReLU	-	-	128
Which decoder	FC + Sig.	-	-	15
When decoder	RNN + ReLU	-	-	128
	FC + Sig.	-	-	15
Where decoder	RNN + ReLU	-	-	128
	2D-TConv + ReLU	1	1	128
	2D-TConv + ReLU	4	2	128
	2D-TConv + Sig.	3	2	128

Table B.2: **1D-CNN architecture.** Based on [31]

Block	Layer type	Kernel size	Stride	Output dimension
Encoder	Input	-	-	4
	1D-Conv + ReLU	3	1	32
	1D-Conv + ReLU	3	1	64
	1D-Conv + ReLU	3	1	128
	1D-Conv + ReLU	3	1	128
Which decoder	FC + Sig.	-	-	15
When decoder	1D-TConv + ReLU	4	2	128
	1D-TConv + ReLU	4	2	128
	1D-TConv + ReLU	4	2	128
	1D-TConv + Sig.	4	2	128
Where decoder	1D-TConv + ReLU	4	2	128
	1D-TConv + ReLU	4	2	128
	1D-TConv + ReLU	4	2	128
	1D-TConv + ReLU	4	2	128
	2D-TConv + ReLU	1	1	128
	2D-TConv + ReLU	4	2	128
	2D-TConv + Sig.	3	2	128

Table B.3: **2D-1D CNN architecture.**

Block	Layer type	Kernel size	Stride	Output dimension
Encoder (9)	Input	-	-	15
	2D-Conv + ReLU	3	1	64
	2D Max pool	2	-	64
	2D-Conv + ReLU	2	1	128
	2D Max pool	2	1	128
	1D-Conv + ReLU	3	1	256
	1D-Max pool + ReLU	2	1	256
	1D-Conv + ReLU	3	1	256
Encoder (18)	Input	-	-	15
	2D-Conv + ReLU	5	1	64
	2D Max pool	2	-	64
	2D-Conv + ReLU	3	1	256
	2D Max pool	5	1	256
	1D-Conv + ReLU	3	1	256
	1D-Max pool + ReLU	2	1	256
	1D-Conv + ReLU	3	1	256
Encoder (27)	Input	-	-	15
	2D-Conv + ReLU	5	1	32
	2D Max pool	2	-	32
	2D-Conv + ReLU	3	1	128
	2D Max pool	2	1	128
	2D-Conv + ReLU	3	1	256
	2D Max pool	2	1	256
	1D-Conv + ReLU	3	1	256
	1D-Max pool + ReLU	2	1	256
	1D-Conv + ReLU	3	1	256
Which decoder	FC + Sig.	-	-	15
When decoder	1D-TConv + ReLU	4	2	256
	1D-TConv + ReLU	4	2	128
	1D-TConv + ReLU	4	2	64
	1D-TConv + ReLU	4	2	32
	1D-TConv + Sig.	1	1	15
Where decoder	1D-TConv + ReLU	7	1	256
	1D-TConv + ReLU	7	2	256
	1D-TConv + ReLU	6	3	256
	2D-TConv + ReLU	1	2	128
	2D-TConv + Sig.	4	2	64
	2D-TConv + Sig.	3	2	15

Table B.4: **3D CNN architecture.**

Block	Layer type	Kernel size	Stride	Output dimension
Encoder (9)	Input	-	-	15
	3D-Conv + ReLU	3	1	64
	3D Max pool	2	-	64
	3D-Conv + ReLU	3	1	256
	3D Max pool	1	-	256
Encoder (18)	Input	-	-	15
	3D-Conv + ReLU	3	1	64
	3D Max pool	1	-	64
	3D-Conv + ReLU	3	1	128
	3D Max pool	2	-	128
	3D-Conv + ReLU	3	1	256
	3D Max pool	1	-	256
Encoder (27)	Input	-	-	15
	3D-Conv + ReLU	1	1	64
	3D Max pool	1	-	64
	3D-Conv + ReLU	3	1	128
	3D Max pool	1	-	128
	3D-Conv + ReLU	3	1	256
	3D Max pool	2	-	256
	3D-Conv + ReLU	3	1	256
	3D Max pool	1	-	256
Which decoder	FC + Sig.	-	-	15
When decoder	1D-TConv + ReLU	4	2	256
	1D-TConv + ReLU	4	2	128
	1D-TConv + ReLU	4	2	64
	1D-TConv + ReLU	4	2	32
	1D-TConv + Sig.	1	1	15
Where decoder	3D-TConv + ReLU	5	2	256
	3D-TConv + ReLU	5	2	256
	3D-TConv + ReLU	5	2	128
	3D-TConv + Sig.	4	2	15

Table B.5: **CNN-GRU architecture.**

Block	Layer type	Kernel size	Stride	Output dimension
Encoder (9)	Input	-	-	15
	2D-Conv + ReLU	3	1	128
	2D Max pool	2	-	128
	2D-Conv + ReLU	2	1	256
	2D Max pool	2	1	256
	GRU + ReLU	-	-	256
Encoder (18)	Input	-	-	15
	2D-Conv + ReLU	5	1	64
	2D Max pool	2	-	64
	2D-Conv + ReLU	3	1	256
	2D Max pool	5	1	256
	GRU + ReLU	-	-	256
Encoder (27)	Input	-	-	15
	2D-Conv + ReLU	5	1	32
	2D Max pool	2	-	32
	2D-Conv + ReLU	3	1	128
	2D Max pool	2	1	128
	2D-Conv + ReLU	3	1	256
	2D Max pool	2	1	256
	GRU + ReLU	-	-	256
Which decoder	FC + Sig.	-	-	15
When decoder	GRU + ReLU	-	-	256
	FC + Sig.	-	-	15
Where decoder	GRU + ReLU	-	-	256
	2D-TConv + ReLU	1	2	128
	2D-TConv + Sig.	4	2	64
	2D-TConv + Sig.	3	2	15

Bibliography

[1] Olly Styles, Arun Ross, and Victor Sanchez. Forecasting pedestrian trajectory with machine-annotated training data. In *Intelligent Vehicles Symposium (IV)*, 2019.

[2] Olly Styles, Tanaya Guha, and Victor Sanchez. Multiple object forecasting: Predicting future object locations in diverse environments. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[3] Olly Styles, Tanaya Guha, Victor Sanchez, and Alex Kot. Multi-camera trajectory forecasting: Pedestrian trajectory prediction in a network of cameras. In *Computer Vision and Pattern Recognition Workshops (CVPR-W)*, 2020.

[4] Olly Styles, Tanaya Guha, and Victor Sanchez. Multi-camera trajectory forecasting with trajectory tensors. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[5] Jeff Hawkins and Sandra Blakeslee. *On intelligence*. Macmillan, 2004.

[6] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, 28:802–810, 2015.

[7] Kahkashan Afrin, Bimal Nepal, and Leslie Monplaisir. A data-driven framework to new product demand prediction: Integrating product differentiation and transfer learning approach. *Expert Systems with Applications*, 108:246–257, 2018.

[8] Takashi Kimoto, Kazuo Asakawa, Morio Yoda, and Masakazu Takeoka. Stock market prediction system with modular neural networks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 1990.

[9] Riku Turkki, Dmitrii Byckhov, Mikael Lundin, Jorma Isola, Stig Nordling, Panu E Kovanen, Clare Verrill, Karl von Smitten, Heikki Joensuu, Johan

Lundin, et al. Breast cancer outcome prediction with tumour tissue images and machine learning. *Breast Cancer Research and Treatment*, 177(1):41–52, 2019.

[10] Amir Rasouli. Deep learning for vision-based prediction: A survey. *arXiv preprint arXiv:2007.00095*, 2020.

[11] Michael Milford, Sam Anthony, and Walter Scheirer. Self-driving vehicles: key technical challenges and progress off the road. *IEEE Potentials*, 39(1):37–45, 2019.

[12] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J*, 3016:1–16, 2014.

[13] Markus Roth, Fabian Flohr, and Dariu M Gavrila. Driver and pedestrian awareness-based collision risk analysis. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 454–459. IEEE, 2016.

[14] Christoph G Keller, Thao Dang, Hans Fritz, Armin Joos, Clemens Rabe, and Dariu M Gavrila. Active pedestrian safety by automatic braking and evasive steering. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1292–1304, 2011.

[15] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[16] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Tracking by prediction: A deep generative model for mutli-person localisation and tracking. In *Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[17] Hiroaki Minoura, Ryo Yonetani, Mai Nishimura, and Yoshitaka Ushiku. Crowd density forecasting by modeling patch-based dynamics. *IEEE Robotics and Automation Letters*, 6(2):287–294, 2020.

[18] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[19] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Agreeing to cross: How drivers and pedestrians communicate. In *Intelligent Vehicles Symposium*, pages 264–269. IEEE, 2017.

[20] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Understanding pedestrian behavior in complex traffic scenes. *IEEE Transactions on Intelligent Vehicles*, 3(1):61–70, 2018.

[21] Sarah Schmidt and B Färber. Pedestrians at the kerb–recognising the action intentions of humans. *Transportation Research Part F: Traffic Psychology and Behaviour*, 12(4), 2009.

[22] Tianjiao Wang, Jianping Wu, Pengjun Zheng, and Mike McDonald. Study of pedestrians' gap acceptance behavior when they jaywalk outside crossing facilities. In *Intelligent Transportation Systems (ITSC)*, pages 1295–1300. IEEE, 2010.

[23] Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Towards reaching human performance in pedestrian detection. *Transactions on Pattern Analysis and Machine Intelligence*, 40(4):973–986, 2018.

[24] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[25] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE International Conference on Computer Vision*, pages 843–852, 2017.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[27] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282, 1995.

[28] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Transactions on Graphics (TOG)*, 25(3):1160–1168, 2006.

[29] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] Qi Dang, Jianqin Yin, Bin Wang, and Wenqing Zheng. Deep learning based 2d human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676, 2019.

[31] Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. Future person localization in first-person videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[32] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Computer Vision and Pattern Recognition*, 2016.

[33] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] Irtiza Hasan, Francesco Setti, Theodore Tsesmelis, Vasileios Belagiannis, Sikandar Amin, Alessio Del Bue, Marco Cristani, and Fabio Galasso. Forecasting people trajectories and head poses by jointly reasoning on tracklets and vislets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4):1267–1278, 2019.

[35] Kuo-Hao Zeng, William B Shen, De-An Huang, Min Sun, and Juan Carlos Niebles. Visual forecasting by imitating dynamics in natural sequences. In *IEEE International Conference on Computer Vision*, pages 2999–3008, 2017.

[36] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection–a new baseline. In *Computer Vision and Pattern Recognition*, pages 6536–6545, 2018.

[37] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual motion gan for future-flow embedded video prediction. In *IEEE International Conference on Computer Vision*, pages 1744–1752, 2017.

[38] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[39] Ruben Villegas, Dumitru Erhan, Honglak Lee, et al. Hierarchical long-term video prediction without supervision. In *International Conference on Machine Learning*, pages 6038–6046. PMLR, 2018.

[40] Yu Yao, Xizi Wang, Mingze Xu, Zelin Pu, Ella Atkins, and David Crandall. When, where, and what? a new dataset for anomaly detection in driving videos. *arXiv preprint arXiv:2004.03044*, 2020.

[41] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[42] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *IEEE International Conference on Computer Vision*, pages 648–657, 2017.

[43] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[44] Adam Terwilliger, Garrick Brazil, and Xiaoming Liu. Recurrent flow-guided semantic forecasting. In *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019.

[45] Pauline Luc, Camille Couprie, Yann Lecun, and Jakob Verbeek. Predicting future instance segmentation by forecasting convolutional features. In *European Conference on Computer Vision (ECCV)*, pages 584–599, 2018.

[46] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[47] Tewodros Legesse Munea, Yalew Zelalem Jembre, Halefom Tekle Weldegebriel, Longbiao Chen, Chenxi Huang, and Chenhui Yang. The progress of human pose estimation: a survey and taxonomy of models applied in 2d human pose estimation. *IEEE Access*, 8:133330–133348, 2020.

[48] Karttikeya Mangalam, Ehsan Adeli, Kuan-Hui Lee, Adrien Gaidon, and Juan Carlos Niebles. Disentangling human dynamics for pedestrian locomotion forecasting with noisy supervision. In *Winter Conference on Applications of Computer Vision*, pages 2784–2793, 2020.

[49] Sebastian Agethen, Hu-Cheng Lee, and Winston H Hsu. Anticipation of human actions with pose-based fine-grained representations. In *Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[50] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Computer Vision and Pattern Recognition*, pages 2891–2900, 2017.

[51] Tomoyuki Suzuki, Hirokatsu Kataoka, Yoshimitsu Aoki, and Yutaka Satoh. Anticipating traffic accidents with adaptive loss and large-scale incident db. In *Computer Vision and Pattern Recognition*, pages 3521–3529, 2018.

[52] Ashesh Jain, Avi Singh, Hema S Koppula, Shane Soh, and Ashutosh Saxena. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *Robotics and Automation (ICRA)*, pages 3118–3125. IEEE, 2016.

[53] Youngjin Yoo, Lisa W Tang, Tom Brosch, David KB Li, Luanne Metz, Anthony Traboulsee, and Roger Tam. Deep learning of brain lesion patterns for predicting future disease activity in patients with early symptoms of multiple sclerosis. In *Deep Learning and Data Labeling for Medical Applications*, pages 86–94. Springer, 2016.

[54] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *Computer Vision and Pattern Recognition*, pages 98–106, 2016.

[55] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997.

[56] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, pages 41–48. ACM, 2009.

[57] Mohammad Sadegh Aliakbarian, F Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. Encouraging lstms to anticipate actions very early. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[58] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[59] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *IEEE International Conference on Computer Vision*, pages 3192–3199, 2013.

[60] MS Ryoo, Chia-Chih Chen, JK Aggarwal, and Amit Roy-Chowdhury. An overview of contest on semantic description of human activities (sdha) 2010. In *Recognizing Patterns in Signals, Speech, Images and Videos*, pages 270–285. Springer, 2010.

[61] Yu Yao, Mingze Xu, Chiho Choi, David J Crandall, Ella M Atkins, and Behzad Dariush. Egocentric vision-based future vehicle localization for intelligent driving assistance systems. In *International Conference on Robotics and Automation*, 2019.

[62] Alexandre Alahi, Vignesh Ramanathan, and Li Fei-Fei. Socially-aware large-scale crowd forecasting. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[63] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European Conference on Computer Vision*, pages 549–565. Springer, 2016.

[64] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In *International Conference on Computer Vision Workshop (ICCV-w)*, pages 206–213, 2017.

[65] Vipin Kumar Kukkala, Jordan Tunnell, Sudeep Pasricha, and Thomas Bradley. Advanced driver-assistance systems: A path toward autonomous vehicles. *IEEE Consumer Electronics Magazine*, 7(5):18–25, 2018.

[66] Shuai Tang, Mani Golparvar-Fard, Milind Naphade, and Murali M Gopalakrishna. Video-based motion trajectory forecasting method for proactive construction safety monitoring systems. *Journal of Computing in Civil Engineering*, 34(6):04020041, 2020.

[67] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *Transactions on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2016.

[68] Sandro Hauri, Nemanja Djuric, Vladan Radosavljevic, and Slobodan Vucetic. Multi-modal trajectory prediction of nba players. *Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[69] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *IEEE International Conference on Computer Vision*, pages 300–311, 2017.

[70] Yuan Liu, Ruoteng Li, Yu Cheng, Robby T Tan, and Xiubao Sui. Object tracking using spatio-temporal networks for future prediction location. *European Conference on Computer Vision*, 2020.

[71] Pim Dijt and Pascal Mettes. Trajectory prediction network for future anticipation of ships. In *International Conference on Multimedia Retrieval*, pages 73–81, 2020.

[72] Tsubasa Hirakawa, Takayoshi Yamashita, Toru Tamaki, Hironobu Fujiyoshi, Yuta Umezu, Ichiro Takeuchi, Sakiko Matsumoto, and Ken Yoda. Can AI predict animal movements? Filling gaps in animal trajectories using inverse reinforcement learning. *Ecosphere*, 9(10), 2018.

[73] 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 *Computer Vision and Pattern Recognition*, 2017.

[74] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[75] Ashraf Elnagar. Prediction of moving objects in dynamic environments using kalman filters. In *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No. 01EX515)*, pages 414–419. IEEE, 2001.

[76] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *International Conference on Computer Vision (ICCV)*, volume 9, pages 261–268, 2009.

[77] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *Eurpoean Conference on Computer Vision*. Springer, 2012.

[78] Kota Yamaguchi, Alexander C Berg, Luis E Ortiz, and Tamara L Berg. Who are you with and where are you going? In *Computer Vision and Pattern Recognition*, pages 1345–1352. IEEE, 2011.

[79] Chiho Choi and Behzad Dariush. Looking to relations for future trajectory forecast. *International Conference on Computer Vision (ICCV)*, pages 921–930, 2019.

[80] Igor Gilitschenski, Guy Rosman, Arjun Gupta, Sertac Karaman, and Daniela Rus. Deep context maps: Agent trajectory prediction using location-specific latent maps. *Robotics and Automation Letters*, 2020.

[81] Amir Sadeghian, Ferdinand Legros, Maxime Voisin, Ricky Vesel, Alexandre Alahi, and Silvio Savarese. Car-net: Clairvoyant attentive recurrent network. In *European Conference on Computer Vision (ECCV)*, 2018.

[82] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[83] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Pedestrian behavior understanding and prediction with deep neural networks. In *Eurpoean Conference on Computer Vision*, pages 263–279. Springer, 2016.

[84] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Computer Vision and Pattern Recognition*, 2018.

[85] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision (ECCV)*. Springer, 2020.

[86] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5(2), 2020.

[87] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10508–10518, 2020.

[88] Nicolas Schneider and Dariu M Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *German Conference on Pattern Recognition*, pages 174–183. Springer, 2013.

[89] Julian Francisco Pieter Kooij, Nicolas Schneider, Fabian Flohr, and Dariu M Gavrila. Context-based pedestrian path prediction. In *European Conference on Computer Vision (ECCV)*, pages 618–633. Springer, 2014.

[90] Julian FP Kooij, Fabian Flohr, Ewoud AI Pool, and Dariu M Gavrila. Context-based path prediction for targets with switching dynamics. *International Journal of Computer Vision*, pages 1–24, 2018.

[91] Mirko Meuter, Uri Iurgel, Su-Birm Park, and Anton Kummert. The unscented kalman filter for pedestrian tracking from a moving host. In *Intelligent Vehicles Symposium*. IEEE, 2008.

[92] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. In *Computer Vision and Pattern Recognition*, 2018.

[93] Yu Yao, Mingze Xu, Yuchen Wang, David J Crandall, and Ella M Atkins. Unsupervised traffic accident detection in first-person videos. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 273–280. IEEE, 2019.

[94] Hyun Soo Park, Jyh-Jing Hwang, Yedong Niu, and Jianbo Shi. Egocentric future localization. In *Computer Vision and Pattern Recognition*, pages 4697–4705, 2016.

[95] Xiaogang Wang. Intelligent multi-camera video surveillance: A review. *Pattern Recognition Letters*, 34(1), 2013.

[96] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer Graphics Forum*, 2007.

[97] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. In *Computer Vision and Pattern Recognition*, pages 3457–3464. IEEE, 2011.

[98] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *Computer Vision and Pattern Recognition*, pages 3153–3160. IEEE, 2011.

[99] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[100] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *2012 Computer Vision and Pattern Recognition*, pages 2871–2878. IEEE, 2012.

[101] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision (ECCV)*, pages 17–35. Springer, 2016.

[102] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[103] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*, 2020.

[104] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

[105] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Computer Vision and Pattern Recognition*, pages 2636–2645, 2020.

[106] Jake Satisky. *A Duke study recorded thousands of students' faces. Now they're being used all over the world*, 2019 (accessed 20th February, 2020). https://www.dukechronicle.com/article/2019/06/duke-university-facial-recognition-data-set-study-surveillance-video-students-china-uyghur.

[107] Alina Bialkowski, Simon Denman, Sridha Sridharan, Clinton Fookes, and Patrick Lucey. A database for person re-identification in multi-camera surveillance networks. In *2012 International Conference on Digital Image Computing Techniques and Applications (DICTA)*, pages 1–8. IEEE, 2012.

[108] Shaogang Gong, Marco Cristani, Shuicheng Yan, and Chen Change Loy. *Person re-identification*. Springer, 2014.

[109] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005.*, volume 1, pages 886–893. IEEE, 2005.

[110] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition*, 2009.

[111] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Citypersons: A diverse dataset for pedestrian detection. In *Conference on Computer Vision and Pattern Recognition*, 2017.

[112] Markus Braun, Sebastian Krebs, Fabian Flohr, and Dariu M Gavrila. Eurocity persons: A novel benchmark for person detection in traffic scenes. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[113] Wei Liu, Shengcai Liao, Weiqiang Ren, Weidong Hu, and Yinan Yu. High-level semantic feature detection: A new perspective for pedestrian detection. In *Conference on Computer Vision and Pattern Recognition*, pages 5187–5196, 2019.

[114] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram, and Ling Shao. Generalizable pedestrian detection: The elephant in the room. In *Computer Vision and Pattern Recognition (CVPR)*, pages 11328–11337, 2021.

[115] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018.

[116] Kai Briechle and Uwe D Hanebeck. Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*, volume 4387, pages 95–102. International Society for Optics and Photonics, 2001.

[117] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence*. Vancouver, British Columbia, 1981.

[118] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[119] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI Conference on Artificial Intelligence*, 2020.

[120] Axel Sauer, Elie Aljalbout, Sami Haddadin, and Machine Intelligence. Tracking holistic object representations. In *British Machine Vision Conference (BMVC)*, 2019.

[121] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[122] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and real-time tracking with a deep association metric. In *International Conference on Image Processing (ICIP)*, 2017.

[123] Kuan Fang, Yu Xiang, Xiaocheng Li, and Silvio Savarese. Recurrent autoregressive networks for online multi-object tracking. In *IEEE Winter*

Conference on Applications of Computer Vision (WACV), pages 466–475. IEEE, 2018.

[124] Liang Zheng, Yi Yang, and Alexander G Hauptmann. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984*, 2016.

[125] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *International Conference on Computer Vision (ICCV)*, 2015.

[126] Charu Sharma, Siddhant R Kapil, and David Chapman. Person re-identification with a locally aware transformer. *arXiv preprint arXiv:2106.03720*, 2021.

[127] Ruijie Quan, Xuanyi Dong, Yu Wu, Linchao Zhu, and Yi Yang. Auto-reid: Searching for a part-aware convnet for person re-identification. *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[128] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *Computer Vision and Pattern Recognition Workshops (CVPR-w)*, 2019.

[129] Guangcong Wang, Jianhuang Lai, Peigen Huang, and Xiaohua Xie. Spatial-temporal person re-identification. In *AAAI Conference on Artificial Intelligence*, 2019.

[130] Samvit Jain, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, and Joseph Gonzalez. Scaling video analytics systems to large camera deployments. In *International Workshop on Mobile Computing Systems and Applications*, 2019.

[131] Hung-Min Hsu, Tsung-Wei Huang, Gaoang Wang, Jiarui Cai, Zhichao Lei, and Jenq-Neng Hwang. Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models. In *Computer Vision and Pattern Recognition Workshops (CVPR-w)*, 2019.

[132] Octavia Camps, Mengran Gou, Tom Hebble, Srikrishna Karanam, Oliver Lehmann, Yang Li, Richard J Radke, Ziyan Wu, and Fei Xiong. From the lab to the real world: Re-identification in an airport camera network. *Transactions on Circuits and Systems for Video Technology*, 27(3), 2016.

[133] Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Person reidentification in a distributed camera network framework. *Transactions on Cybernetics*, 47(11), 2016.

[134] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009.

[135] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *European conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[136] Tewodros A Biresaw, Tahir Nawaz, James Ferryman, and Anthony I Dell. Vitbat: Video tracking and behavior annotation tool. In *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 295–301. IEEE, 2016.

[137] Antonio Torralba, Bryan C Russell, and Jenny Yuen. Labelme: Online image annotation and applications. *Proceedings of the IEEE*, 98(8):1467–1484, 2010.

[138] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *Computer Vision and Pattern Recognition*, pages 859–868, 2018.

[139] Hubert Lin, Paul Upchurch, and Kavita Bala. Block annotation: Better image annotation with sub-image decomposition. In *IEEE International Conference on Computer Vision*, pages 5290–5300, 2019.

[140] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Eurpoean Conference on Computer Vision*. Springer, 2016.

[141] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.

[142] Joon Son Chung and Andrew Zisserman. Out of time: automated lip sync in the wild. In *Asian Conference on Computer Vision*, pages 251–263. Springer, 2016.

[143] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Computer Vision and Pattern Recognition*, pages 5505–5514, 2018.

[144] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*, 2015.

[145] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Computer Vision and Pattern Recognition*, pages 1058–1067, 2017.

[146] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[147] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *IEEE International Conference on Computer Vision*, pages 5898–5906, 2017.

[148] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[149] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization. *Advances in Neural Information Processing Systems*, 2018.

[150] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[151] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *International Conference on Computer Vision (ICCV)*, 2015.

[152] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv:1604.07316*, 2016.

[153] Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In *IEEE International Conference on Computer Vision*, pages 609–617, 2017.

[154] Osama Makansi, Ozgun Cicek, Kevin Buchicchio, and Thomas Brox. Multimodal future localization and emergence prediction for objects in egocentric view with a reachability prior. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4354–4363, 2020.

[155] Junaid Ahmed Ansari and Brojeshwar Bhowmick. Simple means faster: Real-time human motion forecasting in monocular first person videos

on cpu. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 10319–10326. IEEE, 2020.

[156] Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. How far are we from solving pedestrian detection? In *Computer Vision and Pattern Recognition*, 2016.

[157] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *European Conference on Computer Vision (ECCV)*, 2014.

[158] Eike Rehder and Horst Kloeden. Goal-directed pedestrian prediction. In *Computer Vision and Pattern Recognition workshop*, 2015.

[159] Hao Zhu, Ka-Veng Yuen, Lyudmila Mihaylova, and Henry Leung. Overview of environment perception for intelligent vehicles. *Transactions on Intelligent Transportation Systems*, 18(10), 2017.

[160] Berthold Färber. Communication and communication problems between autonomous vehicles and human drivers. In *Autonomous Driving*, pages 125–144. Springer, 2016.

[161] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv:1805.04687*, 2018.

[162] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018.

[163] 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*, 2015.

[164] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[165] Ross Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[166] Ya Tian, Yong Liu, and Jindong Tan. Wearable navigation system for the blind people in dynamic environments. In *2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, pages 153–158. IEEE, 2013.

[167] Tung-Sing Leung and Gerard Medioni. Visual navigation aid for the blind in dynamic environments. In *Computer Vision and Pattern Recognition Workshops*, pages 565–572, 2014.

[168] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *International Conference on Computer Vision*, 2017.

[169] Ricardo Sanchez-Matilla, Fabio Poiesi, and Andrea Cavallaro. Online multi-target tracking with strong and weak detections. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[170] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. Mars: A video benchmark for large-scale person re-identification. In *European Conference on Computer Vision*, pages 868–884. Springer, 2016.

[171] NEIL Owens, C Harris, and C Stennett. Hawk-eye tennis system. In *2003 International Conference on Visual Information Engineering VIE 2003*, pages 182–185. IET, 2003.

[172] Milind Naphade, Zheng Tang, Ming-Ching Chang, David C Anastasiu, Anuj Sharma, Rama Chellappa, Shuo Wang, Pranamesh Chakraborty, Tingting Huang, Jenq-Neng Hwang, et al. The 2019 ai city challenge. In *Computer Vision and Pattern Recognition Workshops (CVPR-w)*, volume 8, 2019.

[173] Jeremy Onyan. Why accurate time is essential for video surveillance and security programs. Technical report, Orolia, 2019.

[174] Prarthana Shrstha, Mauro Barbieri, and Hans Weda. Synchronization of multi-camera video recordings based on audio. In *International Conference on Multimedia*, pages 545–548, 2007.

[175] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Computer Vision and Pattern Recognition*, pages 5203–5212, 2020.

[176] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *Transactions on Pattern Analysis and Machine Intelligence*, 30(2), 2008.

[177] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, 2016.

[178] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014.

[179] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Eurpoean Conference on Computer Vision*. Springer, 2016.

[180] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.

[181] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[182] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[183] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[184] 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. *Empiricial Methods in Natural Language Processing*, 2014.

[185] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *European Conference on Computer Vision (ECCV)*, 2018.

[186] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. In *Computer Vision and Pattern Recognition*, 2017.

[187] Ergys Ristani and Carlo Tomasi. Features for multi-target multi-camera tracking and re-identification. In *Computer Vision and Pattern Recognition*, 2018.

[188] Chen Long, Ai Haizhou, Zhuang Zijie, and Shang Chong. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *International Conference on Multimedia and Expo*, volume 5, page 8, 2018.

[189] Hilke Kieritz, Stefan Becker, Wolfgang Hübner, and Michael Arens. Online multi-person tracking using integral channel features. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 122–130. IEEE, 2016.

[190] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[191] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[192] Shruti Vyas, Yogesh S Rawat, and Mubarak Shah. Multi-view action recognition using cross-view video prediction. In *European Conference on Computer Vision*, 2020.

[193] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[194] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4), 2019.

[195] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *European Conference on Computer Vision (ECCV)*, 2018.

[196] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet. In *Computer Vision and Pattern Recognition*, 2018.

[197] *Lexus and Toyota will make automated braking standard on nearly every model and trim level by end of 2017*, 2016 (accessed 27th August 2021). https://pressroom.toyota.com/lexus-toyota-automated-braking-standard-2017/.

[198] Puneet Kohli and Anjali Chadha. Enabling pedestrian safety using computer vision techniques: A case study of the 2018 uber inc. self-driving car crash. *Advances in Information and Communication: Proceedings of the Future of Information and Communication Conference (FICC)*, 2019.

[199] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312, 2017.

[200] Junqing Shi, Fengxiang Qiao, Qing Li, Lei Yu, and Yongju Hu. Application and evaluation of the reinforcement learning approach to eco-driving at intersections under infrastructure-to-vehicle communications. *Transportation Research Record*, 2672(25):89–98, 2018.

[201] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FairMOT: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2020.

[202] Trung-Nghia Le, Akihiro Sugimoto, Shintaro Ono, and Hiroshi Kawasaki. Toward interactive self-annotation for video object bounding box: Recurrent self-learning and hierarchical annotation based framework. In *Winter Conference on Applications of Computer Vision*, pages 3231–3240, 2020.

[203] Smail Ait Bouhsain, Saeed Saadatnejad, and Alexandre Alahi. Pedestrian intention prediction: A multi-task perspective. *arXiv preprint arXiv:2010.10270*, 2020.

[204] Annika Meyer, N Ole Salscheider, Piotr F Orzechowski, and Christoph Stiller. Deep semantic lane segmentation for mapless driving. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 869–875. IEEE, 2018.

[205] Atanas Poibrenski, Matthias Klusch, Igor Vozniak, and Christian Müller. M2p3: multimodal multi-pedestrian path prediction by self-driving cars with egocentric vision. In *Annual ACM Symposium on Applied Computing*, pages 190–197, 2020.

[206] Ziyi Yin, Ruijin Liu, Zhiliang Xiong, and Zejian Yuan. Multimodal transformer network for pedestrian trajectory prediction. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2021.

[207] Oluwafunmilola Kesa, Olly Styles, and Victor Sanchez. Joint learning architecture for multiple object tracking and trajectory forecasting. *arXiv preprint arXiv:2108.10543*, 2021.

[208] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.

[209] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *International Conference on Computer Vision*, 2019.

[210] Bryan James Prosser, Wei-Shi Zheng, Shaogang Gong, Tao Xiang, Q Mary, et al. Person re-identification by support vector ranking. In *British Machine Vision Conference*, 2010.

[211] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *Conference on Computer Vision and Pattern Recognition*, 2018.