Tracking Multiple Targets in Warehouses

PhD Thesis
by

Syeda Fouzia Noor

Submitted in Partial Fulfillment of the
Requirements of the
PhD Program

Department of Electrical and Electronic Engineering
Auckland University of Technology
New Zealand
December 2019

For my family, especially my children, Nusayba and Mustafa.

Abstract

Optimising underlying processes and increasing work safety for employees are two key concerns for the warehouse industry. Common technologies for warehouses are warehouse management systems (WMS) and radio frequency identifications (RFID). WMS aims at handling warehouse operations in an improved way and to store relevant inventory information. RFID enables products to be automatically identified, resulting in cost savings through shorter handling timelines. Despite the benefits associated with WMS or RFID, there is still a margin for improvement in the commonly employed technologies used in the warehouse industry.

Computer vision algorithms have been used to solve complex video-based analytical solutions, but their potential in warehouse surveillance applications is not yet fully explored. The recent use of deep learning-based architectures has brought a revolution in the global information technology industry. When supplied with a bulk of training data and with the availability of fast computational resources like graphical processing units (GPUs), these models have brought wonders in improving detection and localisation accuracy of objects of interest. These models have successfully been used in solving a wide diversity of computer vision tasks.

This project aims at exploring and studying the multiple object tracking (MOT) problem in the context of a typical warehouse environment. Industrial vehicles (i.e. forklift trucks) and pedestrians are detected and tracked in complex warehouse environment featuring challenges like occlusions, clutter or illuminations variations. In industrial warehouses, a robust multiple object tracking framework could provide useful information on forklift truck and pedestrian movements, to be used for processes improvement. It can help in enhancing situational awareness in warehouses and increase safety levels of the pedestrians working there.

This thesis describes the implementation, evaluation and analysis designed and selected for model-based computer vision algorithms for tracking multiple targets. Owing to the success of deep feature-based tracking mechanisms, we train, evaluate and explore their potential in a warehouse environment using a *detection based tracking* approach. Detection to track association methods are revisited, with a novel proposed track association algorithm. The use of various hand-crafted and deep feature-based methods and their limitations in different tracking mechanisms are studied in detail. A novel composite feature-based ensemble tracker is also proposed, leading to robust visual tracking results.

This thesis also reports about an experimental comparison for robust multiple object trackers for warehouse environments, discussing model-based versus deep learning-based methods, thus highlighting their limitations and future research directions.

Keywords: Computer Vision, Deep Learning, Multiple Object Tracking, Safety, Warehouses

Acknowledgments

First of all, I thank my primary supervisor, Professor Reinhard Klette. His consistent support and kind encouragement throughout this Ph.D. study helped me to accomplish my project goals and kept me going for my work. I wish the best for him, in his future endeavours, and always extend my warm regards. He really has been an inspiration for me in my academic journey.

My special thanks go to my other supervisor, Dr. Waqar Khan, for assisting me in my iterative thesis revisions, providing suggestions and giving me his precious time, out of his busy work routine. I really appreciate his time and concerns. I also thank my co-supervisor, Prof. Fay Huang, for her support.

I thank Callaghan Innovation for a scholarship, thus providing me the opportunity to undertake an interesting and challenging research project. The scholarship helped me to explore and unwind my hidden technical research abilities and to polish them even further.

I am very much grateful to Crown Equipment Limited, Auckland, New Zealand, for providing continuous support and assistance during this project. The team around Kashyap Chandrasekar really helped me to understand the studied research problem in the context of a warehouse environment. I deeply acknowledge their support, especially for the data collection phases at Crown Warehousing premises. I am especially grateful to Mark Bell; the precious worthwhile interactions with Mark during our scheduled meetings at Crown really helped me in my way to proceed through the project.

I thank Kashyap Chandrasekar for helping and guiding me throughout this project and keeping me updated about the Crown policies and procedures. I extend my warm thanks to Dr. Junli Tao; I was always able to look forward to her, asking my questions and queries when faced with a new research challenge. I wish all the best for Jacqueline Lewis; she has been really kind and helpful for me in all these years. I thank her deeply for all her administrative and other support she extended to me when I needed it the most. I also say thanks to all the other technical and administrative staff at Crown for supporting me in whatever situation during the course of this project.

I am very much thankful to my family, my husband Ahsan for being of continuous support in good and hard times, my parents, and my children Nusayba and

Mustafa for being very calm kids.

Syeda Fouzia
Auckland
24-10-2019

List of Symbols

Images and tracking variables

I	An image
C	A monocular camera
A_m	m number of A multiple objects
i	Frame index
S_i	State vector to represent locations
l_i^m	Location of m_{th} object in i_{th} frame
$l_{s:f}^j = (l_s^j, ..., l_f^j)$	Initial to final sequential locations
$O_i = (o_i^1, o_i^2, o_i^3, ..., o_i^m)$	Collected observations
gt	Ground truth location
(x, y)	2D centroid location of the target
(x, y, z)	World coordinates
x_k and y_k	Co-ordinates of centroid of bounding box
v_k	Velocity of the window center
a_k	Acceleration of target
d	Euclidean distance d

Gaussian mixture model variables

$P(X_t)$	Multivariate probability density function
k	Number of Gaussian components
$\sigma_{i,t}^2$	Variance of RGB channel
α and ρ	Learning rates
T	Threshold for Gaussian assignment
$w_{i,t}$	Weight of i_{th} Gaussian distribution
$\mu_{i,t}$	Gaussian mean

Pixel saliency map variables

N	Number of superpixel nodes
σ^2	Fall-off rate
α	Fitting constraint parameter

Kalman filter variables

x_{k+1}^p	Predicted state vector
A	State transition matrix
w	Gaussian distribution noise
Q	Variance of Gaussian distribution noise
P_{k+1}^p	Predicted state error co-variance matrix
H	Measurement matrix
v	Measurement noise
R	Co-variance matrix
z_{k+1}	Measurement residual vector
S	Residual variance matrix

Saliency guided data association variables

bdist	Bhattacharyya distance
γ	Aspect ratio of bounding boxes
τ_m	Mahalanobis-distance threshold
τ_{bdist}	Bhattacharyya-distance threshold
$\mathbf{d}^1(i,j)$	Mahalanobis distance
$S(x,y)$	Degree of saliency
I_μ^*	Mean image feature
I_σ^*	Gaussian blurred image
$\mathbf{p}_u(x)$	ROI colour or saliency distribution
δ	Kronecker function
k	Kernel function
N_p	Number of pixels in the ROI
Λ	A hyperparameter for cost estimation

MOT evaluation metrics

MOTA	Multiple object tracking accuracy
MOTP	Multiple object tracking precision
FN_t	Number of false-negatives or misses at t
FP_t	Number of false-positives at t
c_t	Total number of matches in frame t
$d_{t,i}$	Bounding box overlap of target with ground truth

Multi-cue correlation-filter variables

f_{sal}	Feature maps for saliency
$f_{\text{HOG-CNN}}$	Feature maps HOG and CNN
g_{sal}, g_{feat}	Learned correlation filters for saliency and other features
R_{sal}, R_{feat}	Saliency response and other feature response maps
Sal_t	Vectorised saliency maps
$\mathcal{C}_{sim}$	Cosine similarity
$w\,(t)$	Saliency weight frame t
M	Maximum saliency contribution

DCF variables

$\mathbf{x}(m,n)$	Shifted training samples M and N dimensions
$\mathbf{w}$	A weighted correlation filter
$\mathbf{X}$	Data matrix for all circular shifts
λ	Regularisation parameter
$\mathbf{w}_d$	Learned filter weight on the d-th channel
$\odot$	Hadamard or element-wise product
DFT	Discrete Fourier transform for the vector
$\mathbf{z}$	ROI patch of the same size as $\mathbf{x}$
η	Learning rate
$\hat{\mathbf{w_d}}^*$	Online filter update weight
T_i	Hypothesis tracker
$B^t_{T_i}$	Tracker i's bounding box

$O^t_{T_i,T_j}$	Overlap ratio tracker pair
$Score^t_{pair}\left(T_i\right)$	Tracker pair robustness score
$M^t_{T_i}$	Mean of the overlap ratios
$F^t_{T_i}$	Fluctuation of tracker overlap ratios
σ_{E_i}	Average of the width and height of bounding box
$R^t\left(T_i\right)$	Tracker′s robustness degree
$O\left(T, GT\right)$	Overlap between ground truth and result
AUC	Area under curve
OPE	One-pass evaluation
SRE	Spatial robustness evaluation
TRE	Temporal robustness evaluation

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Syeda Fouzia
04-01-2020

Fouzia

Contents

Chapter 1

Introduction

This introductory chapter reports about motivations and challenges for the presented study of multiple target tracking in warehouses. The chapter also summarises contributions reported in the thesis and concludes with an outline of the structure of the work.

1.1 Objectives

This study aims at detecting, localising and tracking multiple targets of interest in warehouses. Multiple targets of interest are *pedestrians* and *forklift trucks*, sharing the same warehouse premises. Applied technologies are various computer vision and *deep learning* algorithms, designed for accurate recognition and tracking of objects of interest.

Computer vision is commonly used for target recognition and tracking, but its potential is not yet fully explored for solving object tracking problems in the context of warehouse environments. With the availability of better acquisition systems (CCTV cameras), producing bulks of data, and of faster computational resources, neural networks and deep learning has gained more interest in the past few years. Deep learning is based on computational models built on neural nets as building blocks. These deep architectures have been used to aim for better accuracy for complex computer vision tasks.

Based on the installation of monocular cameras inside of warehouses, the movement of pedestrians and forklift trucks may be observed by an operator or a computer vision algorithm.

A robust computer vision-based tracking framework has to track pedestrians while taking on various poses such as standing, sitting, walking, facing backwards, possibly lifting goods of various sizes and shapes – and all those poses may be partially occluded by background clutter.

More specifically, human detection in the context of forklift applications needs to focus on improving work safety levels in warehouses [41]. Refer to Table 1.1.

Table 1.1: Key surveillance tasks in warehouses

Key tasks	Impact	Extent
Human detection	Warehouse safety	high
Goods volume	Process efficiency	high
Heat maps measure	Warehouse layout decisions/productivity	medium
Visual goods tracking	Complaint handling/productivity	high
Object recognition	Safety and efficiency	high
Quality inspection/pallets	Process improvement/efficiency	medium
Truck turnover time	Productivity	medium
Queue management	Inventory handling/productivity	low

The table lists key warehouse tasks. The third column depicts the extent of the impact, visual surveillance should have on the listed key warehouse tasks. This study will show how computer vision may contribute to those desired levels of impact by implementing detection and tracking efficiently and robustly, for process improvement in various warehouse domains.

Typically, forklift trucks operate in a warehouse in constrained environments and their motion may include frequent accelerations, decelerations, reversing and turns in the vicinity of other trucks or pedestrians [42]. Ensuring a safe work environment inside a warehouse is also very important to enhance work efficiency and to promote a productive work environment for logistics handling. Warehouses and manufacturing plants can be cluttered and busy, with trucks and pedestrians working in close proximity. Racking infrastructure can create blind spots as shown in

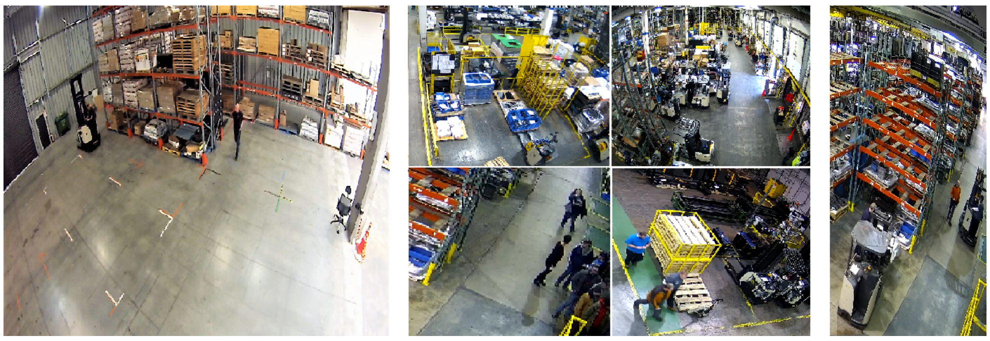


Figure 1.1: Scenarios. *Left*. Possible blind spot. *Middle*. Four challenging warehouse scenes. *Right*. Pedestrian and forklift in a warehouse

Fig. 1.1 left. Moreover, as per *Work Safe New Zealand*, warehouse related injuries by accidents define 8 percent of all the workplace injuries.

In computer vision, the goal is in general to successfully shift from model-driven paradigms to data-based approaches like deep learning to improve recognition accuracy and computational efficiency. Needless to say, that requirements for the deployment of robust tracking modules for forklift vehicles is challenging. But taking an initiative in the exploration of multiple object tracking problem in warehouses and experimenting with various state of art frameworks will open new avenues of research into industrial process improvement. Moreover, such steps will help to obtain an insight into the potentials of computer vision algorithms and deep learning models in warehouses.

Traditionally there are many methods which are normally used to increase work safety at warehouses. One of these is the use of domed mirrors, commonly known as convex mirrors, to allow the drivers and pedestrians to look around corners. Traffic lighting and gating systems are also used to eliminate the risk of forklift trucks and pedestrians being alarmingly close to each other. Systems like vehicle reversing beeps, or rules, requiring a driver to sound their horn when entering or exiting a shed, are also typical.

Robust multiple target frameworks can be utilised in a variety of ways in various domains of warehouses. Tracking multiple forklift trucks, and hence the associated goods, can aid in product complaint handling applications and in recording of any deviations. The count of the number of objects, either forklifts or persons, passing within the camera's field of view, will benefit the processes involving inventory counts. It will also help in identifying abandoned or misplaced objects. The detection of the various categories of forklift trucks (reach trucks, stock pickers and so forth) could contribute in optimising resource usage inside warehouses.

1.2 Challenges

Tracking the objects of interest over time is a challenging computer vision problem. It involves two essential phases: Localising the targets in a video frame and associating each target to a unique identity over time. Comparatively, *single object tracking* (SOT) involves fewer tasks than *multiple object tracking* (MOT). MOT includes the creation of new track identities, an optimum association measure for track assignment, matching of detection hypotheses to existing tracks based on cost affinity , deletion of a lost track when an object leaves the camera field of view, the output tracking quality and computational efficiency, just to name a few challenges.

Refer to Fig. 1.1, *middle*, for challenging warehouse scenes. Unlike SOT , the number of objects might vary over subsequent frames in MOT.

A typical warehouse scene

A typical warehouse scene is comprised of blocks of goods stacked in storage racks and shelves forming *aisles*; concrete floor stripped into zones using safety tapes of different colour codes distinguishes work zones. Components are forklift traffic lanes, pedestrian crossings, equipment placement or guard rails for safety and the *warehouse furniture* including pallets , racks, instruction signs (directing traffic and hazard warnings), with various types of *obstacles* like other forklifts trucks, pedestrians or parked trucks.

Typically, any tracking system combines three main components: target visual representation, its dynamic modelling and a search mechanism employed to look for the matching candidate detection hypothesis. The search mechanism is derived from a statistical similarity measure. Examples are *normalised cross-correlation* (NCC) for affinity computation based on raw pixel templates, or the Bhattacharyya distance measure for colour histogram models.

The visual representation of a target to be tracked is encoded in an appearance model (i.e. categories of unique features). Target appearance modelling is a vital step as the target may move in various challenging scenarios including due to clutter, full or partial occlusions, illumination changes or shape deformations. The respective motion models encode object dynamics to predict the target location in subsequent frames. Either linear motion models (i.e. constant velocity or constant acceleration assumption) or non-linear motion models are employed. Nonlinear motion models are used to represent the complexity of real-world motion. Composite model-based tracking methods have also been employed which aim at striking a balance between motion and appearance modelling schemes.

Motivated by the desire to track targets in warehouse scenes, it is essential to use a feature encoding scheme that is suited to match complex industrial environments. According to our[1] analysis, Table 1.2 lists vital MOT challenges encountered in warehouses, together with possible solutions.

[1]The use of "our" or "we" throughout this thesis is purposeful. It is used to involve the reader with the thesis as recommended in [72].

Table 1.2: MOT challenges in warehouses.

Challenges	Possible solution
Clutter	Using target representation which is robust to identify truck or a person from the background clutter.
Illumination	Using descriptors that are less sensitive to scene illumination changes.
Re-entering objects	Using a sub-module that can handle the truck or person re-identification problem and is able to preserve their identity, after re-entering the scene.
Occlusion	Using an approach robust to occlusions and able to separate the occluder and the actual target.
Similar looking	Sophisticated dynamic object appearance model, might use multiple cues like appearance, shape, colour or texture to handle ; the trucks or pedestrians that appears alike.
Scale	Adapting the corresponding truck or person representation dynamically to make it robust to scale changes.

1.3 Structure of Thesis

Refer to Fig. 1.2 for a chapter-wise outline of this thesis and how components are connected.

Chapter 2 introduces the basic mathematical notations and concepts that are used throughout the thesis. This chapter is brief and should be seen as a rough overview of the basic concepts.

Chapter 3 describes the implementation of a model-based *Gaussian mixture model* (GMM) approach which outputs pixel-wise moving segmented foreground and back-

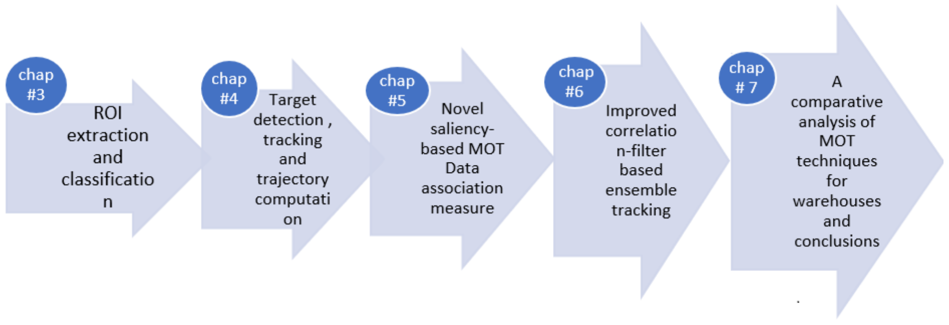

chap #3
ROI extraction and classificatio n
chap #4
Target detection , tracking and trajectory computati on
chap #5
Novel saliency-based MOT Data association measure
chap #6
Improved correlatio n-filter based ensemble tracking
chap # 7
A comparative analysis of MOT techniques for warehouses and conclusions

Figure 1.2: Thesis structure

ground regions for moving targets in warehouses. Target classification is implemented using the Inception v3 deep learning framework and onward tracking is based on a linear Kalman filter. Parts of this chapter have been published in [3]. This work was a major part of my research for 2017.

Chapter 4 outlines the details of novel warehouse forklift and pedestrian detector training and implementation details using novel warehouse acquired data. For training a deep model (i.e. *Faster-RCNN*), a pre-trained Alex-Net model is used [74]. It is a 25 layered architecture pre-trained on $1,000$ image categories. We re-trained the last three layers of the model with acquired forklift truck recorded data. A Kalman filter is used to compute the resultant trajectory of targets. Parts of this chapter have been published [1] where I was the main contributor to the paper.

A novel data association measure for MOT is explained in Chapter 5. For training *Yolo-v3*, CNN weights that are pre-trained on the ImageNet dataset are used, i.e. weights from the *DarkNet53 model* [112]. To handle track identity switches, an *appearance saliency map guided data association measure* is exploited to verify the track identity. A saliency distribution dissimilarity measure between a detected ROI and predicted candidate track locations is described by the Bhattacharyya coefficient in this work. This work has been published in [2].

Chapter 6 outlines the novel saliency-enhanced correlation filter-based visual tracker for tracking single targets. This paper has been published in [4].

Chapter 7 outlines the detailed quantitative and qualitative analysis on both groups of tracking methods, applied for warehouse image data (i.e model-based versus deep learning-based). The results are published in the *International Journal of Fuzzy Logic* [5], where I also was the leading author.

Chapter 8 finally concludes. It also lists a few recommendations for future work. Due to confidentiality limitations, tracking video output for multiple targets in warehouse is not publically available.

Chapter 2

Basic Concepts and General Review

This chapter reports about the basic concepts used for the reported research. It provides a condensed overview about the main concepts employed in this work.

2.1 Computer Vision in Warehouses

A typical warehouse is comprised around five main modules of operation. Right from the start, there is order receiving, storing of the received goods, picking them for a specific order, packing or loading for the dispatch, and the final shipment. There are many sub-processes involved. These operations need to be monitored and well taken care of for handling logistics efficiently.

Multiple types of sensors, like laser-based range sensors, radio frequency-based sensors or cameras, have been used to optimise and improve these tasks in various ways. Recently it is been seen that video-based analytics have been employed by the warehousing industry for improved productivity and safety at premises. Making use of recorded video data from *closed-circuit television* (CCTV) cameras, either offline or in real-time, helps to measure and improve performance metrics of warehouses and record unusual events for surveillance applications. Refer to Fig. 2.1. The figure shows generalised key warehouse tasks.

According to our study, Fig. 2.2 sketches the three main areas that divide the use of computer vision techniques in the warehousing field. They include, but are not limited to: optimising and simplifying key warehousing processes, perceiving the warehouse environment, the self localisation of forklift vehicles, and improving the situational awareness for the workers (enhancing work-related safety).

Laser, LiDARs (i.e., *light detection and ranging*) and cameras have been used to understand the environment in warehouses and ensure safety [42, 41, 52, 56, 73, 99]. On-board camera setup was employed for *automated guided forklift vehicles* (AGVs) for various tasks, for example, for improved pallet localisation and recognition as

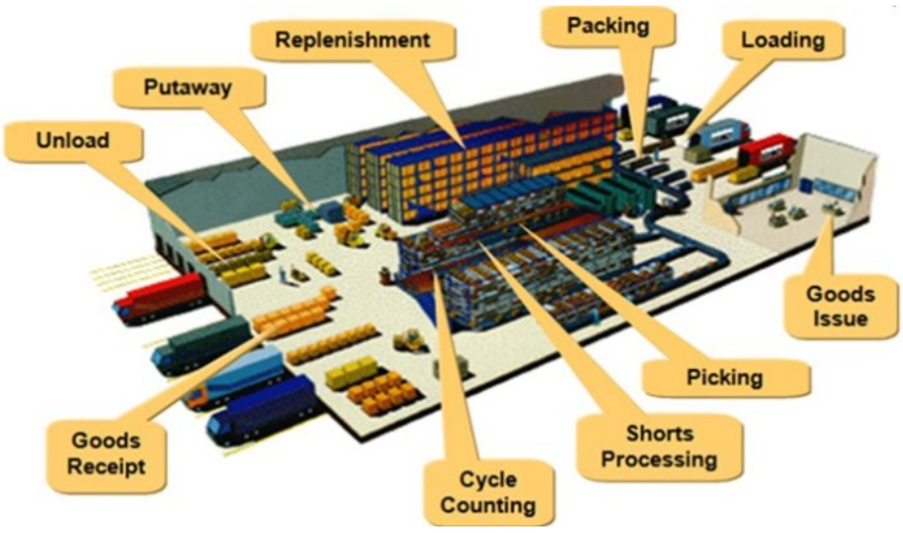


Figure 2.1: Key warehouse tasks [140].

in [70, 34]. Here, a vision system, designed for detecting the 3D position of pallets for autonomous forklift vehicles, employed image segmentation methods based on colour and geometric characteristics of pallets. It also generated a vehicle trajectory to track the 3D oriented pallets [107]. For 3D guided robotic pelleting, inven-

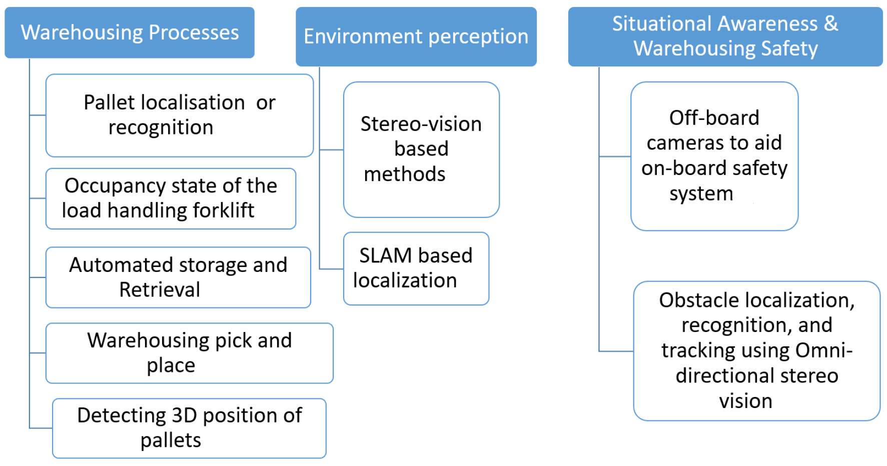

Warehousing Processes
Pallet localisation or recognition
Occupancy state of the load handling forklift
Automated storage and Retrieval
Warehousing pick and place
Detecting 3D position of pallets
Environment perception
Stereo-vision based methods
SLAM based localization
Situational Awareness & Warehousing Safety
Off-board cameras to aid on-board safety system
Obstacle localization, recognition, and tracking using Omni-directional stereo vision

Figure 2.2: Computer vision applied in warehouse areas.

tory package inspection, package volume scanning and smart conveyance, on-board camera sensors have been used in conjunction with other technologies [123].

A computer vision-based line detection and tracking algorithm was implemented in an automated guided vehicle in [7]. A robust system, based on the detection of *salient regions*, was able to perform tasks like navigation and grasping, was proposed in [110]. An on-board camera was used for warehousing pick-and-place operations in [56]. For warehouse pick-and-place in shelves, the use of additional cues is under consideration. Depth sensors, such as RGB colour combined with a depth channel (RGB-D), were tested for detecting the pose of a known object. Shelves having issues like low illumination inside shelves, being cluttered, or having texture-less and reflective objects are common in this domain [109].

An on-board vision module may be combined with *radio frequency identification* (RFID) for automated storage and retrieval in warehousing, for efficient storage handling and for preventing inventory discrepancies [144]. Camera-based approaches were fused with other sensor-based methods to find the occupancy state of the load handling device of a forklift truck [12].

Improved autonomous load handling was done by using stereo cameras in warehouses [135]. There are many solutions for stereo vision-based approaches [76]. For example, in [51] the authors proposed a 6D vision approach based on tracking of individual 3D points using an optical-flow solution. Later stereo data such as *digital elevation maps* (DEMs) or occupancy grids were introduced for better perception accuracy. A DEM provides the height information in addition to occupancy values, which is suitable for perception in crowded environments. On-board stereo-vision based methods are also used to improve perception accuracy and trajectory path planning.

Obstacle localisation, recognition, and tracking was done using omni-directional stereo vision in [42]. Fish eye cameras were used for up to $360°$ perception, with a possible application for stereo vision. An obstacle hypothesis was generated using a classified DEM-based representation. To ensure pedestrian safety, both monocular and stereo vision were employed in this case.

Sensor fusion is being used to perceive the surrounding environment of automated guided vehicles inside a warehouse. Common sensors for AGVs are laser scanners (LiDAR) that provide 2.5D perception (i.e. the visible 3D geometry) of the environment.

Systems with many non-overlapping cameras are used for tracking vehicles and pedestrians over larger surveillance areas. Multiple cameras form a wider baseline stereo that covers the scene from several directions. This helps in scenes with occlusions, particularly in crowded scenes and determining the accurate location of people. Disparity discontinuities were used to aid in segmentation, as in [131], to divide the image of the scene into multiple layers. In [42], stereo guided an active contour model for pedestrians.

Several video-based analytics solutions are also being offered recently to improve logistics handling in warehousing. To optimise the standard traffic flow inside the warehouses, *heat map* analysis is very useful. Heat maps enable to see heavy traffic areas in the warehouse and to decide what could be the reason for high density in such areas. Through using these maps, decisions about shifting of certain products to other areas that experience lower traffic could be made. It will also limit customer wait time and even shipping or loading time.

In summary, the diversity of research demonstrates that the analysis of recorded video data, for any specific or abnormal event monitoring, and generating useful condensed information is useful for warehouse surveillance applications. Restricted zone monitoring, detection of entry or exit of people or vehicles from restricted zones, parking at loading docks, are also a few tasks that are optimised using video-based analytic solutions.

In this study, multiple monocular off-board cameras are fixed at key warehouse locations to acquire video data featuring vital warehousing events. A novel (extensive) recorded video data set is applied in our design and analysis, for targets of interest and particular tracking events, under various challenging scenarios. We carry out a review, implementation and comparison of traditional computer vision techniques versus deep learning frameworks, for achieving improved localisation accuracy of the targets of interest and robust tracking results.

2.2 Multiple Object Tracking Problem Formulation

A monocular camera C_1 is fixed at a main location in a warehouse capturing images (i.e. video *frames*) denoted as $I_1, I_2, I_3, \ldots, I_N$. Every frame contains a varying number of m objects, quantified by

$$A_1, A_2, A_3, \ldots, A_N \tag{2.1}$$

An object in a frame can either be a person or a truck; hence we have a two classes. As the objects move in the scene, the number m will vary over time.

We use a state vector S_i to represent locations in pixel co-ordinates for the $m = A_i$ objects present in frame i, i.e. $S_i = (l_i^1, l_i^2, l_i^3, ..., l_i^m)$. We denote the subsequent locations of those objects in the range of $i = 1$ to $i = t \leq N$ as

$$S_{1:t} = \{S_1, S_2, S_3, \ldots, S_t\} \tag{2.2}$$

A visible object j has locations

$$l_{s:f}^j = (l_s^j, l_{s+1}^j, \ldots, l_f^j) \tag{2.3}$$

in Frame s to Frame f, for $1 \leq s \leq f \leq N$. Also, $l_{s:f}^j = (l_s^j, \ldots, l_f^j)$ starts and ends with the initial and final subsequent location, respectively, of the j-th object in the range s to f (where it may not be a single frame range and might involve a union of ranges, as the object may leave the scene and reappear at a later stage).

Moreover, using a tracking-by-detection approach, we measure the states of these objects at every frame using some detection algorithm. We denote the collected observations from objects for Frame i as

$$O_i = \left(o_i^1, o_i^2, o_i^3, \ldots, o_i^m\right) \tag{2.4}$$

for $m = A_i$. Subsequent observations of those multiple objects, between Frame 1 to Frame t are combined in

$$O_{s:f} = \{O_s, O_{s+1}, \ldots, O_f\} \tag{2.5}$$

for $1 \leq s \leq f \leq N$.

The objective of multiple object tracking is to estimate, for every corresponding Object j belonging to Frame i, i.e. o_i^j, a location l_i^j for all sequential frames in which it appears. This location should be close to the *ground truth location* $[l_i^j]^{gt}$, assumed to be measured in some accurate way (e.g. for test data by manual labelling).

Depending on the approach, this location estimate can be a *2-dimensional* (2D) centroid location (x, y) of the target in pixel coordinates, or it can be a *3-dimensional*

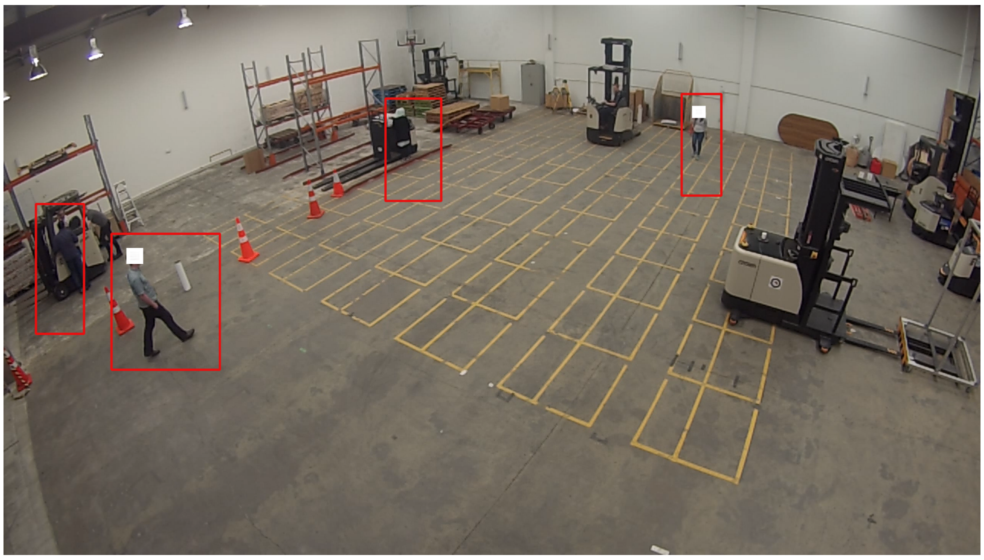


Figure 2.3: A frame of a warehouse recording depicting multiple objects belonging to the class `pedestrians`.

(3D) world coordinate location estimate (X, Y, Z) if a target is mapped from 2D images into the 3D world.

When seeking a solution to the multiple object tracking problem, some physical constraints need to be considered. Two different objects cannot occupy the same physical space in the real world. Since we deal with multiple detection hypotheses, a constraint is that in the same frame, two detection-hypotheses cannot be assigned to the same track.

Figure 2.3 shows multiple objects belonging to the class `pedestrians`. The tracker should be able to detect and track various instances of multiple classes over time.

2.3 Detection-based Tracking Approaches

Based on how objects are initialised, the MOT problem can be classified into two categories.

In *detection-based tracking*, objects are first detected and linked into trajectories to form corresponding tracks. This strategy might involve detecting objects based on training an object detector in advance [146]. Alternatively, objects are detected using motion blobs (background modelling). In a background modelling-based tracking approach, a *region of interest* (ROI) is detected by finding a background representation and then finding deviations from it in the following frames.

In *detection-free tracking*, targets of interest are initialised in the first frame and localised in subsequent frames. Processing of the frames is done either *online* using the detection hypothesis from the current frame or *offline* which is based on the collection of hypotheses from all the frames in advance to estimate the output track [84]. The robustness of resulting tracks depends heavily on the detection quality of the employed detector.

Some image segmentation algorithms are also used to partition the image into clusters or segments. In data-association-based tracking approaches, detection responses are linked to trajectories with global optimisation (based on size or similar appearance); see [9].

Based on the used search mechanism, tracking can be categorised into either deterministic or probabilistic approaches. Probabilistic inference is based on a representation of object states as a distribution with some degree of uncertainty. Based on current observations, tracking algorithms are used to estimate the probabilistic distribution of target states. These approaches are based on past and current observations and are most suitable for online tracking. A linear Kalman filter, an extended Kalman filter, or particle filter frameworks come under this category [64].

Deterministic optimisation approaches aim to estimate the *maximum-a-posterior* (MAP) solution for the tracking problem. Observations from a specific time window for all the frames are needed in advance to estimate the trajectory of the target by globally associating them. These methods are more suitable for offline tracking applications. Bipartite graph matching, dynamic programming, min-cost max-flow network flow, conditional random field and finding the *maximum-weight independent set* (MWIS) of the graph are examples [71, 87].

To deal with occluded and challenging tracking environments, further detection approaches have been explored. They rely on a correct detection output from the object detector. In some cases, 3D information or trained detectors were used for individual object parts or application-specific motion models [84]. Learning object view angles from a set of training examples employing supervised learning is also

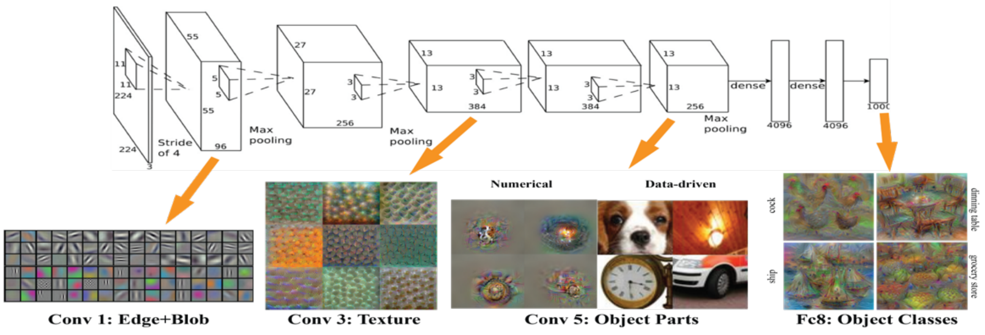


Figure 2.4: Features are learnt in increasing abstraction [121].

used to detect and classify ROIs [106].

Representation modelling of target appearance using *deep features* is used as an effective cue in many recent tracking frameworks. Many such methods proved to be robust in tracking under occlusions and varying scales of objects. These features are extracted from a deep neural network (trained for image recognition or classification task) and used in places of conventional hand-crafted features in existing MOT paradigms. Tracking methods employing deep learning frameworks are also used to model the target of interest appearance using networks pre-trained for other tasks (transfer learning) and to associate these target objects over multiple frames.

2.4 Model-based ROI Extraction Methods

Tracking is often preceded by interest object localisation in single or multiple frames. The representation of a ROI is based on object shape or an appearance model and which unique features are suitable to encode object representation. MOT algorithms might also use multiple cues to represent application-specific objects of interest.

Hand-crafted features have been used to encode the appearance of targets of interest. Feature descriptor schemes might include optic flow, the *scale-invariant feature transform* (SIFT) [85] or gradient based-features, *histogram of oriented gradients* (HOG) [38], colour, bag-of-features [28], or deformable part-based models [47], classic classifiers such as *support vector machines* (SVMs) [32] and random forests [18]. The *probabilistic occupancy map* (POM) and depth features have also been studied.

2.4.1 Gaussian Mixture Model

The values of particular pixels are modelled as a *mixture of adaptive Gaussian distributions* (MOG). As a simple example, pixel value variations of a pixel (x, y, u_i) in its history of its previous t values, say from Frame 1 to Frame t, can be represented by the set $\{u_1, ..., u_i, ..., u_t\}$, where $1 \leq i \leq t$. In this example, the pixel location (x, y) would remain constant, and values u_i are assumed to be scalars (as in a grey-level image). As a more general example, distributions of values u are considered for the whole frame, assumed to be a combination of Gaussian distributions, and these combinations are studied over time, say from Frame 1 to Frame t.

The probability density function of the univariate *Gaussian* or *normal distribution* is given by

$$G(X, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2\right\} \tag{2.6}$$

for $-\infty < X < \infty$, where μ is the mean and $\sigma^2 > 0$ is the variance of the random variable X (here our image values u).

The probability of observing a specific mixture component at Frame t is given by products of *probability density functions* with their weights. For more than one density function, we have a multivariate case, such as

$$P(X_t) = \sum_{i=1}^{k} w_{i,t} \cdot G(X_t, \mu_{i,t}, \sigma_{i,t}) \tag{2.7}$$

Here, $w_{i,t}$ is the weight of the i_{th} Gaussian distribution at Frame t.

Aiming at a probabilistic model for separating the *background pixels* from the *foreground* by looking at the distributions, [115] proposed an update of the background model as follows:

1. *Constructing an adaptive mixture of multi-modal Gaussians per pixel*. The number k of Gaussian components depends on the environmental complexity one wants to model. In a typical warehouse indoor environment, we observed relatively minor contrast in colours and brightness. Outdoor scenes have different conditions. Following [115], we keep $k = 3$. Targeting red-green-blue (RGB) colour images, we also assume that all three RGB channels have the same σ^2, thus defining a 3×3 covariance matrix $\Sigma_{i,t}$ being the product of a variance with the unit matrix.

2. *Method for updating the Gaussian parameters*. For every new pixel state for the next frame, we check whether it lies $X_t \leq 2.5$ standard deviations from the

mean; we label it *matched* in this case. We update weight, mean, and variance as per the following update equations:

$$w_{i,t} = (1-\alpha)\cdot w_{i,t-1} + \alpha \cdot M_{i,t} \tag{2.8}$$

$$\mu_{i,t} = (1-\rho)\cdot \mu_{i,t-1} + \rho \cdot X_t \tag{2.9}$$

$$\sigma^2_{i,t} = (1-\rho)\cdot \sigma^2_{i,t-1} + \ \rho(X_t - \mu_{i,t})^\top (X_t - \mu_{i,t}) \tag{2.10}$$

where $\rho = \alpha \cdot P(X_t|\mu_{i,t-1}, \Sigma_{i,t-1})$, $0 < \alpha < 1$ is a selected learning rate, and $M_{i,t}$ equals 1 for a model which is matched, and equals 0 for other models.

If the i^{th} Gaussian is marked as *unmatched*, we decrease its initial weight as per below equation:

$$w_{i,t} = (1-\alpha)\cdot w_{i,t-1} \tag{2.11}$$

If all the k Gaussians in the mixture model, for pixel value X_t, are not matched to the pixel, we mark that specific pixel as a *foreground pixel*. If this is the case, then we find the Gaussian distribution with the lowest weight in the mixture and set the mean equal to X_t. We also adjust the corresponding variance to a higher value and lower the weight of this distribution.

3. *Heuristics for determining the background.* For finding the background distributions, we rearrange the distributions in descending order by w/σ. We add up the corresponding weights of the Gaussians in this order, till the final sum is greater than a pre-set threshold T. We set $T = 0.9$ in our case. We observe that there are fewer salient or moving objects and more background portions in the frames. Refer to Fig. 2.5 for the concept.

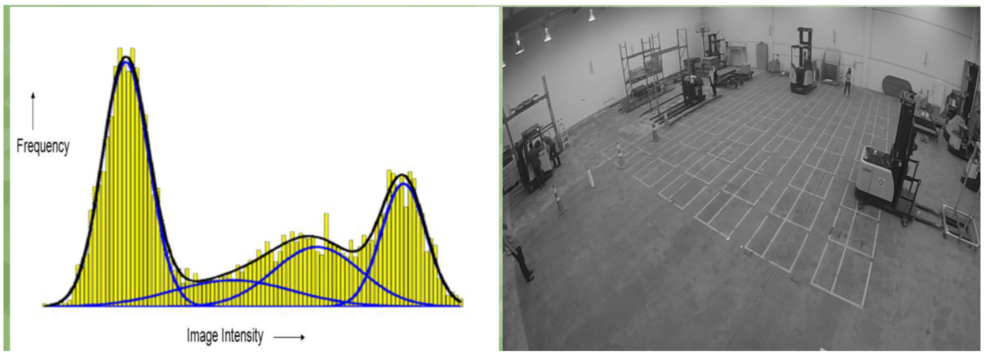


Figure 2.5: Mixture of three Gaussian distributions is trained to changing pixel values in every subsequent frame, for modelling the background.

Recent Variants for Addressing Challenges

Gaussian mixture models have been an active field of research for two decades. Many variants have been introduced for dealing with various challenges when dealing with real time moving target detection.

Shadow elimination has been a major subject. Foregrounds obtained by a MOG technique have shadow pixels as part of the foreground. Much work has been targeted towards shadow elimination [50, 130, 143, 152]. Shadow detection in colour space is considered in [50, 75].

For the detection of *slowly moving objects*, see [50]. Challenges arise when these objects are incorporated into the background due to less variance.

Adaptation of algorithms to scene changes. This is very important and controlled to some extent by learning rate and parameter selection, see [55, 75, 156]. *Background recovery rate improvement* was also studied in [143] for solving real-time surveillance issues. To incorporate abandoned objects for surveillance applications, *abandoned/removed object detection* is also thoroughly researched in [130].

Update to learning equations. This involves controlling the scene changes and slowly or fast-moving objects, see [55, 103, 156]. *Learning rates* and their significance for incorporating scene changes is studied in [142]. The *number k of Gaussian components or modes* depends on scene complexity and pixels modes [156]. For run-time improvements, to adapt MOG to real-time, see [143, 156].

Initialisation of parameters is very important for initialisation of a MOG model [143]. Parameter analysis and setting as per scenario is dealt with in [152].

Dirichlet-Gaussian distribution. [58] used a Dirichlet process and a Gaussian mixture model to estimate a per-pixel background distribution, which is followed by probabilistic regularisation. This work was able to accurately model dynamic backgrounds.

Neighbourhood correlation, to update the parameters of MOG [103], is also found effective. Importance of spatial information other than temporal one, for detecting accurate foregrounds [142, 143], was able to improve the foreground quality. For using other cues such as intensity and texture, for better foregrounds, see [130]. This approach was not able to deal with resultant holes in foreground masks.

2.4.2 Pixel Saliency-map-based Detection

Saliency refers to the visual contrast of the object compared to its background region. Such a contrast might be based, for example, on the difference between colour, texture, or shape. Thus, it is based on measuring human attention that any salient object attracts in an input image. Saliency detection has received considerable progress in the field of computer vision.

Saliency map computation is used in many computer vision areas like object recognition, image segmentation, video or image data compression, visual tracking and robotics. Moreover, various methods have been proposed for saliency detection, based on some priors. Such priors might include uniqueness, background contrast, compactness, random walks, deep learning and others [31].

Traditional saliency detection models are classified into two main domains. These are named as *bottom-up* and *top-down approaches*.

The bottom-up model is based on low-level visual features like compactness or uniqueness. Uniqueness-based methods are further split into *local* and *global* contrast methods. Most uniqueness-based methods use low-level features such as colour, direction, or intensity to determine the contrast between image regions and their surrounding pixels.

Compactness-based methods use the variance of spatial features. Salient pixels tend to have a small spatial variance in the image space. The background is distributed on the whole image space and tends to have high spatial variance. Since single visual cue-based salient region detection methods have few limitations in detecting accurate salient pixels, different cues could be combined to make a composite framework [105]. Some methods are based on this approach, but the selection of visual cues depends on the context. Compared with the global contrast method, the local contrast is a relatively better cue, to be combined with the compactness cue.

Local contrast methods can identify the foreground region [60], but they have a limitation that they identify visible object boundaries rather than all the area. This effect could be minimised by the propagation of saliency information based on diffusion [60].

The top-down model is task-specific and has shown high performance based on supervised learning with labels. A detailed study on the significance of using saliency-based ROI extraction and onward tracking, depicts the usefulness for using saliency feature advantages over various other visual cues such as motion, texture, gradient or colour. An object is represented and tracked based on a human attentive mechanism using visual saliency maps. Such methods have an advantage to work better for occluded and cluttered scenes.

A pixel saliency map was also being combined with other features types, like colour-texture, colour-saliency, or colour-orientation [6]. Tracking the targets of interest employed a simple saliency visual descriptor which counts the number of similar pixels lying in the local neighbourhood named *local similarity number* (LSN). It was used to model the amount of saliency in corresponding target patches. It was being used in a mean-shift tracking framework with a saliency-colour histogram model [126]. Spatio-temporal discriminative saliency maps were used to track non-rigid objects; this outputs accurate regions occupied by the targets as tracking results.

Saliency detection using deep learning techniques have displayed very promising results, since the recent progress in deep learning models. A *tailored fully convolutional neural network* (TFCN) was developed to model the local saliency of regions of interest. Latter local saliency maps were generated with the help of a multi-scale multi-region mechanism that takes into account the visual perceptions with varying spatial layouts and scales. Finally, these local saliency maps are fused with a weighted entropy method, resulting in a final discriminative saliency map [157].

Improvement in discrete correlation filters-based trackers with saliency-based filter responses helped to handle tracking in challenging scenes. Filter weights were selected adaptively based on temporal consistency of visual saliency maps [13]. Particle filters suffer from tracking artefacts in occlusions, clutter and illumination variations. Saliency information incorporated in the framework aided to improve tracking results in complex scenes. A bottom-up saliency-based tracker that tracked any salient target in the scene used colour features and sparse optical flow [118].

Static and motion features based on saliency are first extracted from the frames of videos locally, regionally or globally. Then they are combined in a conditional random-field fashion. The salient region in the frame was tracked using a particle filter. Tracking was robust w.r.t. changes of illumination and shape and can track any object category as long as it is salient in the scene [159].

A smooth pursuit tracking algorithm is proposed that uses three kinds of saliency maps. These were appearance, location and motion saliency maps. Appearance saliency maps use deep CNN-based features along with gnostic fields. A location map predicts where the object location will be in the next frame and motion saliency maps show which objects are moving in the scene. All three maps were fused together in a smooth pursuit fashion. The resulting map was used to generate bounding boxes for the tracked targets [149].

A deep contrast net for saliency detection has also be employed. A multiple-scale fully convolutional neural network detects visual contrast saliency, and the spatial pooling stream simulates saliency discontinuities for every segment along with object respective boundaries in an end-to-end fashion [86]. Through a super pixel-wise convolutional neural network, hierarchical contrast features for saliency were detected. Also, colour uniqueness and color distribution were embedded into the CNN at various scales. [66].

Zhang et al. [153] utilised a *fully convolutional neural network* (FCN) (encoder) and

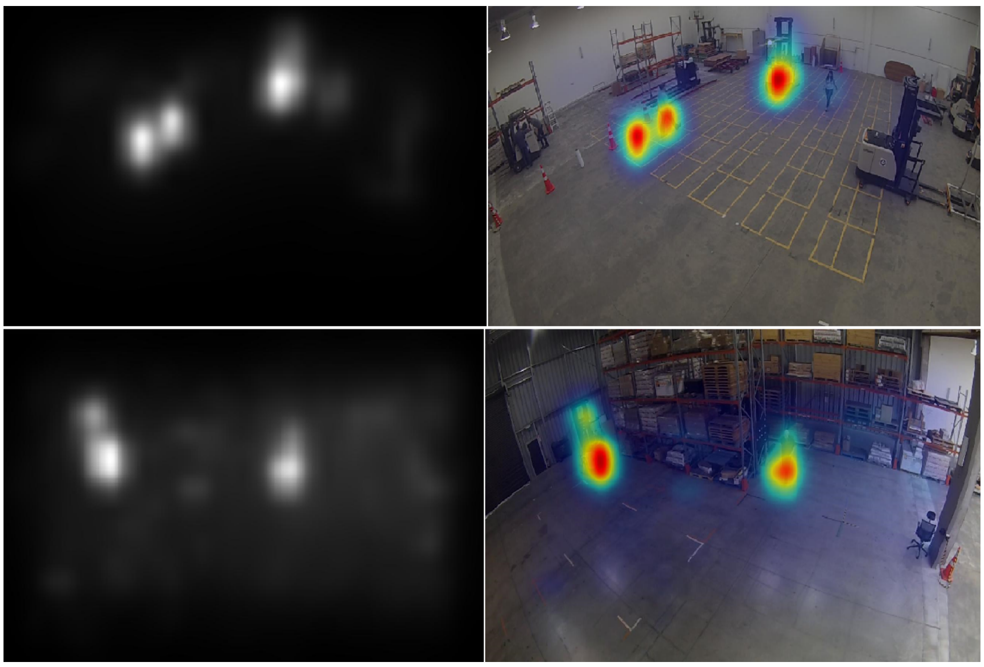


Figure 2.6: An example of saliency detection for two objects (forklift and person) with a corresponding heat map overlayed on the right.

a corresponding decoder to detect the salient objects in the scene. A *reformulated dropout* (R-dropout) was introduced to construct an uncertain ensemble of internal feature units. To reduce the checkerboard artefacts of deconvolution operators, a hybrid up-sampling method was designed in the network.

A deep hierarchical saliency network was formed by the integration of a CNN with a *global-view CNN* (GV-CNN) and a *hierarchical recurrent CNN* (HRCNN), where the GV-CNN computes a coarse saliency map, the HRCNN recovers the image details from local context information [96].

For this work, pixel saliency map computation is tested for extracting the ROIs; it has been reported in [3]. We experimented with the computation of static and motion saliency maps, as previously suggested in [65, 154]. *Deep hierarchical saliency network prediction* (DHSnet) is also employed to output the saliency map of an image using a deep CNN in [4], for better efficiency. The whole image (i.e. frame) is used as computational unit and feed-forwarded for testing without any post-processing; see [96] for the original proposal.

Refer to Fig. 2.6 for a saliency detection example for an image acquired in a test warehouse environment. For the *heat map*, the computed saliency for a ROI is superimposed (using a colour key) on the respective image.

2.5 Deep Learning Models

In addition to handcrafted feature extraction, automatic feature learning by neural networks, especially deep convolution architectures, is state of the art.

Deep learning is a form of representation learning. A computer is fed with large amounts of raw data and it finds out the features needed for detection, based on learning. Deep convolutional neural nets, proposed by Krizhevsky et al. [74], have achieved tremendous success on bigger benchmark datasets, such as ImageNet. ImageNet is a dataset of over 15 million labelled high-resolution images belonging to roughly $22,000$ categories. It took between five and six days on two GTX 580 3GB GPUs to train a network with the ImageNet dataset. Some modern object recognition models have millions of parameters and may take some weeks to be fully trained. Hence, traditional deep learning models need a huge amount of training data for training and resources, such as multiple GPUs. They have successfully been used to categorise images and activities or tasks once trained with an excess of data samples [43].

Googles Inception [122] is one of the CNN architecture that provides a good-performance network with relatively low computational costs. It has 9 inception modules. Each colour in Fig. 2.8 depicts different operations: Blue colour for convolution operations, red for pooling, yellow for softmax, and green for others.

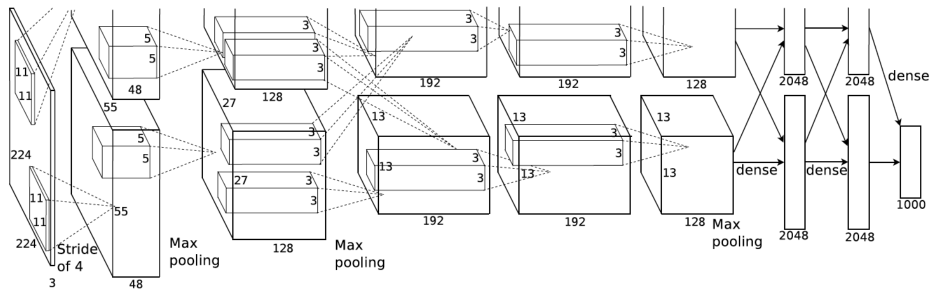


Figure 2.7: AlexNet architecture [74]

Another famous architecture model is the AlexNet [74]; it achieved a top-5 error rate of 15.3. See Fig. 2.7. Inception (GoogLeNet) [121] achieved 6.67 in the same category. See Fig. 2.8.

Deep learning models have been employed to solve MOT problems in a variety of ways. We divide the way deep models have enhanced the solution of MOT problems into three streams.

Using deep network features

Deep features, being more discriminative than hand-crafted conventional features, are used to replace them inside the same MOT framework to enhance the tracking performance. Training a *convolution neural net* (CNN) as the used model can be done by using extensive classification datasets already available, or by using tracking data, or pre-training and fine-tuning the model. Fully trained CNNs are employed for region proposal generation for object detection [111]. Siamese CNN architecture was employed to learn the matching features for multiple object tracking problems.

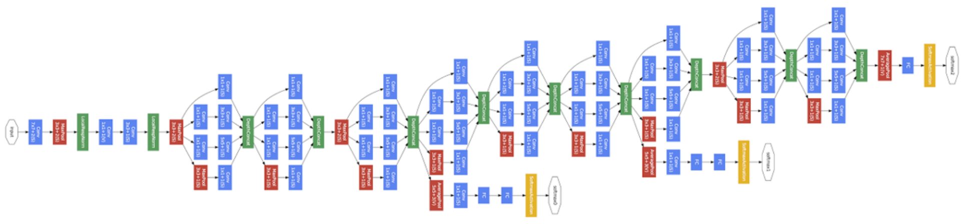


Figure 2.8: GoogLeNet (Inception v1)

Leal-Taixe et al. formulated a deep architecture by fusing deep features and a motion prediction algorithm; a linear programming approach was used to solve the tracking problem [88]. Since the optic flow is a useful feature to learn track association, learning deep flow through deep architecture made it more efficient and accurate for MOT. Similarly, pair-wise images were fed to a Siamese network to compute cost affinities for tracks. Xiang et al. proposed to learn a triplet-loss based CNN to find the distance between trackers and detections. In the tracking process, this distance between trackers and detections constructs the cost affinity for bipartite graphs, which is solved by the Hungarian algorithm [145].

Deep learning for action classification in videos

Recognizing actions, events and activities in video data is being explored using deep learning, but the state of the art is still far away from human recognition abilities. Using CNN for feature extraction (both spatial and temporal) and employing it for action classification is done already, but it still did not achieved the requisite level of accuracy. Although scene labelling domain is still a success in this regard.

Compared to object classification, action classification in videos suffers from challenges like variations in object motion and respective viewpoints. Thus, it needs a lot of training examples per action class. It also needs better CNN frameworks that can extract not only the spatial (appearance) features but also temporal information precisely, to be able to classify actions and events in video streams.

Instead of using two dimensional (2D) convolution that aimed at extracting only spatial information, three dimensional (3D) convolution were being used for spatio-temporal feature learning [128]. 3D CNNs were being trained on large supervised training video datas that modelled the object appearance and motion simultaneously. Thus employing these 3D kernel operations for all layers and using a simple linear classifier output promising results for varying tasks and popular benchmarks.

The task of video recognition, captioning the image and video description was also being done on architecture based on connecting a CNN with a deep *long-short term memory* (LSTM) network. These composite visual models were able to capture challenging state dependencies for accurate recognition of videos [39] and were end-to-end trainable.

One of the approach was an improvement on two-stream architecture proposed by Simonyan in [114]. To incorporate temporal motion information, a separate CNN was trained based on optical flow feature. For video action recognition, two-stream fusion was proposed [49]. Spatial and temporal cues were fused at several levels (spatial as well as temporal integration). The architecture was revised with fusion operation occurring at convolution layers rather than at the softmax layer done earlier.

RNNs are known to be deep temporally. Precise mapping from pixels in images to sentence-level description, describing the event occurring in the scene is achieved. These models have been used to generate scene captions from various intermediate visual features derived from conventional neural network models.

Using core modules learned by a deep neural network

If core MOT modules and processes are learned using a deep neural network, the tracking performance is improved. This study splits the domain into the following:

Discriminative network learning is known as a tracking-by-detection approach. For instance, a target-specific particle filter framework was employed. Features from Faster-RCNN VGG-16 were used, where the top and bottom layer output is used to obtain weights for the particles. A resulting track was estimated through a particle filter framework [29, 113].

Deep metric learning for MOT is learning affinity or a distance measure of detection pairs for the *tracklet* association problem (a tracklet is a fragment of a track followed by a moving object).

One of the main processes in MOT is the association of detection hypotheses over subsequent frames with a correct track. An optimum association measure was constructed by using learned appearance features from a person re-identification dataset [139, 147]. To handle track identity switches, an *appearance saliency map guided data association measure* is exploited to verify the track identity. A saliency distribution dissimilarity measure between a detected ROI and predicted candidate track locations is described by the Bhattacharyya coefficient in our work.

Extending deep models for generative learning is used to learn MOT vital parameters for data distribution and can be learned through *generative adversarial networks* (GAN). Fang et al. proposed to model object motion and appearance using a posterior probability Gaussian distribution, using an auto-linear regression method. A generative *long short-term memory* (LSTM) model was used to generate a confidence map output. A pixel-wise probability map was generated through a decoder following an LSTM layer [146].

Using an end-to-end deep network

This approach is applied to model the whole process of target tracking. Since many related sub-processes are intertwined inside the framework, the model relies on a few assumptions like Markov properties, or a fixed distribution. For the online MOT problem, a recursive Bayesian filter comprises prediction and update modules. Object states, observations, matching matrix and existing probabilities are fed as input into the *recurrent neural net* (RNN). Predicted states, updated results and existence probabilities are the output. A set of LSTMs was used to compute a matching matrix. Either one was used to find a match among one of the object states and the current observations. Multiple tracking segments (i.e. tracklets) were used to train a group of LSTMs and an RNN [48].

Deep networks are also used to learn regression models in SOT and also a few instances in MOT for regression learning. In comparisons with CNNs, RNNs are more suitable for target sequence modelling and for predicting the target's next state according to its historical information. However, to simplify learning the appearance features, some existing CNNs were used to extract deep features and fed as input for RNNs [114].

2.6 Kalman Filter

The original Kalman filter is a recursive adaptive filter that estimates the state of a linear, discrete-time dynamic system. Each updated estimate of the state is computed from the previous estimate and the new input measurements data, based on knowledge about process noise and measurement noise. Kalman filter is computationally more efficient than the methods based on computing the estimate directly from the entire past observed data.

One key reason for employing a Kalman filter in tracking is that it can be configured to deal with occlusions or predicting during missing detections. A separate Kalman filter can be configured for each detected truck, when dealing with multiple target tracking. To use the Kalman filter, the object is assumed to have a linear motion model (moving at constant velocity) which is reasonable for short time intervals.

The state vector, denoted by $x_k \in \mathbb{R}^n$, characterises the system state at the previous time k. $x_{k+1} \in \mathbb{R}^n$ is the state at the current time $k+1$.

The Kalman filter has two main steps, a prediction step and a correction step. The prediction step uses a previously estimated state x_k^p and the linear model A to predict the value of the next state as well as the state's estimation covariance:

$$x_{k+1}^p = A \cdot x_k^p + B \cdot u_k + w \tag{2.12}$$

x_{k+1}^p is the predicted state vector at instant $k+1$. A is the state transition matrix taking the state x_k from time k to time $k+1$. To account for the uncertainty resulting from the inaccuracy of the model, white noise to the model is added (i.e. w is a Gaussian distribution noise with a mean 0 and a variance Q).

The predicted state error co-variance matrix P_{k+1}^p is updated by the following equation:

$$P_{k+1}^p = A \cdot P_k^p \cdot A^T + Q \tag{2.13}$$

Also,

$$y_{k+1} = H \cdot x_{k+1} + v \tag{2.14}$$

where y_{k+1} is the measurement observable at time $k+1$ and H is the measurement matrix. The measurement noise v is assumed to be additive, white, and Gaussian, with a co-variance matrix defined by R. Moreover, the measurement noise is uncorrelated with the process noise.

For the correction stage, the Kalman filter corrects the state estimated based on the current measurement, such as object location, based on the *Kalman gain* K. It optimally updates the state-estimation vector (i.e., the prediction at instant k) and the state co-variance matrix, by an *innovation step* of the filter:

$$\begin{aligned} S &= HP^p_{k+1}H^T + R \\ K &= P^p_{k+1}H^T S^{-1} \\ x^e_{k+1} &= x^p_{k+1} + Kz_{k+1} \\ z_{k+1} &= y_{k+1} - Hx^p_{k+1} \\ P^e_{k+1} &= (I - KH)P^p_{k+1} \end{aligned} \tag{2.15}$$

where z_{k+1} is the measurement residual vector and S is the residual's variance matrix, also updated as described above.

2.7 Track Association Methods

Detection output by detector to track association is a vital challenge in designing multiple-object tracking algorithms. The robustness of such assignments is very important for seamless tracking results. Such an association is mainly classified into two broad categories. Local methods employ a pair of frames for data association; this is fast, but irrelevant factors like camera motion or pose variations are likely to cause a track miss-assignment problem [146].

Global techniques perform the same association using a batch of frames. Moreover, a few global association techniques consider this association as a constrained network flow problem. K-shortest path algorithm is used for associating the tracks in [19].

Tracking partially occluded, closely located targets suffers from shortcomings incurred by object detectors, especially when following the tracking-by-detection paradigm. Bochinski et al. [17] employed the intersection over union (IOU) measure and predefined threshold values for the association of targets in frames. Authors depicted acceptable tracking performance at 100 frames per second for tracking. Chen et al. [78] proposed a multiple hypothesis tracking method by exploiting detection correlation across video frames. This method claimed to handle objects lying very close to each other or partially occluded one. In a few other techniques, dense detections were also being used without non-maximum suppression for tracking, mainly to handle scenarios where targets are in close proximity [132].

Track association methods employing deep learning frameworks were used to model the target of interest appearance using networks pre-trained for other tasks (transfer learning) and to associate these target objects over multiple frames. A network trained for learning data association in the context of MOT was employed in [124]. A Siamese network was modified to learn deep features for multiple-object tracking with object association [20].

A multipurpose CNN for both detection and tracking under common region-based fully convolution neural network (R-FCN) was proposed [61]. A model for person tracking using head-joints was employed based on human pose estimation; it failed to deliver desired results under occlusions [69]. Associations of objects in consecutive pairs of frames was jointly modelled and associations are learnt in an end-to-end manner. Object affinities were estimated in frame-pairs using a deep affinity network [113]; it is suggested to handle occluded scenes.

Despite the rich literature regarding deep learning-based target tracking, results still suffer from the lack of robust track association strategies especially for close proximity and partially occluded objects. A novel data association criterion was proposed in this research work for robust track association, especially for partially or fully occluded targets [2].

2.8 Correlation-filter-based Visual Tracking Methods

Visual object tracking (VOT) is a challenging computer vision problem where the target of interest needs to be tracked in every frame. Challenges might include target deformation caused by frequent appearance changes, abrupt motion, background clutter, or partial or full occlusions. Two types of approaches exist to handle such problems, either *generative* [95, 93] or *discriminative* methods [21, 62].

Generative methods for visual tracking tackle the problem by searching for regions which seem to be most alike the target candidate model. The models can either be based on templates or subspaces.

The discriminative approach aims to differentiate the target from the background by considering tracking as a binary classification problem. It uses both the target object and background information to search for a decision boundary for differentiating the target object from the background region.

In *discrete correlation filter* (DCF) based tracking (see next subsection for details), an initial state of the object to be tracked is known, either by detecting the target automatically or by manually specifying its position in the initial frame. The correlation filters predict the maximum filter response by learning a least-squares classifier in the region of interest of the target where the maximum response map corresponds to the location of the target [23]. The key to these filter successes is exploiting all the negative training data by including all shifted versions of the training patches. Moreover, this kind of method mostly follows the tracking-by-detection paradigm where each frame of the target is regarded as a detection problem. The classifier is trained to distinguish the target object from its background.

Bolme et al. proposed a correlation filter, a minimum output sum of square errors (MOSSE) for target appearance, based on a single luminance channel in visual tracking [23]. Ma et al. suggested that multiple correlation filters can be learned on hierarchical convolution layers [102]. Semantic and spatial information was captured by constructing multiple DCFs on low, middle and higher convolution layers.

By exploiting the circulant structure for training samples, Heriques et al. proposed to derive closed-form solutions for training and detection with several types of kernels, including Gaussian and polynomial kernels [15]. Zhang et al. incorporated spatio-temporal contextual relationships between the target of interest and its local context in a Bayesian framework; this aid to model the statistical correlation between low-level features from the target and outside regions for visual tracking [160].

Spatially regularised discriminative correlation filters (SRDCFs) introduced spatially regularised components in learning by alleviating the unwanted boundary effects by penalising correlation filter coefficients depending on the spatial location [45]. Qi et al. proposed an algorithm to fuse several DCFs through an adaptive hedged method. Staple-complimentary learners for real-time tracking combined DCF and complimentary cues like colour histograms to achieve a real-time state-of-the-art tracking model. DCF with a channel and spatial reliability (CSR-DCF) introduced a spatial reliability map in filter learning and the updating process.

Continuous convolution operators for visual tracking (C-COT) adopted a training of continuous-domain convolution filters in learning DCF and integration of multi-resolution deep feature maps, leading to top performance on several tracking benchmarks [46]. The enhanced version of C-COT is ECO (efficient convolutional operators for tracking), which improved both speed and performance by introducing several efficient strategies.

ECO is a fast DCF implementation with factorised convolutional operators to reduce the number of parameters in the model [37]. For achieving tracker robustness, *ensemble* based feature fusion methods employing dynamic programming have been used [22]. Though dynamic programming-based tracking methods' computational efficiency was determined by the number of trackers in the ensemble; it is reduced if this number increases. Also, sparse representation-based methods suffered from low frame rates due to the fact that the overall speed being decided by the slowest tracker in the ensemble.

For a *multi-expert entropy minimisation* (MEEM) algorithm, based on exploiting the relation between current tracking and historical samples using an entropy minimization concept, see [158]. A discrete graph-based optimised framework is an extended version for MEEM [94].

Target state prediction was made based on a framework using a partition fusion method to group an ensemble of trackers [77]. In many of the fusion-based methods, the trajectory was calculated twice in the forward and backward direction which makes the tracker run twice. Also, tracker results were not fed back to individual trackers that make a transient drift accumulate and build-up tracking errors.

The *multi-cue correlation filter tracker* (MCCT) employed a decision level fusion strategy and a robustness evaluation criterion for ranking experts in order of their reliability. It employed a training-sample-sharing strategy and adaptive expert updates. The tracker output was fedback to the individual experts to prevent expert corruption over time. We decided to use an MCCT baseline framework for our work and propose an additive saliency feature (in short: *sal-feature*) that will benefit the tracking robustness.

A novel saliency guided DCF-based ensemble tracker is proposed in our work [4]. Improved tracking results are obtained by incorporating saliency alongside other hand-crafted and deep network features in baseline ensemble tracking framework.

2.8.1 Discrete Correlation Filter Basics

As announced above, we provide some basics for DCF. An image patch $\mathbf{x}$ of size $M \times N$ has its centre on the target. Training samples are generated from circular shifts along M and N dimensions. Shifted samples of $\mathbf{x}$, denoted as $\mathbf{x}(m, n)$, are a subset of the rectangular grid $\{0, 1, ..., M-1\} \times \{0, 1, ..., N-1\}$ and used as training samples, with a Gaussian function label of $\mathbf{y}(m, n)$. A correlation filter $\mathbf{w}$ of the same size as $\mathbf{x}$ is learned by minimising the regression problem

$$\min_{\mathbf{w}} ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \lambda||\mathbf{w}||_2^2 \tag{2.16}$$

$\mathbf{X}$ is the data matrix obtained by concatenating all the circular shifts for the training patches. λ is the regularization parameter $(\lambda \geq 0)$.

For $d \in \{1, ..., D\}$, the learned filter weight on the d^{th} channel equals

$$\mathbf{w}_d^* = \frac{\hat{y} \odot \hat{\mathbf{x}}_d^*}{\sum_{i=1}^{D} \hat{\mathbf{x}_\mathbf{i}}^* \odot \hat{\mathbf{x}_\mathbf{i}} + \lambda} \tag{2.17}$$

where operator $\odot$ is the Hadamard or element-wise product. The hat symbol denotes the *discrete Fourier transform* (DFT) for the vector and symbol $*$ denotes the conjugate complex for that vector. The response map R for z is as follows:

$$\mathbf{R} = \mathcal{F}^{-1}\left(\sum_{i=1}^{D} \hat{\mathbf{w_d}} \odot \hat{\mathbf{z_d}}^*\right) \tag{2.18}$$

where $\mathbf{z}$ is a ROI patch of the same size as $\mathbf{x}$, cropped in the next frame to track the target of interest. The target is localised where the response is maximised.

Numerator and denominator are updated online for the filter $\hat{\mathbf{w_d}}^*$ as follows:

$$\hat{\mathbf{A}}_d^t = (1-\eta)\,\hat{\mathbf{A}}_{t-1}^d + \eta\hat{\mathbf{y}} \odot \hat{\mathbf{x}_d^*}^t \tag{2.19}$$

$$\hat{\mathbf{B}}_d^t = (1-\eta)\,\hat{\mathbf{B}}_{t-1}^d + \eta\sum_{i=1}^{D} \hat{\mathbf{x}_i^*}^t \odot \hat{\mathbf{x}}_i^t \tag{2.20}$$

During the filter learning step, a Hann window [63] is applied to avoid the boundary effect problem. The learning rate is adjusted by parameter η for the frame index t and the scale is selected based on the tracker methodology described in [40]:

$$\hat{\mathbf{w}_d^*}^t = \frac{\hat{\mathbf{A}}_d^t}{\hat{\mathbf{B}}_d^t + \lambda} \tag{2.21}$$

2.9 Summary

This chapter summarises the basic concepts and techniques that are used for this study at different stages.

ROI detection, and tracking of objects of interest based on these detections, is very vital step. Tracking performance is highly correlated with the quality of extracted ROIs or detected objects.

Various multiple and visual object tracking concepts are described in this chapter that form a baseline for understanding the novelty of further work. Chapter-wise further specific detailed explanations are provided in the the following.

Chapter 3

Improved Object Recognition in Warehouses

Deep convolution neural nets have an inherent ability to extract features automatically and are used for accurate category classification. To propose a prototype for improved video-based object detection and classification for warehouses, we aim to extract moving foregrounds by using the GMM technique. We improve our findings by incorporating a pixel saliency map. Finally, we assign labels to the foreground using a pre-trained Inception-v3 deep learning detector and classifier.

3.1 Datasets

We acquire warehouse video data in two phases. In phase 1, we record a video dataset by using simple RGB go-pro monocular cameras, fixed at various locations

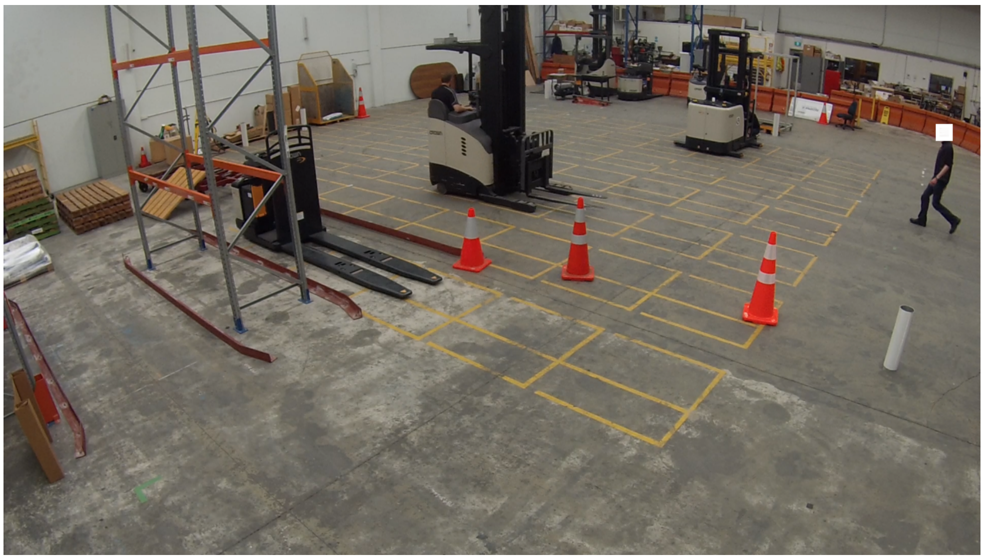


Figure 3.1: Go-pro camera viewpoint 1 for first phase

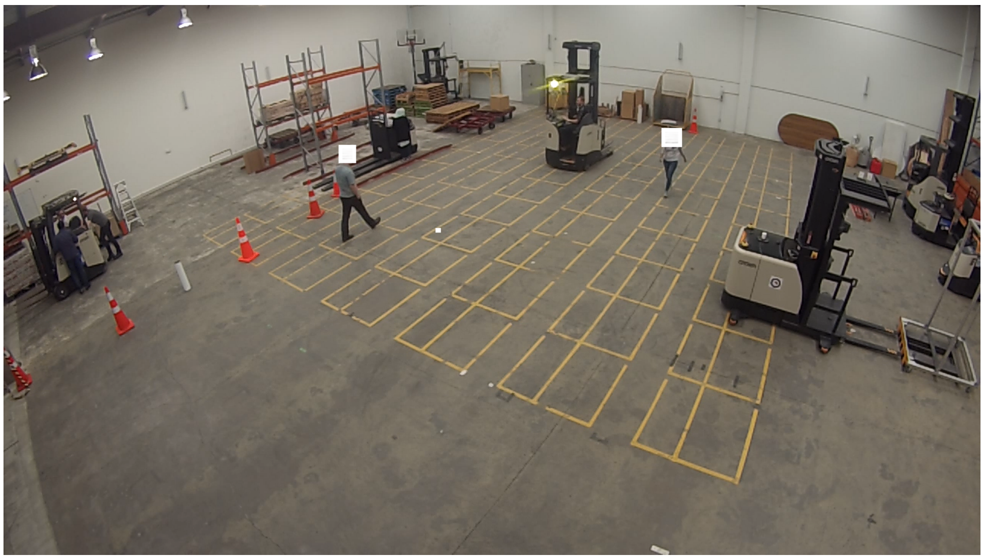


Figure 3.2: Go-pro camera viewpoint 2 for first phase

in an experimental warehouse, to catch the warehouse activities independently.

Several scenarios are modelled and recorded at the site. Each such camera captures images of size $[W \times H] = [1280 \times 720]$ pixels at a rate of 25 images per second.

The captured scenarios also include warehouse blind spot modelling with various vital forklift and pedestrian poses. Refer to Figs. 3.1 and 3.2 for scenes showing two of the camera viewpoints.

In a second phase, a very diverse data set is recorded, with challenging warehousing scenes. We collect data from four different viewpoints with varying levels of background complexity, different lighting conditions, un-occluded and occluded views of pedestrian and truck activities, single-agent (pedestrian/truck) and multi-agent activity.

In this phase, we recorded 160 videos at a frame rate of 29 frames/images per second. Each of the captured images is of resolution $[W \times H] = [2048 \times 1536]$ pixels. The duration of each captured video is 10 minutes.

The dataset is captured by four *point-tilt-zoom* (PTZ) dome monocular cameras,

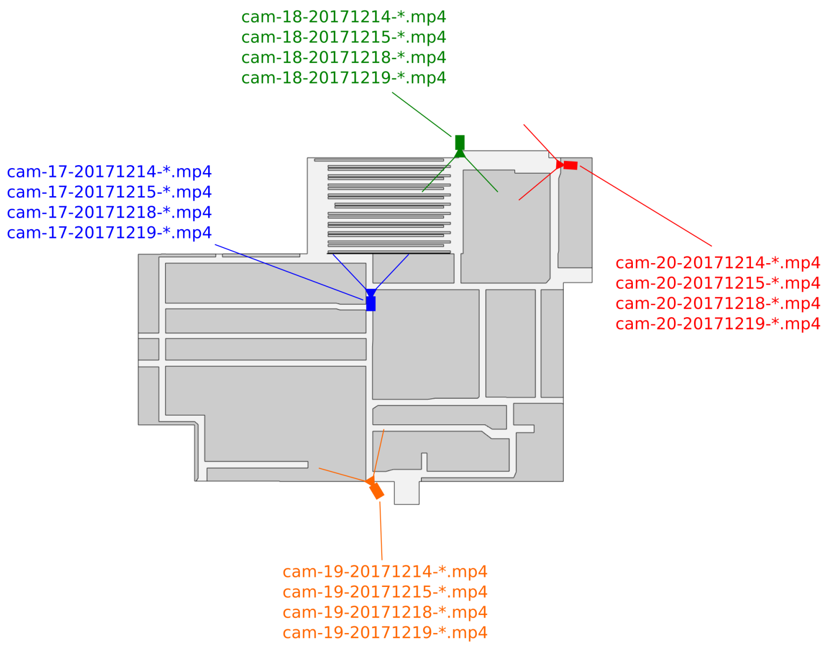


Figure 3.3: PTZ monocular camera layout positions for Phase 2 of data acquisition.

fixed in one of the production warehouses, to monitor the warehousing activities closely. Fig. 3.3 depicts the positioning of the four different cameras in the warehouse. Fig. 3.4 refer to the two warehouse scenes from recorded data in phase 2.

For the work reported in this chapter, for training and experiments, we mostly employ data recorded in Phase 1. Data recorded in Phase 2 are extensively used in later chapters.

3.2 Warehouse Scene Challenges

As per our observations, these particular indoor scenes come with the following environmental challenges:

1. There are multiple moving objects. We have recordings with pedestrians and forklift trucks moving inside a warehouse.

2. Colour contrast between background and foreground is very marginal, most of the time.
3. Multiple occlusions are likely; the environment is semi-cluttered.
4. There are parked forklifts (stationary objects) in some areas.
5. Changes in loads occur frequently for racks in the background; typically these changes are gradual.
6. Illumination changes are also gradual. Warehouse indoor data have usually only a few low-illumination areas.
7. There are entries and exits of vehicles into a scene.
8. When the objects are *static* (i.e. no movement between two subsequent frames) or moving only slightly then they are often wrongly classified as part of the background.

To model object appearance in warehouses, a study is performed for various cues or features that can be employed for targets. For the localisation of salient objects in warehouses, SIFT and saliency-based methods were used [110]. Table 3.1 enlists the studied observations regarding stated feature schemes. We conclude that an adaptive detection algorithm is needed which can detect pedestrians working behind the warehouse clutter, which are often occluded and challenging to detect.

Due to forklift trucks with loading and unloading pallets, people working in aisles might be occluded partially with various poses. It is vital to handle illumination variations, repetitive motions from clutter, and long-term scene changes. A

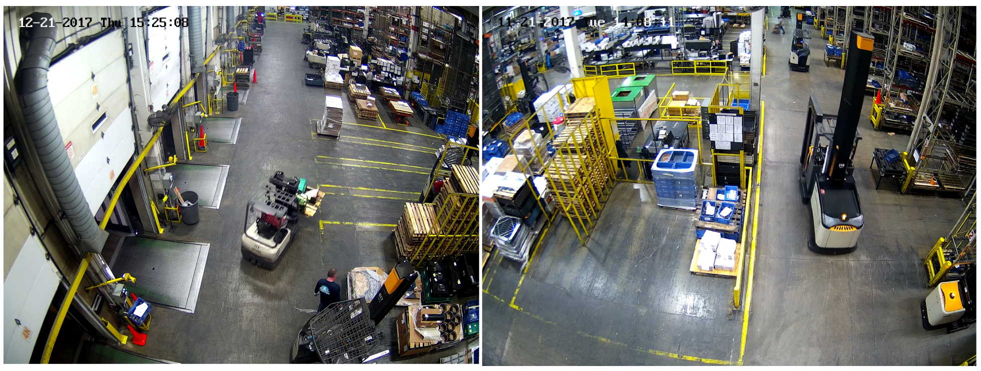


Figure 3.4: Video shots from the recorded dataset at Phase 2.

Table 3.1: Examples of appearance modelling schemes.

Features	Challenges
Gradient-based feature	Can encode target shape robust to illumination variations but cannot handle occlusions and shape deformations
Colour-based feature	For instance colour-histogram is discriminative but spatial contents of the image is ignored
Local features	Out of plane rotation is a challenge but robust to shape deformation (optic flow, KLT)
Region features	Involve wider region but computationally expensive (co-variance matrix) .
Depth feature	Depth measurement needs multiple views for the same scene for depth computation.

GMM based approach is chosen to detect pixel-wise moving segmented foreground and background regions for target detection and onward tracking.

3.3 Object Tracking Pipeline

Deep convolution architectures, employing automatic feature learning and classification, are being researched in recent years. Out of many options, *region-based CNNs* (R-CNNs), or later versions are frequently explored. An R-CNN is a three-stage pipeline process. Features are extracted for every object proposal in an image and are being cached. *SVM* is used as object detector, replacing the *softmax classifier*. In the third stage of training, *bbox regressors* are learned [54]. *Spatial pyramid pooling networks* (SPPnets) were introduced to speed up R-CNNs by sharing the computation burden [59]. This is also a multi-stage process which computes a convolutional feature map for the whole input image and then classifies each object proposal, using a feature vector extracted from the shared feature map.

Fast R-CNNs use single-stage training and a multi-task loss for better detection accuracy and speed. Training can update all the layers of the network at once, and no feature caching is required [111]. These nets still used selective search for region generation; now removed in *faster R-CNNs* came. A cost-free *region proposal network* (RPN) was employed which predicted potential object bounds and an object score at each position in the faster R-CNN. This RPN, integrated with fast RCNN, was trained to share features across layers [111].

Inspired by this work we propose a pipeline for object detection.

First, we select background subtraction, which is well suited for moving targets as in our case. We follow [115] where each pixel value is modelled as a *mixture of Gaussians components*. By this means we can determine whether or not a pixel is part of the background. This supports an effective approach for separating the background from the foreground.

Second, we need to improve extracted foregrounds as they are not yet accurate. See Figs. 3.7 to 3.12. Due to a low background pixel recovery rate and a slow adaptation to scene changes when using the traditional GMM algorithm, foreground quality is not yet fair. We extracted salient pixels using a local contrast method [154], based on a visual saliency map. A pixel-wise saliency map, for each frame, is used to improve the corresponding foreground obtained by the GMM background extraction.

Third, we want to label the detected foreground. We can repurpose features, extracted from a pre-trained deep convolution neural network, for new object category recognition specific to our application. This technique is called *transfer learning*. We transferred the learned features from a pre-trained model (i.e. *Google Inception Model*) for new category classification which are forklifts and pedestrians in a typical warehouse scenario [74, 122]. Refer to Fig. 3.6 for the architectural diagram for the inception-v3 model.

Fourth, we use a Kalman filter to track the resulting targets. Fig. 3.5 refer to the proposed multiple object tracking framework.

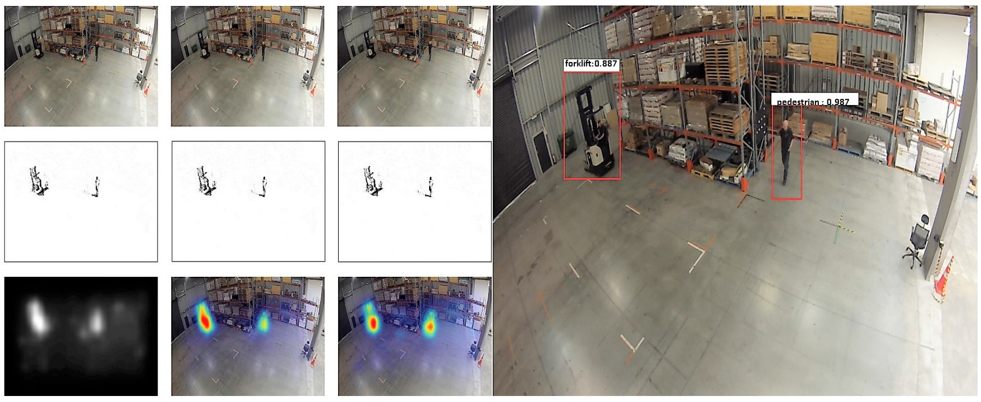


Figure 3.5: Proposed object tracking framework.

3.4 Gaussian Mixture Model for Foreground Extraction

The Gaussian mixture model is a natural choice for our analysis (i.e. for extracting moving targets out of a mostly stationary background). Mixture models are proba-

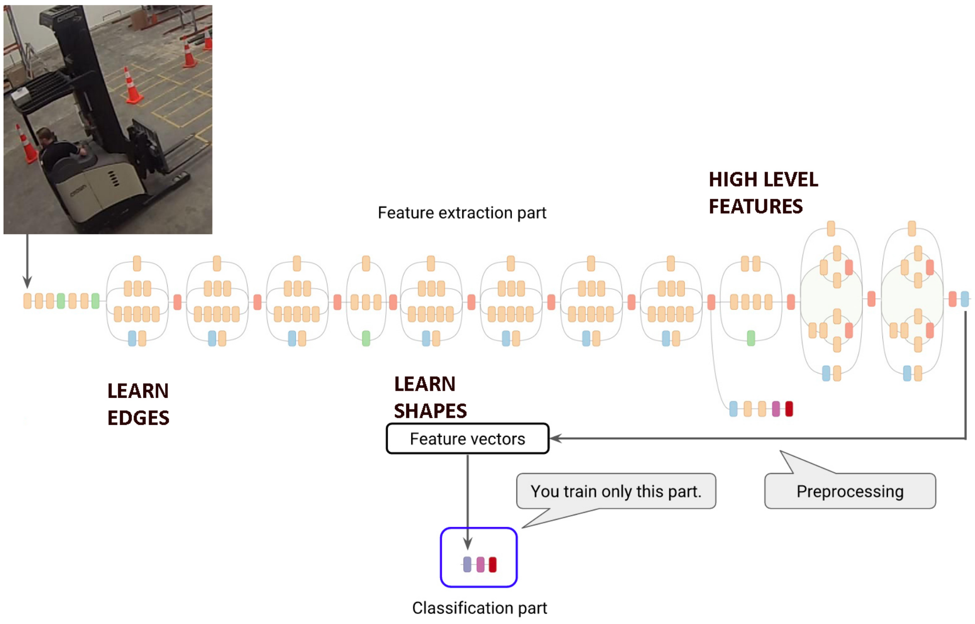


Figure 3.6: Inception-v3 [122].

bilistic models which assume that underlying pixels belong to a particular mixture distribution. To make the model more robust to lighting variations, and to handle multiple surfaces occurring in the view frustum of particular pixels, the mixture models need to be adaptive.

Qualitative Analysis for a Standard GMM Approach

As can be seen in the update equations above, α is the first learning rate. It needs to be adjusted as per the scenario conditions. To incorporate slowly moving objects and large homogeneously coloured objects, we kept α small. For scenarios which are changing quickly, it needs to be larger to adapt to the scene. ρ is the second learning rate. Usually, it is assigned a much smaller value than α. But, as per our trial experimentation, the use of the second learning rate increases the required computation time. Initial mean and variance are adjusted as per the scenario results. The thresholds are the same (value 0.9) for all the experiments.

We apply the mixture of Gaussian algorithm [115] with the following parameter setting: $\alpha = 0.001$ ranging to 0.79, $\rho = 0.00001$, threshold $T = 0.9$, the number of Gaussian components $k = 3$. We obtained our results by using Matlab 2017a.

Refer to Figs. 3.7 to 3.12, for the results obtained after applying the proposed steps.

For lower alpha values, slowly moving objects are detected with good quality foreground, but for higher values, results are not good for the same object.

GMM cannot deal with sudden illumination changes and camera movements. This is as shown in Fig. 3.10. With passage of time, the variance decreases for more stable pixels. If the variance becomes too small, then even camera noise is marked as foreground pixel that effects the foreground quality.

Bigger objects, uniform in colour or slowly moving, are sometimes incorporated into background for a few frames. In conclusion, our extensive experiments, here illustrated by a few examples, lead to the conclusion: *We need some improvement in foreground detection, which makes it more robust to the mentioned challenges.*

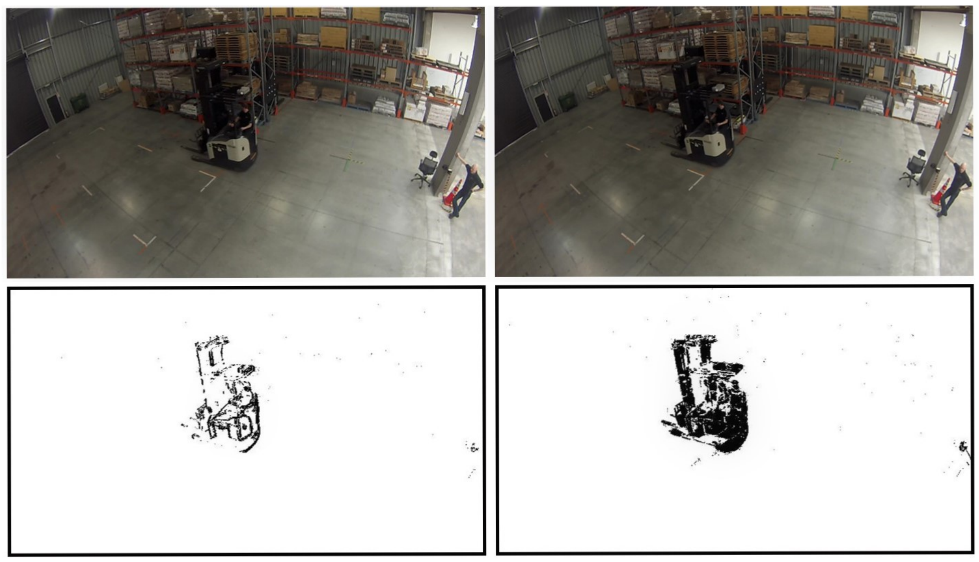


Figure 3.7: Forklift crossing low illumination area. Foreground results in bottom left and right images, after applying GMM. The pedestrians on the right are detected poorly due to a more static posture.

3.5 Pixel Saliency for Foreground Improvement

Due to the stated observations above, we improved GMM-based foreground detection using a *saliency map-based foreground extraction scheme*.

We observed that pixel-based *saliency map* values, generated by the method of [154], can be useful for improving the average foreground quality obtained from the GMM method. It was computationally fast. It took 0.5 seconds or less per frame to compute a saliency map. Visual saliency maps are able to mark salient pixels in the images and have good results with occluded objects in warehouse scenes.

Compared with the global contrast method, the local contrast is a relatively better cue to be combined with the compactness cue. Local contrast methods can identify the foreground region [60], but they have a limitation that they identify visible object boundaries rather than all the area. This effect can be minimized by the propagation of saliency information based on diffusion [60].

To construct the pixel saliency map for the image, we converted it into a super-

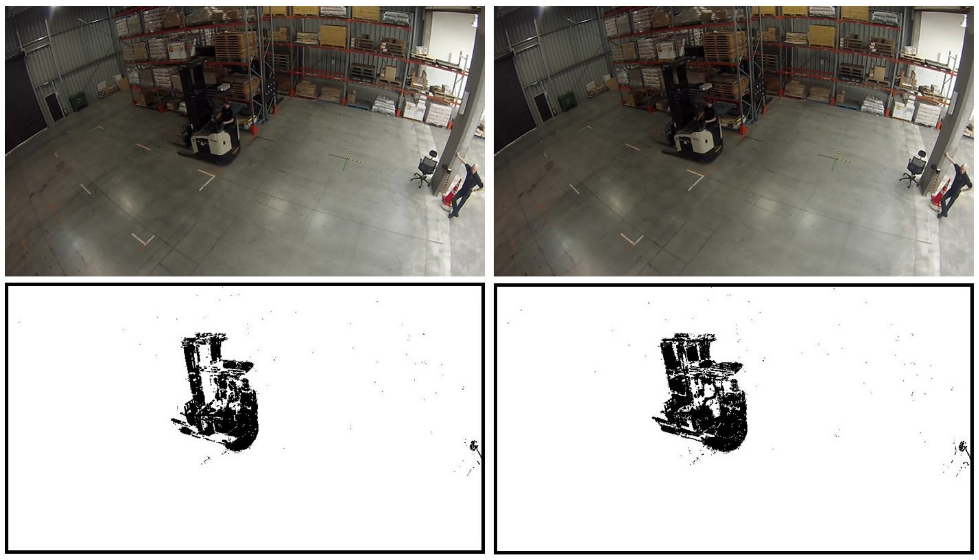


Figure 3.8: A forklift crossing low illumination area. Frame 15 and Frame 20 foreground results in bottom left and right images, after applying GMM. Very slow recovery of background pixels have made the area of forklift foreground bigger than actual.

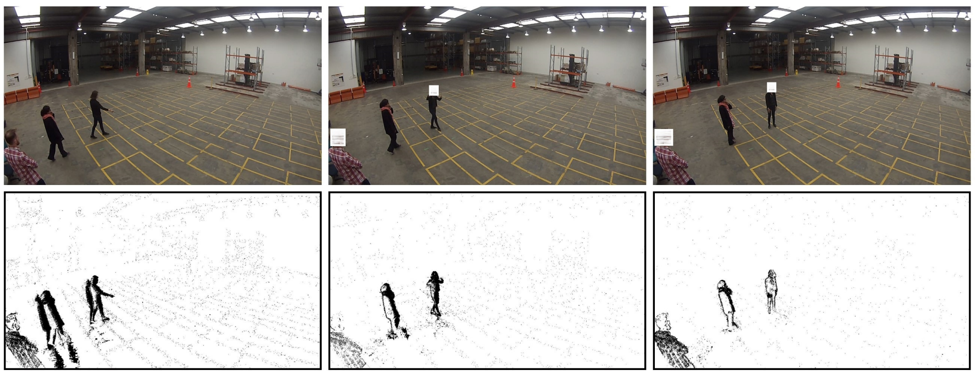


Figure 3.9: Scenario of three pedestrians standing inside warehouse. Starting from bottom extreme left, foreground is somewhat distorted. Some foreground pixels from previous frames are still there. For bottom middle, one of the pedestrian, who is more static, is poorly detected in foreground.

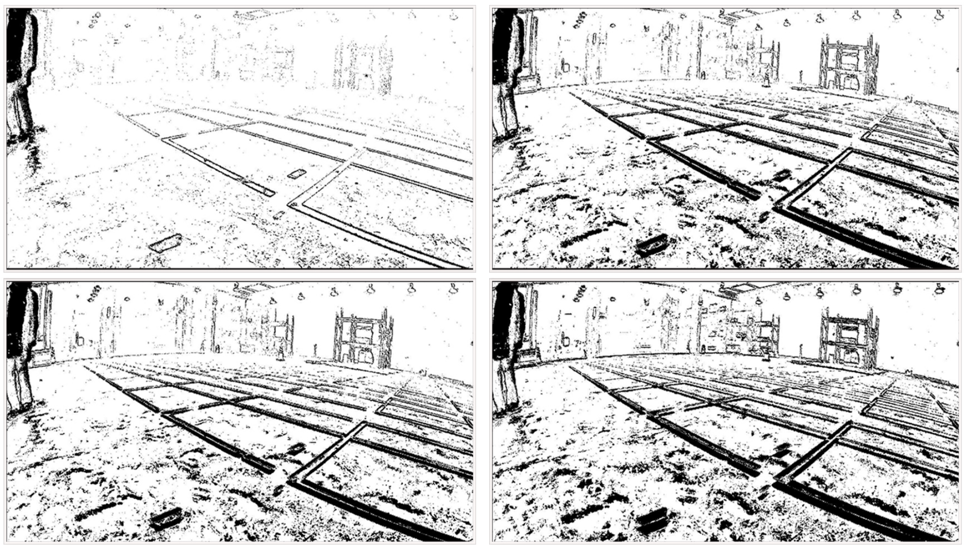


Figure 3.10: GMM results in a sudden camera movement scenario. A pedestrian is detected at the left, standing static in a warehouse scene. Due to variance changes in pixels due to camera motion, most of the background pixels appeared as foreground pixels. See upper right, bottom right, and left images.

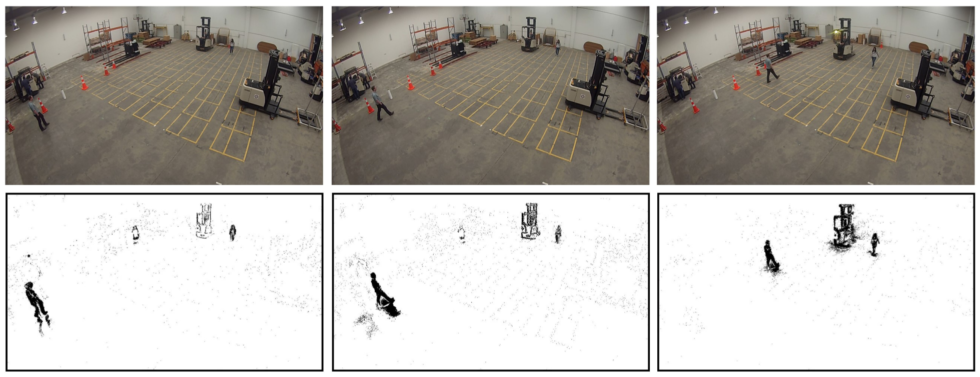


Figure 3.11: A scenario in which three pedestrians and a forklift are detected. One of the three pedestrians is not detected in the bottom right-most result. This pedestrian was having a slightly static posture, so became part of the background pixels.

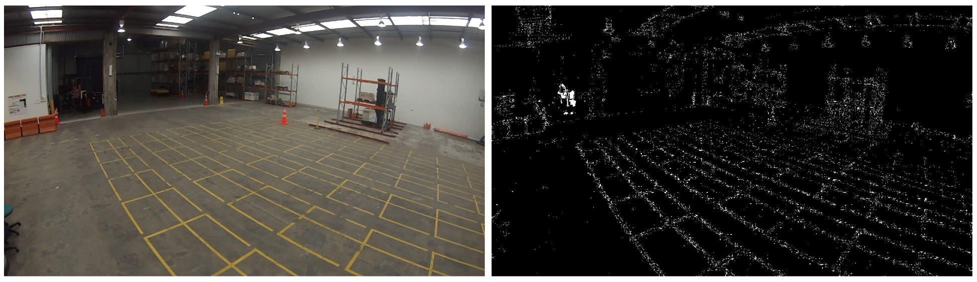


Figure 3.12: GMM results for occluded pedestrians. Two occluded pedestrians shown with clustered white pixels.

pixel representation for constructing a resultant graph. We used SLIC [10] for an abstract graph representation of an image. Each superpixel, generated by SLIC, corresponds to some node. There are three parameters used in here: The number N of superpixel nodes used in Simple Linear Iterative Clustering (SLIC), σ^2 which controls the fall-off rate of the exponential function, and α which balances the fitting constraints of manifold ranking algorithms. We experimentally set the parameters

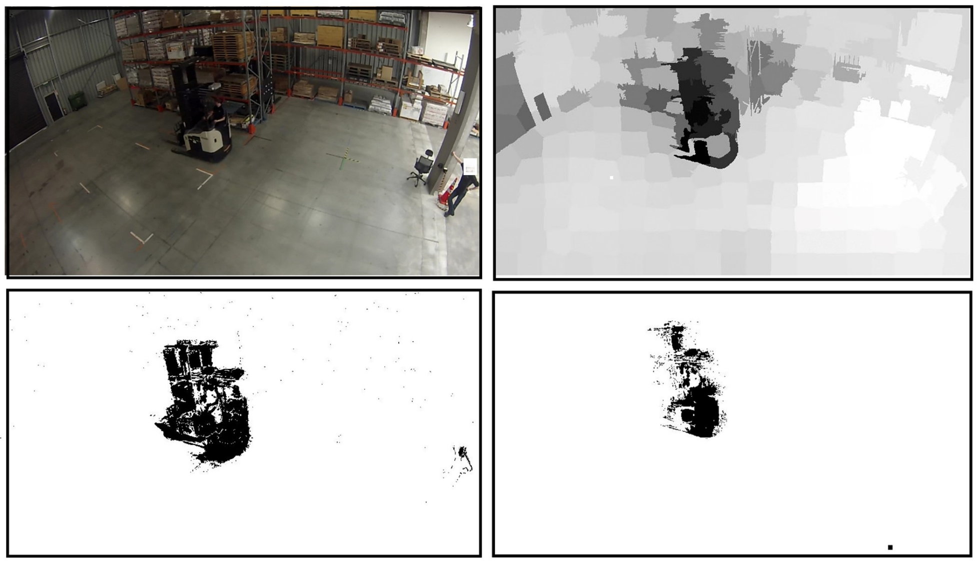


Figure 3.13: Occluded forklift, foreground improvement based on pixel saliency, without morphological improvement.

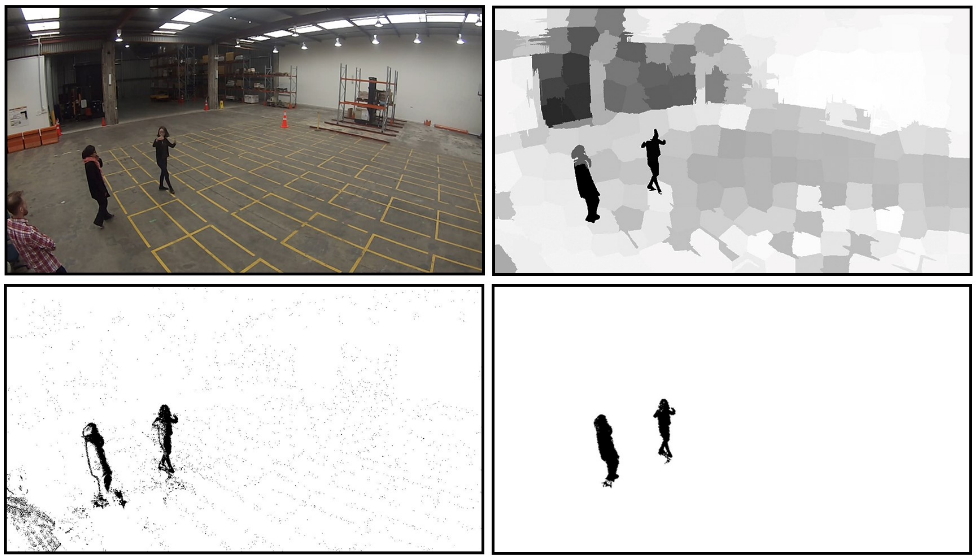


Figure 3.14: Improved foreground based on pixel saliency. One of the pedestrians is missed, due to poor visibility both in GMM result and its saliency map.

to $N = 200$, $\sigma^2 = 0.1$, and $\alpha = 0.99$ for experimentation. Next, the two saliency maps are computed based on the compactness visual cue and local contrast [154].

The resulting saliency maps are propagated using a diffusion process and the constructed graph later. Thus, a pixel-wise saliency map is generated from two computed maps. This pixel-wise saliency map for the specific frame is binary thresholded. We apply logical operations between salient binary thresholded pixels and moving pixels from GMM. Finally, some morphological processing is used to generate improved foreground masks.

See Fig. 3.13 and Fig. 3.14, for the improvement in foregrounds, as per the proposed improvement foreground strategy. It can be seen that foreground is better in quality with less redundant pixels from the background, as part of the foreground. See Fig. 3.13. Fewer foreground holes are present in Fig. 3.14.

3.6 Deep Learning for Object Classification

After obtaining our improved foregrounds based on pixel saliency, we aim at classification. We use a pre-trained model of Google's Inception v3 [122] and re-train the top layer, for new categories.

We aim at overcoming the deficiency of the training data and limitations of computation time or resources by adapting a classifier, trained for other categories, to our dataset.

Basics and a Pre-trained Network

CNNs have a reduced number of parameters and connections compared to the same-sized feed-forward networks. These characteristics make it easier to train and test them. They have successfully been used to categorize images and activities or tasks once trained with an excess of data samples.

We can use a pre-trained convolutional deep learning model for classification tasks of new categories. We can retrain final or more layers of this model, and adapt it to our new categories and to the limited dataset available to us. This approach is called *transfer learning*. We select a pre-trained model architecture, replace the top layer by a new layer, and adapt the newly added layer to our own data classification task. We selected Google's v3 architecture model for moving object classification in warehouse scenes into two categories, either forklift or pedestrian.

Google's Inception v3 [122] is an architecture that provides a good-performance network with relatively low computational costs. To measure the classification accuracy, there are two main measurements used in the deep learning literature [122]; this is the top-5 error rate and the top-1 error rate. They measure the rate at which the architecture fails to include the correct class in the top-5 and the top-1 output, respectively. Refer to 3.15 for the basic inception modular diagram.

Inception v3 achieved a 5.6 % top-5 error rate, and a 21.2 % top-1 error rate. Another famous architecture model, i.e., AlexNet [74] achieved a top-5 error rate of 15.3 % and Inception (Google Net) [121] achieved 6.67 % in the same category. Thus we selected the Inception v3 model architecture, pre-trained on ImageNet. We added a new Softmax and fully connected layer for training and re-trained it in Tensor Flow 1.0 in the Ubuntu operating system. We re-train the model for classifying our two object categories. The top layer receives as input a 2,048-dimensional vector for each image. Since the Softmax layer contains two labels, this corresponds to learning 4,098 model parameters, corresponding to the learned biases and weights.

Training, Validation and Testing

We prepare our training data set containing 4,000 images, for two object categories of forklifts and pedestrians. We limit the training data to these two categories. Refer to Fig. 3.16 depicting the thumbnails for these two categories of objects of interest. Inception network rescales images to $[W \times H] = [299 \times 299]$ pixels. So these are the input-width and input-height flags for the images. Most of the data we processed are from recorded video clips inside of a selected warehouse, showing different scenarios. Testing, validation percentages can be set by adjusting their flags in the script. We use default values for these, i.e. 80 percent training images and remaining 10 percent each for validation and testing. The script uses image file names for this data splitting. We use Intel Core i7 with 16 GB RAM for re-training the pre-trained model.

First, a calculation of *transfer-values* is performed, for each of the images, arranged in training, testing, and validation sets. 'Transfer value' is the term we use for the output feature values, at the layer just before the final top layer. We used these feature values to differentiate the objects for new categories. Since each image is reused many times during training and calculation, the transfer-values are being cached (stored on disk), to be reused repeatedly for training, validation, and testing [129]. Once the transfer value computation is complete, the actual training of the top layer of the network begins, for new labels, and for each image.

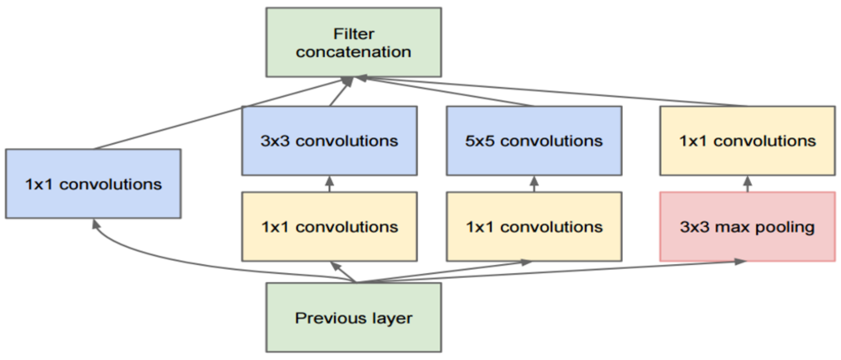


Figure 3.15: A basic Inception module.

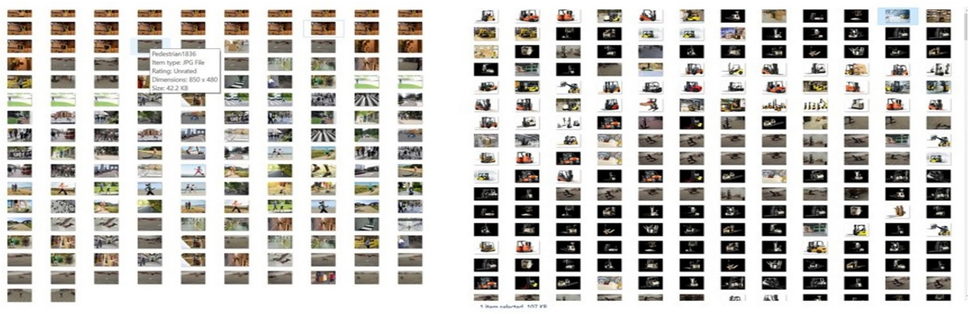


Figure 3.16: Training data with two categories: forklifts and pedestrians.

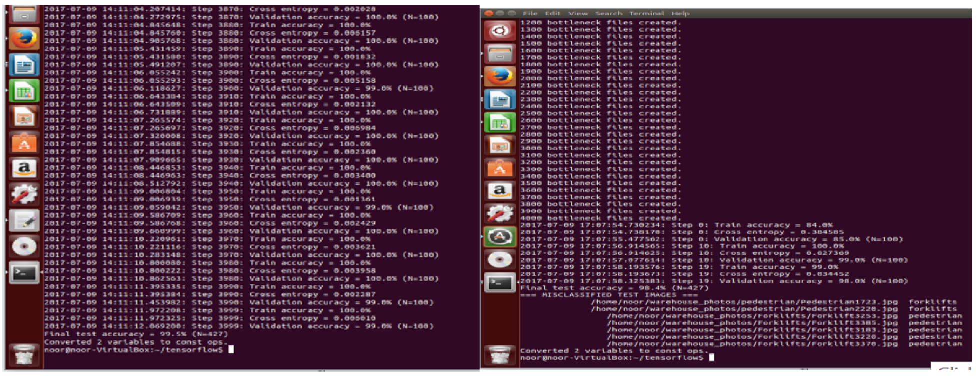
We have 4,000 training steps. Each step chooses ten images at random from the training set, finds their transfer values from the cache, and feeds them into the final layer to get predictions for the category. Those predictions are then compared against the actual labels, i.e. pedestrians or forklift, to update the top layer's weights through the *back-propagation process*. We obtain the following measures for each training epoch.

The *training accuracy* shows what percentage of the images, we used in the present training batch, was labelled correctly with the true class.

The *validation accuracy* is the percentage on a randomly-selected set of images from a different data set, which is not present in the training set. If the training accuracy is very good, but the validation accuracy is not, that means that the network is *over fitting* to training data, and we need more training samples for network generalisation.

Table 3.2: Results after 4,000 training iterations.

Accuracy (in percent)	4,000 steps
Training	98
Validation	96
Cross entropy	0.12
Test	99.5



```
2017-07-09 14:11:04.207414: Step 3870: Cross entropy = 0.002028
2017-07-09 14:11:04.272975: Step 3870: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:04.845648: Step 3880: Train accuracy = 100.0%
2017-07-09 14:11:04.845760: Step 3880: Cross entropy = 0.006157
2017-07-09 14:11:04.905768: Step 3880: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:05.431459: Step 3890: Train accuracy = 100.0%
2017-07-09 14:11:05.431580: Step 3890: Cross entropy = 0.001832
2017-07-09 14:11:05.491207: Step 3890: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:06.055242: Step 3900: Train accuracy = 100.0%
2017-07-09 14:11:06.055293: Step 3900: Cross entropy = 0.005158
2017-07-09 14:11:06.118627: Step 3900: Validation accuracy = 99.0% (N=100)
2017-07-09 14:11:06.643384: Step 3910: Train accuracy = 100.0%
2017-07-09 14:11:06.643509: Step 3910: Cross entropy = 0.002132
2017-07-09 14:11:06.731889: Step 3910: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:07.265574: Step 3920: Train accuracy = 100.0%
2017-07-09 14:11:07.265697: Step 3920: Cross entropy = 0.006984
2017-07-09 14:11:07.320008: Step 3920: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:07.854688: Step 3930: Train accuracy = 100.0%
2017-07-09 14:11:07.854815: Step 3930: Cross entropy = 0.002360
2017-07-09 14:11:07.909665: Step 3930: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:08.446853: Step 3940: Train accuracy = 100.0%
2017-07-09 14:11:08.446963: Step 3940: Cross entropy = 0.003400
2017-07-09 14:11:08.512792: Step 3940: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:09.006804: Step 3950: Train accuracy = 100.0%
2017-07-09 14:11:09.006939: Step 3950: Cross entropy = 0.001361
2017-07-09 14:11:09.059042: Step 3950: Validation accuracy = 99.0% (N=100)
2017-07-09 14:11:09.586709: Step 3960: Train accuracy = 100.0%
2017-07-09 14:11:09.586768: Step 3960: Cross entropy = 0.002429
2017-07-09 14:11:09.660999: Step 3960: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:10.220961: Step 3970: Train accuracy = 100.0%
2017-07-09 14:11:10.221116: Step 3970: Cross entropy = 0.003621
2017-07-09 14:11:10.283148: Step 3970: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:10.800080: Step 3980: Train accuracy = 100.0%
2017-07-09 14:11:10.800222: Step 3980: Cross entropy = 0.003958
2017-07-09 14:11:10.862563: Step 3980: Validation accuracy = 100.0% (N=100)
2017-07-09 14:11:11.395335: Step 3990: Train accuracy = 100.0%
2017-07-09 14:11:11.395384: Step 3990: Cross entropy = 0.002287
2017-07-09 14:11:11.453982: Step 3990: Validation accuracy = 99.0% (N=100)
2017-07-09 14:11:11.972208: Step 3999: Train accuracy = 100.0%
2017-07-09 14:11:11.972325: Step 3999: Cross entropy = 0.006010
2017-07-09 14:11:12.069200: Step 3999: Validation accuracy = 99.0% (N=100)
Final test accuracy = 99.5% (N=427)
Converted 2 variables to const ops.
noor@noor-VirtualBox:~/tensorflow$
```

```
File  Edit  View  Search  Terminal  Help
1200 bottleneck files created.
1300 bottleneck files created.
1400 bottleneck files created.
1500 bottleneck files created.
1600 bottleneck files created.
1700 bottleneck files created.
1800 bottleneck files created.
1900 bottleneck files created.
2000 bottleneck files created.
2100 bottleneck files created.
2200 bottleneck files created.
2300 bottleneck files created.
2400 bottleneck files created.
2500 bottleneck files created.
2600 bottleneck files created.
2700 bottleneck files created.
2800 bottleneck files created.
2900 bottleneck files created.
3000 bottleneck files created.
3100 bottleneck files created.
3200 bottleneck files created.
3300 bottleneck files created.
3400 bottleneck files created.
3500 bottleneck files created.
3600 bottleneck files created.
3700 bottleneck files created.
3800 bottleneck files created.
3900 bottleneck files created.
4000 bottleneck files created.
2017-07-09 17:07:54.738234: Step 0: Train accuracy = 84.0%
2017-07-09 17:07:54.738170: Step 0: Cross entropy = 0.384585
2017-07-09 17:07:55.477562: Step 0: Validation accuracy = 85.0% (N=100)
2017-07-09 17:07:56.914565: Step 10: Train accuracy = 100.0%
2017-07-09 17:07:56.914625: Step 10: Cross entropy = 0.027369
2017-07-09 17:07:57.077614: Step 10: Validation accuracy = 99.0% (N=100)
2017-07-09 17:07:58.193576: Step 19: Train accuracy = 99.0%
2017-07-09 17:07:58.193673: Step 19: Cross entropy = 0.034452
2017-07-09 17:07:58.325381: Step 19: Validation accuracy = 98.0% (N=100)
Final test accuracy = 98.4% (N=427)
=== MISCLASSIFIED TEST IMAGES ===
              /home/noor/warehouse_photos/pedestrian/Pedestrian1723.jpg  forklifts
              /home/noor/warehouse_photos/pedestrian/Pedestrian2228.jpg  forklifts
                 /home/noor/warehouse_photos/Forklifts/Forklift3253.jpg  pedestrian
                 /home/noor/warehouse_photos/Forklifts/Forklift3385.jpg  pedestrian
                 /home/noor/warehouse_photos/Forklifts/Forklift3183.jpg  pedestrian
                 /home/noor/warehouse_photos/Forklifts/Forklift3228.jpg  pedestrian
                 /home/noor/warehouse_photos/Forklifts/Forklift3378.jpg  pedestrian
Converted 2 variables to const ops.
noor@noor-VirtualBox:~/tensorflow$
```

Figure 3.17: Ubuntu training shots for transfer learning inception v3 architecture with warehouse data.

The *cross entropy* is a loss function which evaluates how well the learning process for the model is executing. The training's objective is to make the loss as small as possible. If it is growing lower with each *epoch*, we assume that learning is progressing satisfactorily.

As the process continues, we also observe an improvement in accuracy, i.e. in *test accuracy*, the evaluation which is run on a group of images which are kept separate from the training and validation images. This test accuracy is the best estimate of how well the trained model will work for a specified new classification task. This accuracy is based on the percentage of the images in the test set, that is given the correct label after the model is fully trained. We achieved very good testing accuracy of 99.5 % after 4,000 training iterations. Refer to Fig. 3.17 for shots showing the ubuntu terminal screen, with model training finished..

Results

See Table 3.2 for training, validation and test accuracy after 4,000 training iterations. Cross entropy is also listed.

We tested on images that the model has not been trained for, to check how does it generalize to unseen forklift and pedestrian images. See Fig. 3.18 and Fig. 3.19 for the category % assignment by the model. Figures illustrate our general finding that the model predicts the class with acceptable test accuracy for unseen images. Refer to Table. 3.3 for detection evaluation measures noted after 10 and 20 training steps.

Table 3.3: Evaluation measures for 10 and 20 training steps.

Evaluation Measure	10 steps	20 steps
True Positive	197	200
False Positive	1	2
True Negative	217	217
False Negative	1	8
Precision	99.5	99
Recall	94.7	96.2
f1	97	97.6
Accuracy	97.2	97.7

Hyper-Parameters

We have kept the hyper-parameters constant throughout training and testing,

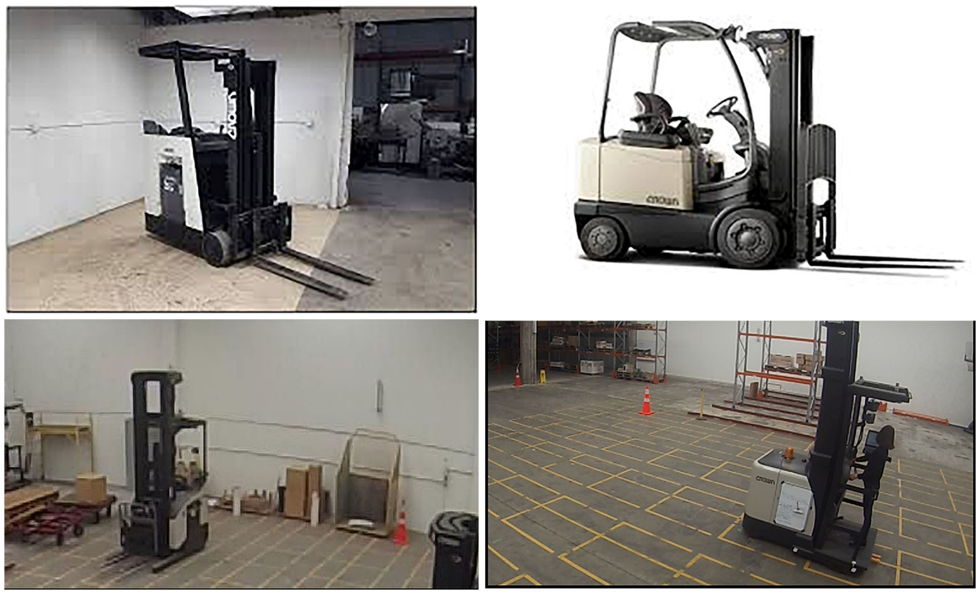


Figure 3.18: Model test accuracy for forklift test images. Upper left: 99.92, upper right: 86.8, bottom left: 46.1, and bottom right: 71.76.

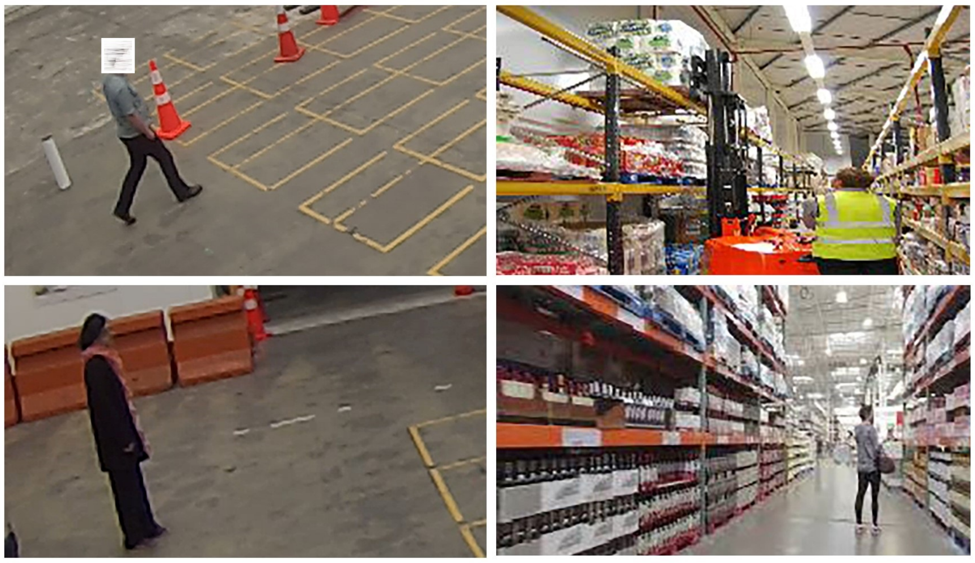


Figure 3.19: Model test accuracy for pedestrian test images. Upper left: 86.7, upper right: 89.23, bottom left: 86.26, and bottom right: 99.5.

with learning rate 0.01. We have edited the hyper-parameter and *print misclassified test images*, to print the evaluation metrics for 20 training iterations for simplicity. These metrics are, true positive (TR), true negative (TN), false positive (FP), and false negative (FN). We define them as follows in our warehouse scenario: *TP* is the number of images categorized as forklifts in testing, *TN* is the number of images

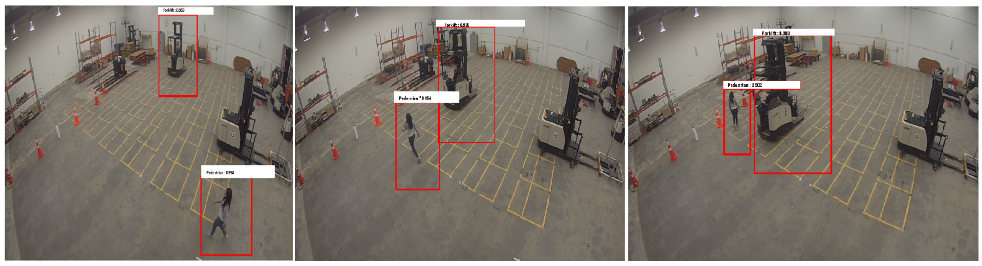


Figure 3.20: Object tracking prototype shown, classifying and tracking the targets of interest.

categorized as pedestrians in testing, *FP* is the number of images categorized as forklifts, although they were pedestrians in real, and *FN* is the number of images categorized as pedestrians, although they were forklifts in real. Refer to Fig. 3.20 for the results, with the proposed tracking prototype. It shows the localised and tracked targets in red bounding boxes, with tests performed on warehouse data acquired in phase 1.

3.7 Summary

For detecting and classifying objects in warehouse scenes, we propose a refinement of a standard GMM method by subsequent use of saliency maps, and the application of a CNN for the final step of object classification for the detected foreground segments.

Using the Inception pre-trained model, with only top-layer re-training, we achieved 99.5 % classification accuracy for the specified task. The model is not able to generalise well for the test images of forklift models for which we have a relatively small number of training images only in our dataset. This can be seen at the bottom-left of Fig. 3.18 where a forklift is misclassified.

Chapter 4

Computing Target Trajectories

Trajectory computation for forklift trucks and pedestrians is of relevance for many warehousing applications. We record a varying range of forklift truck models, frequently occluded in aisles and besides racks. There are various pedestrian activity areas, such as loading docks, who are busy with loading or unloading the pallets. Robust target localisation is essential for seamless tracking results. For localising forklift trucks/pedestrians, we train faster region-based convolution neural network (faster-RCNN) on recorded data. We use detections from the model output to configure a linear Kalman filter (KF) to estimate the trajectories in the pixel co-ordinates. We also improve the forklift truck trajectory by computing pixel saliency maps for the region of interest, detected by faster-RCNN. Our analysis shows that with robust target detection (fewer false positives and false negatives) from our trained network and Kalman-filter-based state correction, tracking results are optimally close to the respective ground truth (labelled by ourselves).

4.1 Introduction

For environments with complex backgrounds as in the case of warehousing (changing object scales and and view angles due to having multiple fixed cameras, background clutter and occlusions), obtaining robust detection results is challenging. It limits most trackers to use only prediction information for *track* (or *trajectory*) prediction for objects of interest. False negatives and false positives are very common in these methods. The resulting track association problem (between detection and target) is challenging.

Since multiple scales and models of the same forklift truck have to be dealt with, detectors based only on shape, appearance, motion, colour or any other cue, may not be sufficient for encoding target object features.

We conduct experiments for tracking objects in single forklift truck scenario sequences as well as in multiple-pedestrian and multiple forklift truck sequences. For single target tracking, we localise a forklift truck bounding box by training a *faster-*

RCNN (i.e. a special deep learning framework). We employ transfer learning with a pre-trained AlexNet model [74]. Detection results are used to initialize a KF [64], used for the prediction of an object's future location in pixel co-ordinates.

Based on localisation coordinate feedback from the faster-RCNN trained network, the KF corrects the predicted trajectory. We also improve the trajectory results by using a *saliency map* of the ROI (as detected by faster-RCNN). Using a ROI saliency map [65], the KF performs state correction for every frame. The resulting trajectory is improved and close to ground truth trajectory.

4.2 Our Tracking Model

We employ a challenging warehouse video dataset recorded in one of the production warehouse, for this work.

We define a forklift truck's current state at frame k as $x_k = (x_k, v_{xk}, a_{xk}, y_k, v_{ky}, a_{ky})$; x_k and y_k are the centroid coordinates for a forklift truck, v_k is the velocity of the window centre, and a_k is the acceleration. The main steps of the algorithm are as follows:

1. For the previous $k-1$ frame, find the *bounding box* for the detected forklift truck using faster-RCNN. A pre-trained AlexNet model is retrained with forklift truck training images involving varying forklift truck models, colours and view angles. We use frames from our pre-recorded data.

2. 2) We calculate the *centroid* of the bounding box for the forklift truck. Using detected centroid from each frame, we initialise a *track*. A track is a structure representing a detected object in the video. The purpose of the structure is to maintain the state of a tracked object.

3. Initialise a KF to predict the centroid of the corresponding track in the current frame k based on the previous initialized state [64].

4. The predicted centroid-location, estimated by KF, is corrected using the corresponding detected centroid for the forklift truck in the current frame.

5. Estimating the trajectory for the collection of frames, showing the path followed by the forklift truck centroid based on KF parameters adjustment and experimentation.

ROI saliency-map-based trajectory improvement

To improve the forklift truck trajectory, we incorporate further improvement steps. We refine the location of a centroid computed by faster-RCNN bounding box criterion, based on ROI pixel saliency map [65, 149]. A centroid computed from faster-RCNN bounding box centre criterion seemed to be less stable. We use improved centroid computation, using the most salient region of the ROI. The points recomputed for trajectory correction by the KF are more consistent and accurate with respect to ground-truth centroid points. Ground-truth centroids are described by pixel co-ordinates for the center of the ground-truth target location. The improvement steps are as follows:

1. We compute a pixel saliency map for the detected ROI for frame k from faster-RCNN [65].
2. We threshold the ROI based on a maximum-saliency area into a *binary threshold saliency map*.
3. We calculate the centroid of the corresponding threshold area.
4. This time, the predicted centroid-location estimate by the KF, is corrected using the corresponding detected centroid points, from a binary threshold area.

Object detection using faster-RCNN

Faster-RCNN [111] is an extension of the fast-RCNN [53] object detection technique. Both techniques use CNN based framework. Refer to Fig. 4.1 for the architecture details.

RCNN [54] and fast-RCNN use a region proposal computation as a pre-processing step before running the CNN. Region proposal algorithms include edge boxes [155] or selective search, which are independent of the CNN. In the case of fast-RCNN, the use of such algorithms becomes the processing bottleneck compared to CNN. Faster-RCNN resolves this issue by implementing the region-proposal mechanism using the CNN, thus making the region proposal a part of the CNN training and prediction steps.

Faster-RCNN proposes a *regional proposal network* (RPN) which shares the fully convolution layers with a fast R-CNN object detection framework. RPN is a fully convolution neural network, which predicts both object proposals and objectness scores. It takes an image (of any size) as input and outputs a set of rectangular

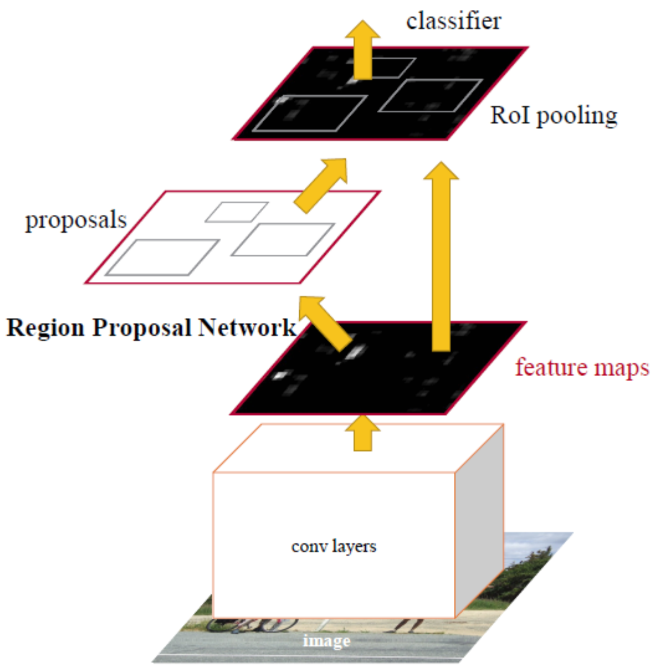


Figure 4.1: Faster-RCNN architecture [111].

object proposals. The proposals and scores are fed into a fast-RCNN network for model training. The RPN implemented as a fully convolutional neural network can be trained end-to-end by back-propagation and *stochastic gradient descent* (SGD).

The faster-RCNN algorithm reduces the time of computation, enabling a cost-free region proposal generation. It has good object detection performance with high *mean average precision* (mAP) at ImageNet large-scale visual recognition competition (ILSVRC 2015) and the Common objects in context(COCO 2015) competition.

Training data and pre-processing

Video data is recorded event-wise and activity-wise. Event-wise (ordered frames, like order collection, reach forklift trucks moving down aisles, pallets are being shifted to the racks, with forklift truck picking up the next order) are events. Above combination of events, formulating an activity, like order storage.

We formulate and annotate a training dataset for warehouse images for forklift trucks category. In our experiments, we use a subset from our dataset, including $3,077$ training images with manually labelled ground truth bounding boxes for forklift trucks and pedestrians defining 8,242 ROIs.

We resize images to $[600 \times 600]$ pixels, to reduce the computation time and then annotate with corresponding label categories. We use faster-RCNN detections both for single and multiple object tracking trials. Each image contains $1 - 7$ labelled instances of a forklift truck or pedestrian. Matlab $R2017a$ is used in model training. We used GeForce GTX 1080 Titanium GPU with compute capability 6.1 and 12GB memory.

Faster-RCNN training and evaluation

We use a *transfer learning* approach. Transfer learning involves retraining a few layers of a pre-trained deep learning model for novel category classification tasks. We select the AlexNet deep learning model. It is a 25 layered architecture pre-trained on 1000 image categories. We re-train the last three layers of the model with our own forklift truck recorded data.

Training has four main phases. In the first two steps, we train the region proposal and detection networks separately as used in faster-RCNN. In the last two steps, we combine the networks from the first two steps. A single network is created for detection [111]. Training steps have different convergence rates. So, we specify independent training options for each step. The *learning rate* for the first two steps is set higher compared to the last two steps i.e. $1e^{-5}$. Since the last two steps are fine-tuning steps, the network weights can be modified more slowly than in the first two steps ($1e^{-6}$).

During model training, image patches are extracted from training data. The two vital training parameters, the *positive overlap range* and the *negative overlap range*, control which image patches are used for training. We specify positive training samples as those samples that overlap with ground-truth boxes by 60% to 100%, as measured by the *bounding-box-intersection-over-union measure*. Negative training sample used overlap by 0% to 30%. The best values for these parameters are chosen by testing the trained detector on a *validation data set* (set separate from training and testing datasets).

To fully evaluate the detector, we test it on a testing warehouse dataset (we used

600 images at multiple scales). We use the *average precision* evaluation measure over all the detection results. This is the ratio of true-positive instances of forklift trucks to all positive instances for the object, based on the ground truth. It provides a single number that incorporates the ability of the model to make correct classifications (Precision) and the ability of the detector to find all relevant objects of interest (Recall).

We achieved true positive = 367, false positive = 33, true negative =83, false negative = 17, thus Precision = 91.75, Recall = 95.5, and f1 = 93.7 We obtained a running time for the detection as 5 frames per second on GPU.

4.3 Qualitative Results

Single forklift truck tracking

We obtain tracking trajectories that are close to manually estimated ground truth trajectories (based on calculating the accurate centroids of forklift trucks in each frame).

Scenario 1 shows a forklift truck moving straight down the aisle. There are not much curves or twists in the path; the forklift truck is assumed to be moving with constant velocity. The faster-RCNN results are shown in Fig. 4.2, left. For trajectory, refer to Fig. 4.2, right: The left image shows the overlaid faster-RCNN detections along with tracking points; right image shows the final (estimated) trajectory.

We improve the scenario 1 trajectory based on ROI saliency map thresholding; see Fig. 4.5, right. It shows the heat map for the overlaid saliency map on original images. Compare the estimated trajectories in Figs. 4.2, right, and 4.3. Fig. 4.3 depicts the trajectory correction based on *ROI saliency thresholding*.

Scenario 2 shows a forklift truck moving up the aisle from the left. Fig. 4.4, left, depicts the faster-RCNN detection output. There is a missing detection for the forklift truck in a couple of consecutive frames in this scenario. The KF robustly predict the path followed by the forklift truck when the missing detection arises, based on motion statistics. For trajectory, refer to Fig. 4.4, right. The left image shows the overlaid faster-RCNN detection along with tracking points. The right image shows the estimated trajectory.

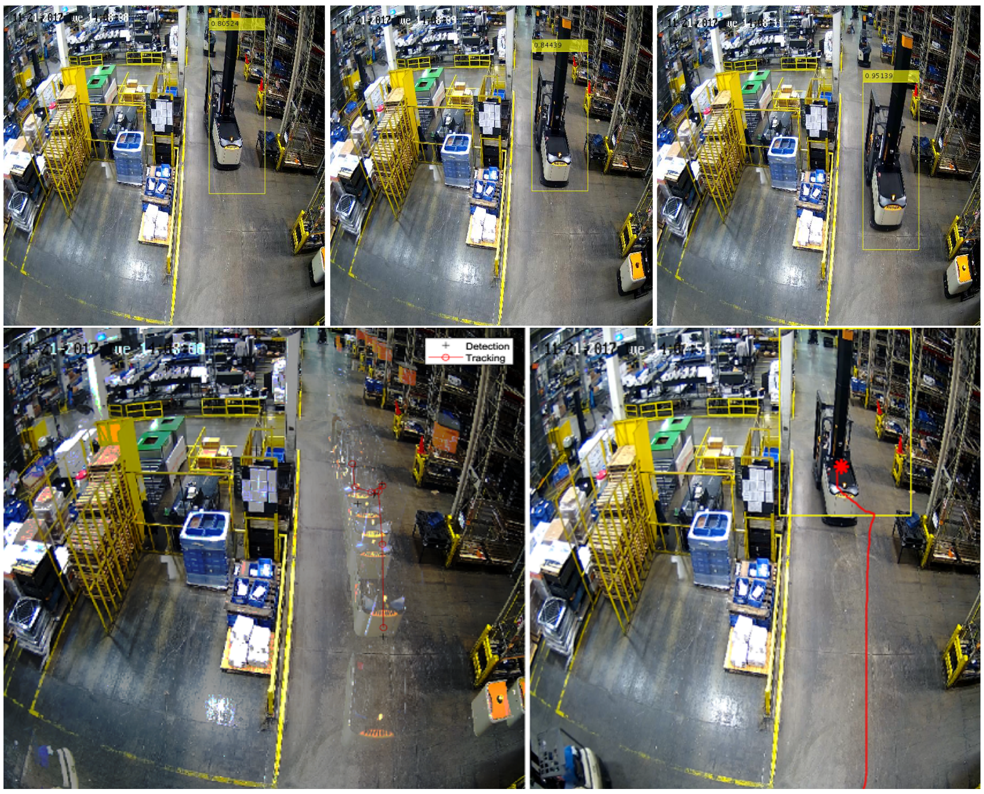


Figure 4.2: Scenario 1. *Left*. Faster RCNN detection results. *Right*. Estimated trajectory.

Scenario 3 has a forklift truck moving from the right, occluded for a few frames behind the racks in the aisle, and then moving up to the left. Our trained forklift truck faster-RCNN model gives an accurate detection for the occluded forklift truck. Refer to Fig. 4.5, left. Left image shows the overlaid faster-RCNN detection along with tracking points. The right image shows the estimated trajectory.

Multiple target tracking

For multiple forklift trucks and pedestrians tracking, a detection association method is used. In every up-coming frame, new RCNN detections are assigned to corresponding tracks from the previous frame using the Hungarian algorithm [81]. The assignment cost is minimized and calculated based on the Euclidean distance d be-

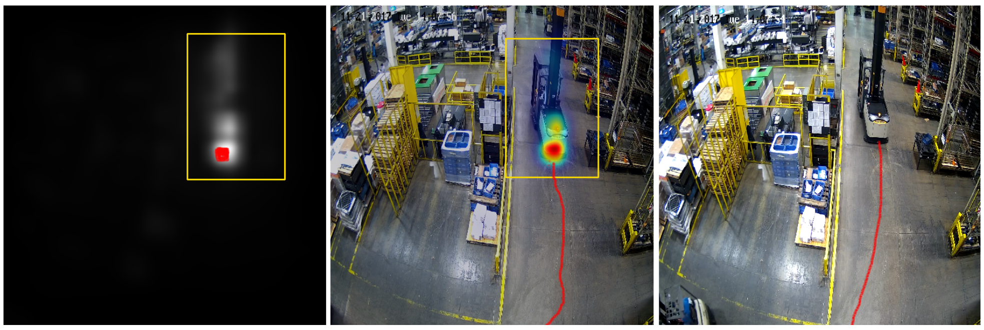


Figure 4.3: *Left*. Saliency map with detected centroid in red. *Middle*. Trajectory estimated by KF. *Right*. Ground truth trajectory.

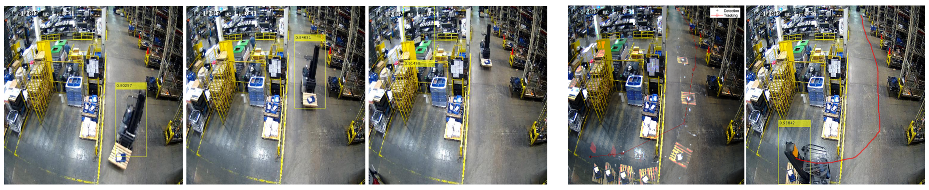


Figure 4.4: Scenario 2. *Left*. Faster RCNN detection results. *Right*. Estimated trajectory.

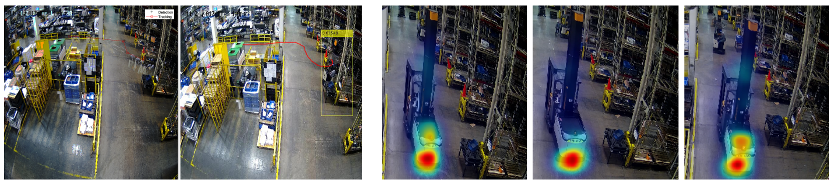


Figure 4.5: *Left*. Scenario 3 trajectory. *Right*. Heat map for pixel saliency overlaid on frames.

tween the centroids of each pair of bounding boxes [26]:

$$d = ||p_c(j) - p_c(i)||_2 \tag{4.1}$$

where $p_c(i)$ and $p_c(j)$ are centroids of boxes i and j, respectively. A maximum distance threshold is used to rate the assignment such that matches are discarded if $d > d_{max}$.

With new detections being assigned to existing tracks, the track is updated by

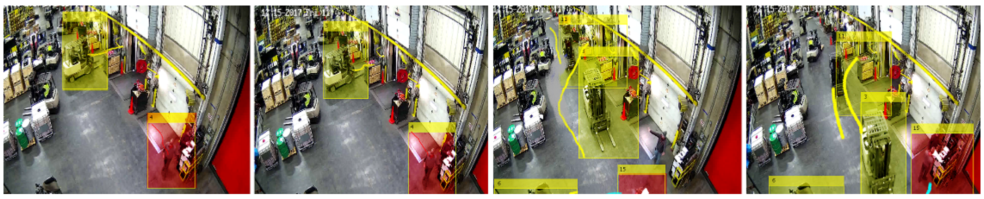


Figure 4.6: Multiple target tracking. *Left to right*. Frames 35, 54, 88, and 121. Coloured lines show track history of bounding box centroids for each tracked target.

estimating the state using the new observation. If a track is not assigned, i.e. a new detection in the current frame, the new bounding box is predicted by the KF. Matches between new detections in two consecutive frames create a new track. In case of false detections, showing up for a few frames, or no detections associated to a track for threshold track age, the track is deleted.

Refer to Fig. 7.5. Forklift truck tracks are shown in yellow and pedestrians with red bounding boxes. Forklift truck tracks 3 and 11, and pedestrian tracks 4, 15, and 6 remain consistent with no ID switching in the sequence we tested (of 840 frames). We have seen a few cases of track-ID switching. coloured lines show the track history of bounding box centroids, for each tracked target.

4.4 Quantitative Evaluation

Tracking performance can be mainly evaluated using two important evaluation measures, i.e. multiple object tracking accuracy (MOTA) and multiple object tracking precision (MOTP). MOTA takes into account all configuration errors made by the tracker i.e. false-positives, misses, number of mismatches, averaged over all frames:

$$\text{MOTA} = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{4.2}$$

where FN_t is the number of false-negatives or misses, FP_t is the number of false-positives, and IDSW is the number of ID mismatches/switches, where t is the frame index and GT is the number of ground truth objects.

The multiple object tracking precision (MOTP) is the average dissimilarity between all true-positives and their corresponding ground truth targets:

$$\mathrm{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{4.3}$$

where c_t denotes the total number of matches in frame t and $d_{t,i}$ is the bounding box overlap of target i with its assigned ground truth object.

Table 4.1: *Left*. Single-object tracking. *Right*. Comparison of tracking results on TUD crossing sequence of [100] also showing our method. Last column shows our results for forklift truck-pedestrian sequence.

Quantitative evaluation		
Metrics	Saliency-based Kalman-track	Non-saliency-based Kalman-track
MOTP	78.76↑	52.1
Miss rate	12.08 ↓	20.86
FP rate	19.17	19.86
Local-error	12.52 ↓	38.76
MOTA	67.36↑	56.88

Sequence:TUD Crossing				
Measure	[29]	[161]	Ours	Results
MOTP	73.1	71.9	72.3	81.3 ↑
MOTA	61.3	33.7	51.2	85.1↑
FAF	0.1	1.3	1.1	1.6
MT	38.5	12.2	61.2↑	65.3↑
ML	15.4	44	41	43.1
FP	14	7762	4033	513
FN	401	325	156	1056
ID SW	12	442	331	254
Fragment	27	823	693	317

Table 4.1 left, outlines the evaluation for single forklift truck tracking using sequence scenarios as discussed before. It outlines the improvement we achieve by incorporating ROI saliency maps in our tracking framework. Corrections for bounding box centroids based on ROI saliency map results in lesser localisation errors and targets missed due to threshold distance. Eventually, MOTP and MOTA have enhanced accordingly. We use a singe object track as a special case of MOT with the CLEAR MOT measure [27].

We benchmark our pedestrian/forklift truck tracking framework and tested it on the TUD crossing public dataset [101]. TUD crossing is an outdoor sequence with 201 frames [100]. Table 4.1, right, outlines the methods we compare and the MOT evaluation results [124].

Since for the forklift truck-pedestrian category, no public benchmarks are available for evaluation, a ground truth for our dataset sequence is formulated and we evaluate tracking results for multiple forklift truck-pedestrian objects accordingly. We obtain very fair results for multiple-object tracking; see Table 4.1, right.

4.5 Summary

We successfully computed target trajectories under challenging and constrained warehouse environment. Kalman filter is able to successfully track corresponding targets, especially for the objects, for whom we are able to get accurate detection results from trained faster-RCNN detector. Few challenging scenes involving partially occluded pedestrians and forklift trucks, offered more false positives and false negatives, that effected the tracker robustness and trajectory quality.

We also observe that faster-RCNN is better at trajectory computation for forklift trucks, rather than pedestrians. Detection quality for pedestrian is not very optimal under clutter and occlusions. We study that the reason is successive downsampling degrades the feature maps associated with small region hypotheses for the pedestrian class and effect its classification accuracy. Calculated pedestrian and forklift truck trajectories lead to further analysis tasks for pedestrian safety in warehousing.

Chapter 5

Improved Data Association Measure for MOT

The association of predicted object locations to its respective correct track over time is vital for robust tracking results. When employing a detection-based tracking approach, it is important for handling object track identity switches which is one of the limiting issues when designing multiple-object trackers. In this chapter, an object model is described by its saliency-colour histogram. The dissimilarity between a reference track model and input candidate detection (obtained by an object detector) is quantified by a measure derived from the Bhattacharyya coefficient. The Bhattacharyya distance measure is used for validating the confidence in tracks based on an adaptively obtained threshold value. This approach results in improved tracker results under a track-identity loss scenario, especially under short-term partial or full occlusions, clutter or scale variations. The suggested method for track assignment, combined with a state-of-art real-time object detector You only look once (YOLO) led to improved tracking results by reducing the number of miss-assignments within tracks.

5.1 Introduction

Multiple methods have been used in literature for the computation of cost affinity, for the task of assigning candidate detections outputs to corresponding tracks in online tracking applications. Traditionally, a cost assignment matrix is generated by using various distance measures based on cues such as motion, appearance, colour, *saliency*, texture, or a combination of them. The popular Hungarian method has been used quite often for solving such an assignment in a *multiple-object tracking* (MOT) paradigm [81].

In detection-based tracking frameworks, rather than employing features beyond the detection component, object bounding box geometry (i.e. its position and size) is often being used for motion estimation and data association. The *simple online and real-time tracker* (SORT) is an example of such an approach. This is a Kalman-filter-based tracking algorithm [25] with frame-to-frame data association being done based on the *intersection over union* (IOU) measure. This measure expresses the bounding-box overlap-ratio, and gives most effective tracking outputs at higher frame rates. SORT used the Hungarian algorithm for solving the cost assignment problem. Its recent version is *SORT with a deep association metric* (DeepSort). See [139].

In DeepSort, for data association both motion and appearance cues were combined to form a cost affinity matrix. It has an improved *multiple-object tracking accuracy* (MOTA) compared to its predecessor. It mainly improved the problem of identity switches in tracks. This is being achieved by adding a deep CNN that has been trained offline to classify pedestrians on a large scale person re-identification dataset. The *cosine distance* between detected pedestrians and their tracks in appearance space is being used to re-identify them. This deep distance metric, in combination with the Mahalanobis distance, is used to compute detection-to-track affinity for assigning multiple detections to their corresponding tracks [139].

Also, the complex *joint probability data association based filter* (JPDAF) tracker has been revisited, in a tracking-by-detection paradigm. It is based on weighing object measurements by their association likelihoods for generating state hypotheses [57]. In a multiple hypothesis tracking approach (MHT) [78], every possible state hypothesis is tracked. For both these approaches, their computational complexity grows exponentially with an increase in the number of objects to be tracked which makes them impractical for online real-time tracking applications.

It is suggested that *appearance-saliency-map guided data association measure* could be an advantage to verify the track identity especially in cases of occluding tracks or multiple bounding boxes on the same target. A visual saliency map of the *ROI* could be an important feature in validating the track assignment problem. A saliency distribution dissimilarity measure between a detected ROI and predicted candidate track locations is described by the Bhattacharyya coefficient. It could be a useful association metric if used in conjunction with the Mahalanobis distance. Though it is used earlier in the mean shift tracking framework, here is it re-purposed for validating confidant tracks. It is an optimal assumption due to its link to Bayesian errors; although there are other distance measures like the Kullback distance measure (but this violates one of the distance axioms) or the histogram intersection measure (which is scale variant) [35].

In order to improve the robustness of the tracker, we combine both colour and saliency features here, weighting their respective contribution automatically to the cost likelihood function, making this a more informed metric for data associations. Colour histogram feature incorporation makes the tracker robust to the scenes where objects seem to be less salient than background and saliency cues seem to be less in-

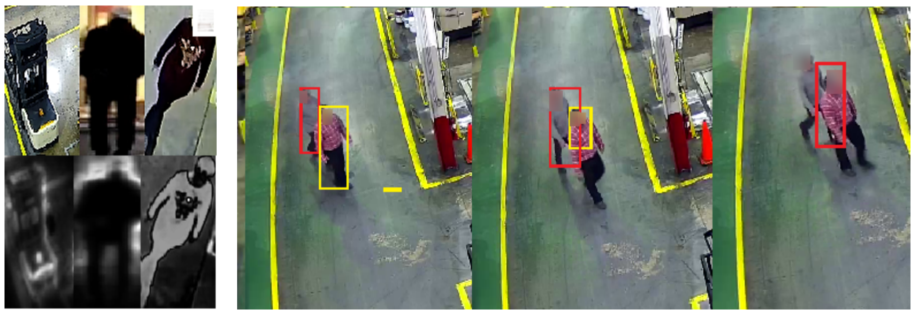


Figure 5.1: *Left*. Saliency detection example. *Right*. Identity switch scenario for short term occlusion

formed measures for the association. The resulting strategy will improve the *mostly tracked trajectories* (MTT) metric and decreases *identity switches* between tracks.

5.2 Concept

See Fig. 5.1. The left figure outlines a saliency detection example and the right depicts a scenario for a short term occlusion. It can be seen that two tracks are partially occluded by each other for a short time. We argue that the IOU distance measure [24], between detections and predicted bounding boxes from the existing track targets, is a non-optimum measure here. Track identity is lost only because the IOU distance measure validates the red track but is actually the yellow track interfering with the target, causing the track miss-assignment here (in the third frame, detection belonging to the yellow track being assigned to the red track).

It is proposed that the corresponding detection will not be assigned to the red track if the *Bhattacharyya distance* validation fails (due to dissimilarity between colour-saliency distribution between predicted red track location and corresponding detection) outlining that it is just *occluding another track* and not the original tracked object. We suggest that the Mahalanobis distance will solve the Kalman motion uncertainty problem; a saliency distribution similarity measure of candidate detection and tracking, quantified by the *Bhattacharyya distance measure* bdist, will solve the short-term occlusion problems. It will make the tracker more robust to occlusions. This work is inspired by various recent multiple-target tracking algorithms which have employed visual saliency maps for robust tracking outputs [6, 13, 26, 105, 118, 149, 157, 159].

5.3 Tracking Framework

An *object* is being modelled based on its appearance and motion features. The displacement of a specific object in the next frame is modelled by a linear velocity assumption using a Kalman filtering framework. The camera is assumed to be static and uncalibrated. The tracking problem is defined in 8-dimensional state space $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$ which combines bounding box centre coordinates (u, v), their aspect ratio γ, height h and velocities in image space. An object state at any time instance t is defined directly by its bounding box coordinates (u, v, γ, h).

5.3.1 YOLOv3 Object Detector

Robust target localisation is attained using deep architectures such as *Faster-RCNN* [111], SSD [98] or YOLO [112]. Few make use of sliding-window approaches or region-proposal methods, to generate region hypotheses, and then to generate objectness scores. These methods proved to be more robust for detection for varying scales of objects [112]. The YOLO architecture for detection has been released in two versions: v2 and v3.

The reason for employing a deep learning framework like YOLO in this work is getting *reliable*, i.e. accurate and real-time detection quality for tracking.

YOLO is an end-to-end single CNN architecture which detects the object's ROIs based on their bounding box predictions and class probabilities. It divides the input image into a $S \times S$ grid. Moreover, if the centre of an object is located within that grid, then the grid will detect the object of interest. Each grid predicts the bounding boxes and a confidence score, where the confidence score quantifies how confident the model is that the bounding box certainly contains a specific class of objects.

The IOU measure between a predicted box and ground truth is used to calculate confidence for a detection. It outputs 1 for a perfect match case, and the opposite if a predicted box is not present in the grid. YOLO is a sliding window-based method; but, unlike a model-based Aggregate Channel Features(ACF) detector, it examines the entire image during training. It learns the context of the object with respect to its surroundings as well.

Algorithm 1: Saliency-guided data association for computing track assignment

```
Input  : A new frame at time t with detections set from YOLO detector $D^t$
         and track set $T^{t-1}$
Output: The new track set $T^t$
1  For each detection candidate from set $D^t$;
2  if Mahalanobis-distance $< \tau_m$ then
3      if Bhattacharyya-distance $< \tau_{bdist}$ then
4          Assign the corresponding detection to a track;
5          Obtain assigned track set $T^{t-1}_{assigned}$ matched detection set $D^t_{matched}$ ,
            Unassigned track set $T^{t-1}_{unassigned}$ and unmatched detection set
            $D^t_{unmatched}$;
6          Predict first track subset by Kalman Filtering based on successful
            association in previous step: $Trackset^t_1$;
7          Predict second track subset from unassigned track set or remove the
            ones who crossed the max track age threshold $\tau_{max-age}$ i.e. output
            is $Trackset^t_2$;
8          Create third track subset from unmatched detection set $D^t_{unmatched}$
            i.e. $Trackset^t_3$;
9          Combine the three track subsets to form new candidate track set
            $T^t_{candidate}$;
10         $T^t_{candidate}$ formed new input for the algorithm at time t+1 with
            detections set $D^{t+1}$ For each detection candidate from set $D^{t+1}$, find
            Mahalanobis-distance b/w each detection and corresponding track
            mean and Bhattacharyya-distance quantifying degree of similarity
            in their saliency distributions, go to Step 3 if condition is met.
11     end
12 end
```

YOLO version 3 (YOLOv3) is an improved version of its earlier version YOLOv2, now using multi-scale training, a better classifier and image patches of 320×320; it runs in 22 ms at 28.2 mean average precision (mAP). The trade-off between speed and accuracy of the network is checked during training by changing the size of the network.

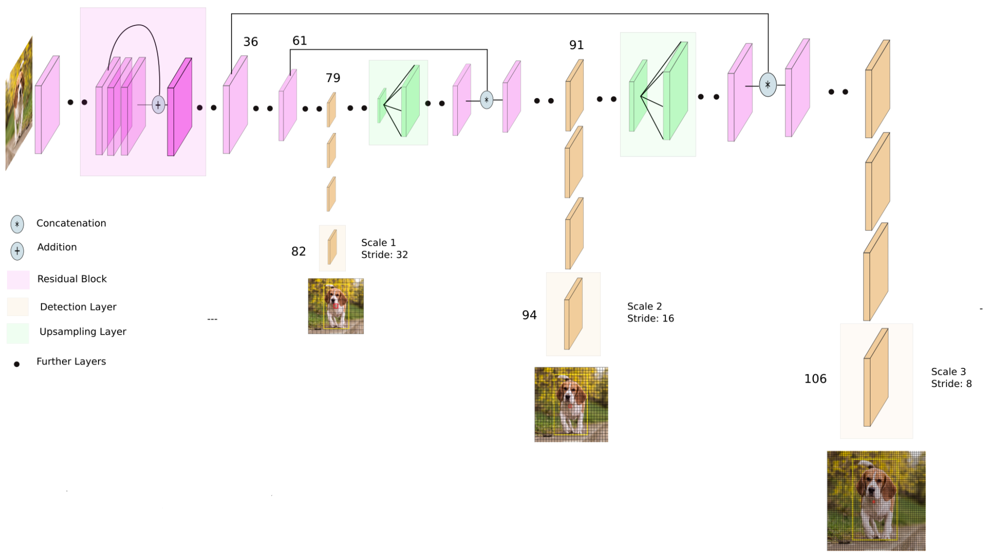


Figure 5.2: Architecture for Yolo-v3 [150]

5.3.2 Training Data

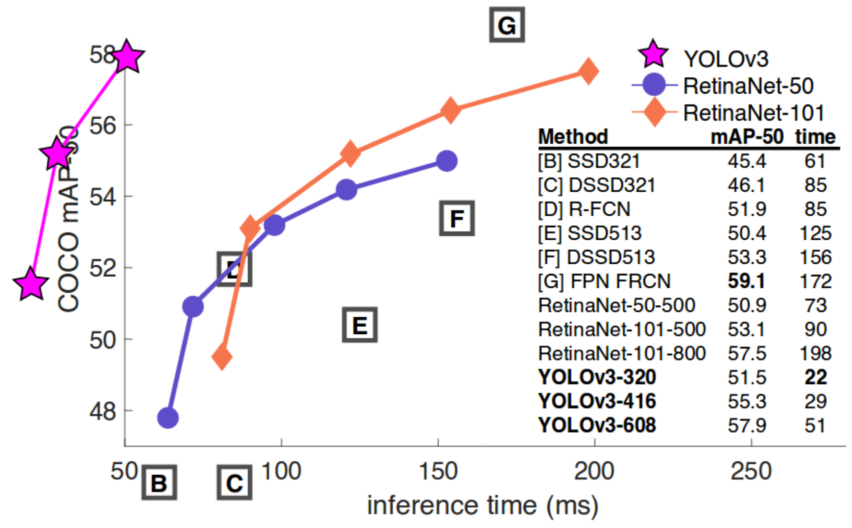
The *YOLOv3* object detector is trained for categories: *pedestrians* and *forklift trucks*. The detection outputs are employed to track pedestrians and trucks in occluded and cluttered warehouse environments (tracking by detection approach).

We have $3,077$ training images with manually labelled ground truth bounding boxes for forklift trucks and pedestrians, defining $8,242$ ROIs. Each frame contains $1-7$ labelled instances of a forklift truck or pedestrian. Subsets of image data from the Caltech pedestrian [30] and INRIA pedestrian datasets [67], are also added to get enough training samples for pedestrians.

The pedestrians, present in these subsets, with different degrees of occlusion, wearing different costumes, have many kinds of scales and changing postures. GeForce GTX 1080 Titanium GPU with compute capability 6.1 and 12GB memory is used for model training. Table 5.1 summarises the number of training images and ROIs used for every image subset.

Table 5.1: Datasets

Dataset	Caltech pedestrian subset [30]	INRIA person subset [67]	Our dataset
Training images	10,000	614	3,077
ROIs	22,000	1,237	8,242



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 5.3: Graphical comparison to other architectures [151]

5.3.3 YOLO Training and Evaluation

For model training, CNN weights that are pre-trained on the ImageNet dataset are used, i.e. weights from the *DarkNet53 model*. DarkNet model inputs a text file for each image and every ground truth object in the image in the format $x, y, width, height$ where these parameters are relative to the image's width and height. A text file with names and paths of the images to be trained was supplied to DarkNet to load the images to be trained on. A test text file for testing and validation text file for validation images is also created.

After training the pre-trained *DarkNet53 model* for about $20k$ iterations ($13,500$ images), a validation accuracy of 91.2 and a testing accuracy of 90 $\%$ is obtained. This was without any data augmentations or other transformations. Random flag parameters for multi-scale training were enabled in the script, resulting in robustness for detecting objects in different image resolutions. The input image size by default was set to a resolution of (416×416), but was altered every 10 batch. Resulting test accuracy was good enough for this work.

5.4 Data Association

The Mahalanobis distance, which is the squared distance between predicted object states (by a Kalman filter) and incoming new measurements, can be written as follows:

$$\mathbf{d}^1\left(i,j\right) = \left(d_j - y_i\right)^T \mathbf{S}_i^{-1}\left(d_j - y_i\right) \tag{5.1}$$

where (y_i, S_i) denotes the projection of the i-th track distribution into measurement space and d_j denotes the j-th bounding box detection. It provides information about possible object locations based on motion that helps in short term prediction of tracking a state. It eliminated unlikely assignments by thresholding at 95 percent confidence interval, computed from the inverse χ^2 distribution $\tilde{\chi}^2$ [139].

We denote by $\mathbf{p}^*$ the reference track saliency-colour model; $\mathbf{p}\left(x_t\right)$ is the candidate detection model. Then, the distance between the two is described by the Bhattacharyya distance (bdist) [118]:

$$\mathbf{d}^2\left(i,j\right) = \mathrm{bdist}\big[\mathbf{p}^*, \mathbf{p}\left(x_t\right)\big] = \left[1 - \sum_{u=1}^{M} \sqrt{\mathbf{p}_u^*\left(x_o\right)\mathbf{p}_u\left(x_t\right)}\right]^{\frac{1}{2}} \tag{5.2}$$

For this work, the degree of saliency of a region with respect to its neighbourhood is described by the Euclidean distance between the pixel vector and the average vector of the input detection in the *Lab* colour space [8]. It is described by:

$$S\left(x,y\right) = ||I_\mu^* - I_\sigma^*\left(x,y\right)|| \tag{5.3}$$

where I_μ^* is the mean image feature and I_σ^* is the Gaussian blurred image, using a5×5 separable binomial kernel.

Also, the colour or saliency distribution of a region of interest ,described by detection, centred at location x, is given by:

$$\mathbf{p}_u\left(x\right) = C\sum_{i=1}^{N_p} k\left(||\frac{x_i - x}{h}||^2\right)\delta\big[b\left(x_i\right) - u\big] \tag{5.4}$$

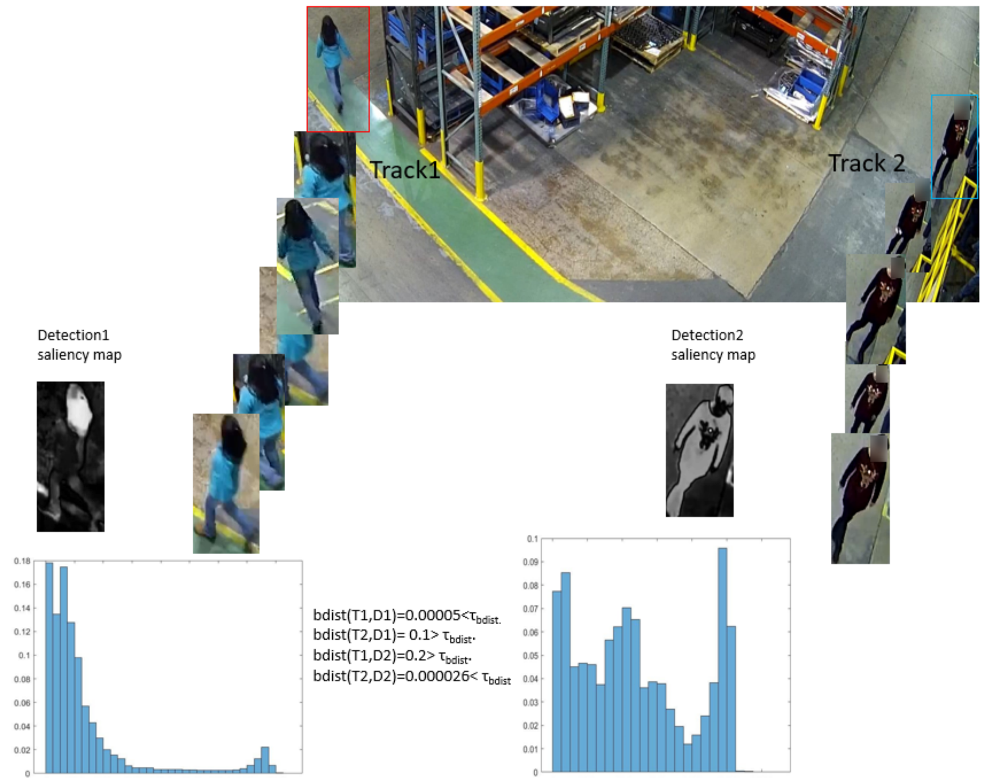

Track1
Track 2
Detection1
saliency map
Detection2
saliency map
bdist(T1,D1)=0.00005<τ_{bdist}.
bdist(T2,D1)= 0.1> τ_{bdist}.
bdist(T1,D2)=0.2> τ_{bdist}.
bdist(T2,D2)=0.000026< τ_{bdist}

Figure 5.4: Saliency guided data association

where C is a normaliser, δ is the Kronecker function, k is a kernel with bandwidth h, N_p is the number of pixels in the ROI and $b\left(x_i\right)$ is a function that assigns one of the m bins to a given colour or saliency at location x_i. The kernel k is used to consider spatial information for adaptively lowering the values for far pixels [118]

The influence of each distance measure on the combined association cost can be controlled through a hyperparameter Λ. Colour and saliency will be weighted automatically according to its value for deciding the final affinity for corresponding detections. The cost optimisation is being solved optimally through the Hungarian algorithm:

$$\mathbf{c}_{i,j} = \Lambda \mathbf{d}^1\left(i,j\right) + \left(1-\Lambda\right)\mathbf{d}^2\left(i,j\right) \tag{5.5}$$

Refer to Fig. 5.4 for visualising the concept. It can be seen that for detection 1, bdist between detection 1 and track 1 fulfils the bdist criterion and also lies within the Mahalanobis distance threshold, so it is being assigned to track 1. The distance of detection 1 to track 2 violates the bdist threshold. The same principle applies for the detection 2 assignment here:

1. For the task of assigning detections to current track targets, every targets bounding box geometry is predicted by the Kalman filter.

2. The tracker is initialised by using the geometry of the object bounding box, with the initial velocity set to value zero.

3. Covariance of the velocity component is initialised with a large value to model uncertainty in assumption.

4. When a detection is associated to the target, the target state is updated by this detection and Kalman filter is used to solve the velocity components optimally. But if the detection is not associated to the target identity, then its state is predicted rather than corrected by the filter using the respective motion model.

5. A threshold is imposed to reject assignments where the detections lie beyond that region of assignment both for *Mahalanobis* and *Bhattacharyya* distance, i.e. should lie within the thresholds τ_m and τ_{bdist} (These thresholds are selected adaptively based on the scenario).

6. For creating trackers, we consider any detection that violates both distance criterion, to signify the existence of an untracked object.

7. Tracker has to undergo a trial period where the target needs to be associated with detections to have enough evidence in order to prevent tracking of false positives. Tracks are terminated if they are not detected for $\tau_{max-age}$ frames. This ensures to prevent an increase in the number of trackers caused by predictions over longer durations without corrections from the detector.

8. In all experiments $\tau_{max-age}$ is set to 30 frames contrary to max-age of one frame (as in SORT) [24] that resulted in many identity switches among tracks).

5.5 Quantitative Evaluation

To evaluate tracking performance, two evaluation measures are employed in this work, i.e. *multiple object tracking accuracy* (MOTA) and *multiple object tracking precision* (MOTP). We conducted experiments with the proposed improvement in data association metric. We use detections for pedestrians and trucks with a confidence

Table 5.2: Tracking metric results improvement depicted by increase in value of *MOTP, MOTA* and *MT*(mostly tracked trajectories).

Quantitative evaluation		
Metrics	With IOU measure	with saliency guided bhattacharyya distance measure
MOTP	71.1	73.9 $\uparrow$
MOTA	61.3	63.7$\uparrow$
FAF	0.1	1.3
MT	13.2	20.1 $\uparrow$
ML	15.4	43.1
FP	267	345
FN	401	325
ID SW	25	19
Fragment.	27	82

threshold of .25 or higher from YOLO for our work.

Refer to Fig. 5.5. Tracking results from our warehouse data is shown for pedestrians and trucks, partially occluded by each other. It can be seen that there is no identity switching due to the use of distinct colour-saliency map validation step.

We also tested it on the TUD-Stadtmitte sequence from the MOT challenge 2015

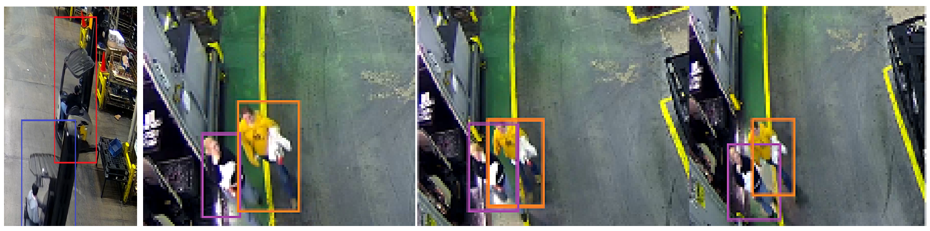


Figure 5.5: *Left*. Occluded forklift tracking *Right*. Multiple occluded pedestrian tracking

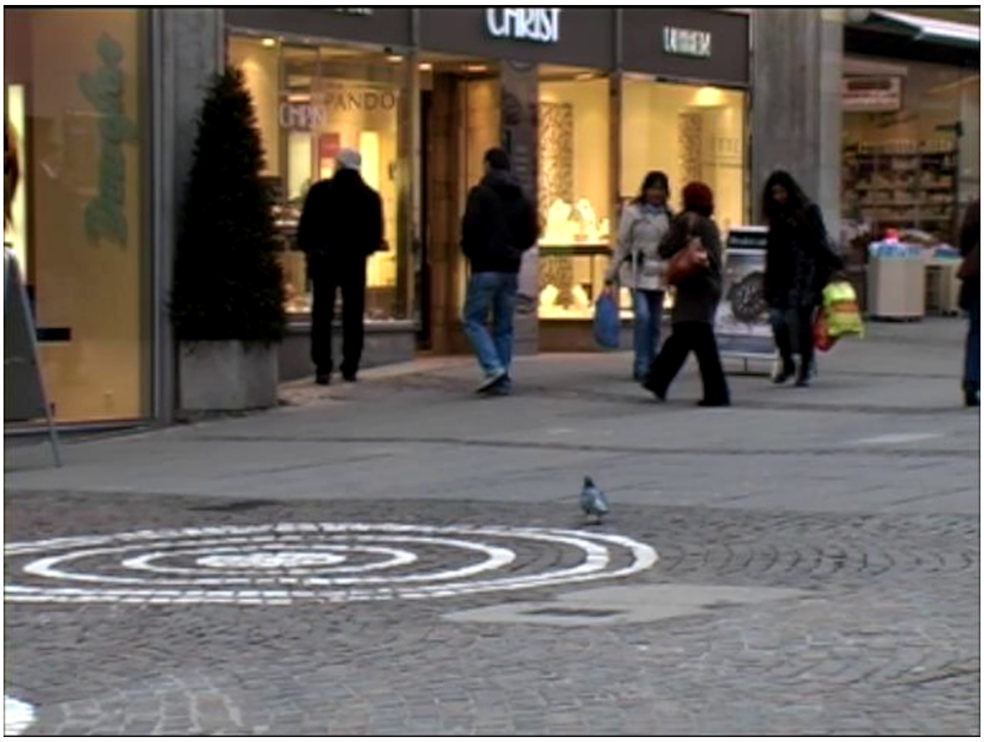


Figure 5.6: Example of a challenging scene from the TUD-Stadtmitte sequence showing pedestrians with similar looking appearance and different poses [100]

dataset [100]. It is recorded with a static camera placed at 2 meters height showing people walking on the street. It is an outdoor sequence with 179 frames, 10 tracks, 1, 156 bounding boxes, 640×480 resolution and 25 fps. Pedestrians are shown partially occluded with low illumination in background and object contrast is minimal. Figure 5.6 is a shot from the sequence.

Check Table 5.2 for quantitative results improvement by incorporation of the proposed saliency guided data association method.

Table 5.3 outlines the comparison with other state of art multiple objects trackers [24, 139]. Our implementation runs at around 20 frames/second. Hence, given a modern GPU, the system will operate in real-time. Since for the forklift tracking category, public ground-truth detections and tracks are not available, we formulated ground-truth dataset for evaluation.

Table 5.3: Quantitative comparison with [24, 139]. Last column lists results for a forklift-pedestrian sequence from warehouse data

Sequence:TUD Crossing				
Metrics	[24]	[139]	Ours	Results
MOTP	63.1	71.9	72.3	73.3 ↑
MOTA	35.3	53.7	56.2	75.1↑
FAF	2.3	1.3	1.1	1.6
MT	12.5	38.2	41.2↑	55.3↑
ML	25.4	14	21	23.1
FP	762	1062	833	513
FN	401	325	256	1056
ID SW	442	245	331	254
Fragment	186	102	142	317

5.6 Summary

We aimed at a deep exploration into an algorithm for robust multi-object tracking, for further limiting problems known from testing previous trackers. Such robust track association criterion will definitely help to deal with occluding moving targets tracking and dealing with track identity switch problems, especially in the MOT paradigm.

Chapter 6

Saliency-enhanced Correlation-filter-based Visual Tracking

A discrete correlation-filter-based multi-cue-analysis framework is constructed by fusing different feature types to form potential candidate trackers that track the target independently. The selection of corresponding cues and the exploitation of their individual or combined strengths is a less researched topic especially in the context of ensemble tracking. Every candidate tracker from the ensemble is chosen according to the degree of its robustness per frame. We argue that, if each of the candidate trackers is guided by higher-level semantic information (i.e. pixel-wise saliency maps in the ensemble-based tracker), this will make tracking better to cope with appearance or viewpoint changes. Recently, saliency prediction using deep architectures have made this process accurate and fast. The formation of multiple candidate trackers by saliency-guided features along with other different handcrafted and hierarchical feature types enhances the robustness score for that specific tracker. It has improved multiple tracker-based DCF frameworks in efficiency and accuracy as reported in our experimental evaluation, compared to state-of-the-art ensemble trackers.

6.1 Introduction

For multi-cue correlation-filter based visual tracking (MCCT), an adaptive tracking switch mechanism is employed where the tracker adaptively transfers the tracking task to one of the ensemble candidate trackers [138]. The candidate, which results in precise target localisation and which is consistent in its results, is chosen to track in a given frame. Such tracking candidates might be constructed by fusing handcrafted low-level features, e.g. HOG [38], colour names [136], or middle and higher level deep activation features using outputs from conv3-4, conv4-4, and conv5-4 layers from a VGG-19 pre-trained model [116]. See Table 6.1 for an example of an ensemble. State-of-the-art results on visual recognition tasks are obtained by exploiting rich hierarchies in features in deep architectures and learning multi-layer correlation response maps. Response maps are analysed in a coarse to refined manner to include the features that are most appropriate for tracking tasks.

Table 6.1: Candidate-trackers selection.

Trackers	Feature type
Tracker I	Sal-feature and low (HOG)
Tracker II	Sal-feature and middle (conv3-4) of VGG-19
Tracker III	Sal-feature and high (conv5-4) of VGG-19
Tracker IV	Sal-feature and low, middle
Tracker V	Sal-feature and low, high
Tracker VI	Sal-feature and middle, high
Tracker VII	Sal-feature and low, middle, high

For deep CNNs, features from later layers convolution layers are more closely related to object category level semantic information. These feature maps though are robust to intra-class appearance variations but they fail the objective to locate targets precisely. The output of earlier convolution layer outputs is more appropriate for target object fine-spatial appearance information, but not for the semantics [102].

We propose that *saliency is a discriminant feature* that aids in differentiating the target from the background and can be a useful and efficient way to attain robustness against target appearance changes at a semantically higher level. We argue that incorporating a *saliency feature* as a complementary higher-level feature, and learning correlation filters for them separately, will aid the tracking quality for each tracker in an ensemble. A candidate tracker employing this semantic aware higher-level feature will benefit in track accuracy and precision and thus the final tracking robustness against appearance and viewpoint changes.

This can also be verified in a performance gain attained for the candidate trackers. The corresponding weight for the saliency response will be chosen adaptively according to its consistency over time for the saliency feature in consecutive frames. We treat saliency and other features separately [14] for each tracker. This benefits the framework effectively, especially under occlusions and frequent scale variations of the target.

6.2 Saliency Feature

Saliency-based trackers show the usefulness for using saliency features over various other cues such as motion, texture, gradient or colour. An object is represented and tracked based on a human attentive mechanism using visual saliency maps. These maps aid to segment the regions based on spatio-temporal characteristics in videos and images.

A combination of various features like colour-texture, colour-saliency, or colour-direction is explored. To track non-rigid objects, spatio-temporal discriminative

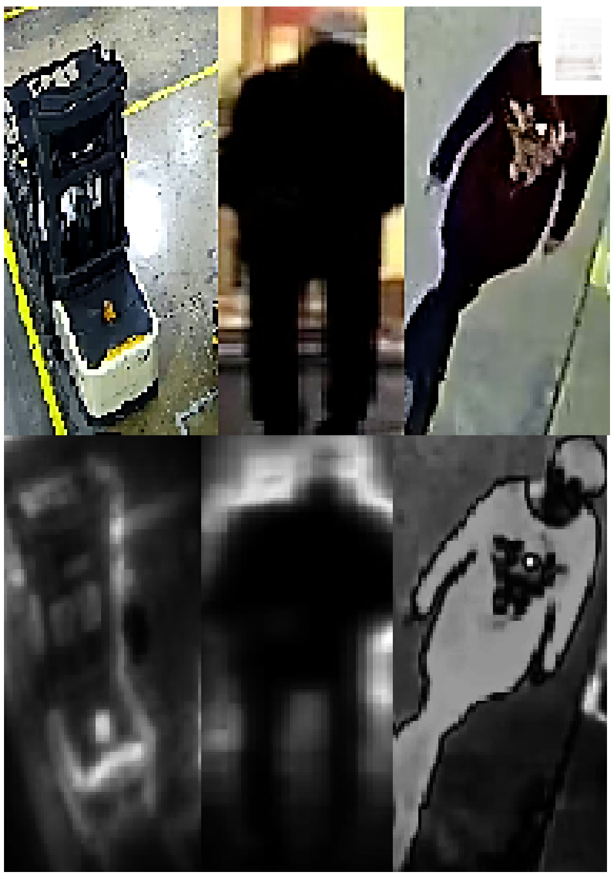


Figure 6.1: Saliency detection example.

saliency maps were used. This gave accurate regions occupied by the targets as tracking results. A *tailored fully convolutional neural network* (TFCN) was developed. It was used to model the local saliency of regions of interest. Finally, these local saliency maps were fused with a weighted entropy method, resulting in a final discriminative saliency map [157].

In this work, *deep hierarchical saliency network prediction* (DHSnet) is employed to output the saliency map of an image using a deep CNN. It uses the whole image as a computational unit and feedforwards the image for testing without any post-processing. First, a coarse saliency map is generated for the global view for a rough estimation of salient objects. By incorporating local context in the image, a refined saliency map is obtained through a recurrent CNN. This refinement is done in many steps hierarchically and successively [96]. Refer to Fig. 6.1 for saliency detection examples.

6.3 Our Approach

A brief outline of our approach is as follows:

1. First, a discrete correlation filter is learned from the appearance of the object at a given frame. Features are extracted from chosen ROI where f_{sal} and $f_{\text{HOG}-\text{CNN}}$ denotes the feature maps for saliency, and HOG and CNN.
2. Second, the previously learned correlation filters for saliency and other features denoted by g_{sal} and g_{feat}, respectively, are convolved with the extracted feature maps for them in order to obtain responses (scores) R_{sal} and R_{feat}.
3. Finally, the filter is updated according to the new appearance which is obtained from the location found in the second step according to the proposed adaptive update strategy.

These steps will become more clear with the explanations below.

The robustness of different trackers varies with regard to various feature models employed for them. One dynamic feature model will not suffice to deal with varying challenges faced by the visual trackers [102]. Low, middle and high-level features are combined into 7 candidate trackers (see Table 6.1), working independently in an ensemble where the low-level HOG feature is 31-dimensional. Settings are re-used as employed in hierarchical convolutional features (HCF) for different level response maps [102]. Tracking diversity is achieved by adaptively choosing the trackers according to its robustness for the current frame.

Trackers using just a single feature may not be sufficiently robust to an optimum extent, but the addition of a complimentary sal-feature improved the results. We tested the sal-feature-guided strategy for our 7 different trackers in the ensemble and found that the response for each candidate was improved. Tracker VII, based on three kinds of feature hierarchy performances, is also improved and more robust to appearance changes. Section 6.4 depicts detailed quantitative experimental results.

Saliency response and other feature response maps from each tracker are obtained separately and denoted as R_{sal} and R_{feat}, respectively. Refer to Fig. 2 for an overview. Every tracker is proposed to employ saliency as a high-level complementary feature, though the reliability of the saliency needs to be estimated for each incoming frame.

First, we estimate the saliency-based on DHSNet [96]. This saliency prediction network is based on a hierarchical end-to-end deep convolution neural network. This network learns global saliency cues like contrast, objectness, compactness with the added recurrent neural network to refine the saliency map details by adding local contextual details. This network is trained by employing a bulk of images and their corresponding saliency masks.

To estimate the similarity between the vectorized saliency maps Sal_t and Sal_{t+1} at frames t and $t+1$, *cosine similarity* is used which is described by:

$$\mathcal{C}_{sim}\left(Sal_t, Sal_{t+1}\right) = \frac{Sal_t \cdot Sal_{t+1}}{||Sal_t||_2 \cdot ||Sal_{t+1}||_2} \tag{6.1}$$

A maximum contribution of the saliency is selected adaptively based on the previous and current saliency response. The weight of the saliency for frame t is given by:

$$w\left(t\right) = M\Bigg[\left(1-\lambda\right) w\left(t-1\right) + \lambda \cdot \mathcal{C}_{sim}\left(Sal_t, Sal_{t+1}\right)\Bigg] \tag{6.2}$$

We choose M to be 0.5; that is the maximum suggested contribution assigned to the saliency feature. This value is adaptively chosen based on scene statistics. The learning rate λ is chosen to be 0.01 owing to the intention that the update of saliency is assumed to be gradual due to noise. The initial weight of saliency is chosen to be $w(t=0) = 0.25$, i.e. half of the possible maximum contribution. We base our choice

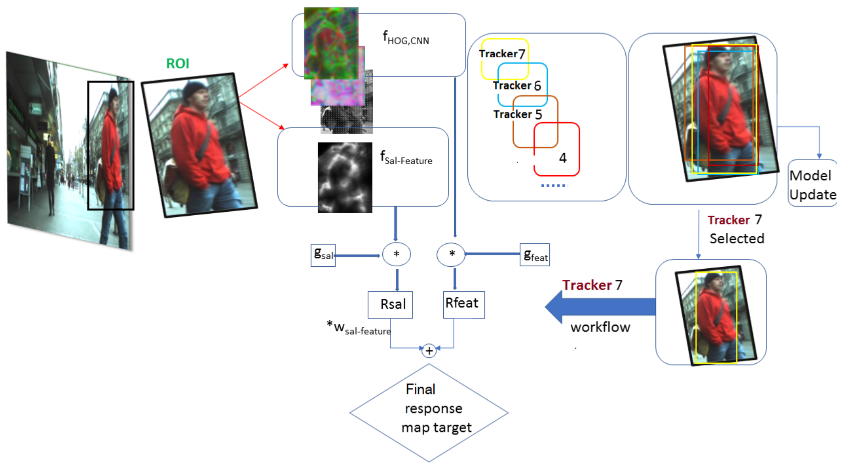


Figure 6.2: Suggested update concept for feature and decision fusion.

on settings employed by [14].

6.3.1 Tracker Robustness Measurement

Each tracker is employing the same region of interest and sharing similar training samples during filter learning and the tracking update process. By exploiting the refined feedback results from each tracker for target localisation, weak tracking outputs, especially during tracking drift scenarios and failures, are improved. This is the main reason for framework computational efficiency as well. The feature extraction is done only twice, rather than 14 times.

Peak to sidelobe ratio (PSR) is commonly used in a standard DCF framework to estimate the reliability of tracking outputs. Often this measure fails to quantify the reliability properly when un-reliable samples have the same PSR as the target of interest. Robustness score for the candidate trackers is computed from two measures. *tracker pairs evaluation* stands for the degree of consistency between pairs of trackers for each frame, and *tracker self-evaluation* stands for the trajectory smoothness of a candidate tracker hypothesis at every frame.

Let $\{T_1, T_2, T_3, ..., T_7\}$ denote the seven hypothesis trackers. In the t-th frame, Tracker i's bounding box is denoted by $B^t_{T_i}$, and Tracker j's bounding box accordingly by $B^t_{T_j}$.

The overlap ratio $O^t_{T_i,T_j}$ is defined by:

$$O^t_{T_i,T_j} = \frac{B^t_{T_i} \cap B^t_{T_j}}{B^t_{T_i} \cup B^t_{T_j}} \tag{6.3}$$

Tracker pair robustness score is given by:

$$Score^t_{pair}\left(T_i\right) = \frac{M^t_{T_i}}{F^t_{T_i} + \eta} \tag{6.4}$$

where $M^t_{T_i}$ is the mean of the overlap ratios and $F^t_{T_i}$ is extent of fluctuation of overlap ratios for t frames computed as in [138].

Self evaluation measure for candidate tracker is measured by *Euclidean distance* shift between centres of the previous and current bounding boxes denoted by distance $d^t_{T_i}$, and is given by

$$\exp\left(\frac{-1}{2\sigma^2_{T_i}}\left(F^t_i\right)^2\right) \tag{6.5}$$

where $\sigma_{E_i}$is the average of the width and height of bounding box given by tracker i.

Tracker robustness degree is the linear sum of both pair-wise evaluation and self-evaluation. After evaluation of the overall reliability for each one, the tracker with the highest robustness score is selected and its tracking result is taken for the current frame:

$$R^t\left(T_i\right) = R^t_{pair}\left(T_i\right) + R^t_{self}\left(T_i\right) \tag{6.6}$$

See Fig. 6.3.

6.4 Experimentation and Analysis

For experimentation, the DCF method is employed for forming candidate trackers [63]. MATLAB 2017a is used for tracker implementation. A GeForce GTX 1080 Titanium GPU with compute capability 6.1 and 12GB memory is used for experimentation and testing. The MatConvNet toolbox is used for extraction of deep features from VGG-19 [116]. This framework runs at about 1 FPS on CPU. The GPU

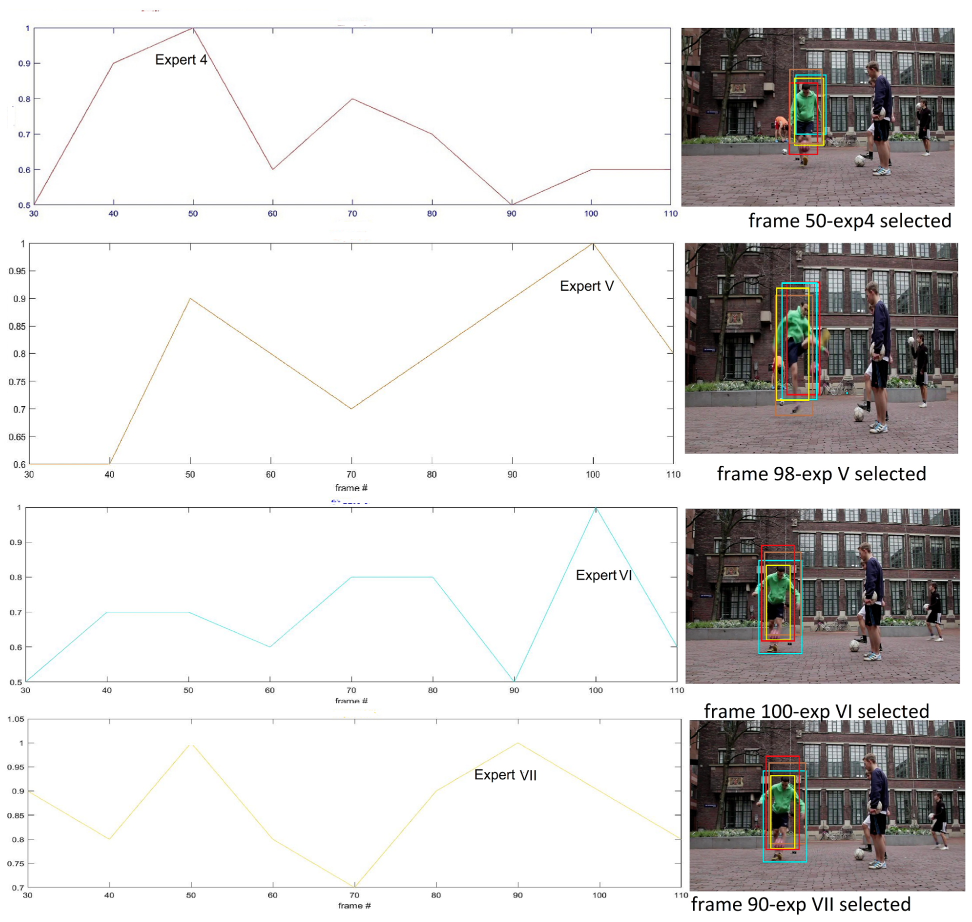


Figure 6.3: *Left*. Tracker with peak robustness measure of 1 will be chosen for tracking for that frame. *Right*. Depicts bounding-box tracking outputs for all candidate trackers.

version of MCCT tracker runs at about 7 FPS.

The evaluation measure for tracking is chosen as the *area under success curve* (AUC). It is plotted between overlap threshold and success rate. The overlap between the ground truth and tracking result is calculated as follows:

$$O(T, GT) = \frac{|T \cap GT|}{|T \cup GT|} \tag{6.7}$$

If $O(T, GT)$ is greater than an overlap threshold, then tracking is a success. The success rate is the ratio of successfully tracked frames to the total number of frames in the sequence.

AUC is the average of all success rates at different overlap thresholds when these threshold values are evenly distributed [79].

In *One-Pass evaluation* (OPE), the tracker is initialized in the first frame, so tracking is sensitive to frame number and bounding box spatial location for the target. The average success is calculated.

Spatial Robustness evaluation (SRE), tracking is done by starting it at various positions employing 4 spatial shifts around the centre, corner and scale variations. Amount of this shift is 10% target with scale variations of 0.8, 0.9, 1.1, and 1.2. The average performance is reported with tracker evaluated 12 times.

For *temporal robustness evaluation* (TRE), the video is divided into 20 segments and tracking is performed for 20 segments with different initial frames each time. Average performance is calculated for all video segments.

We employed two datasets for evaluation. One is the $OTB - 2015$ [141] dataset and the other one is $VOT2016$ [79, 80]. Trackers are evaluated using an overlap threshold of 0.5. Overlap success plots using *one-pass evaluation* are generated as well for OTB-2015.

Refer to Fig. 6.4 for OPE comparative values for mean distance precision at a threshold of 20 pixels, mean overlap precision at 0.5 threshold and AUC for OTB-2015 dataset.

Also under challenging attributes like low illumination, clutter, occlusion or motion blur, OPE is evaluated for the dataset for experimental comparison to other state of art trackers like MEEM [158], ECO [37], ECO-SAL [14], VTD [82], VTS [83], LSK [97], FRAG [11], CXT [36] and baseline MCCT tracker [138]. Refer to Fig. 6.5 for OPE results for those.

The visual object-tracking challenge 2016 (VOT2016) [79] benchmark provides an evaluation toolkit. It re-initializes the tracker to the correct position to continue tracking when tracking failure occurs. In VOT2016, the expected average overlap (EAO) is used for ranking trackers, which combine the raw values of per-frame

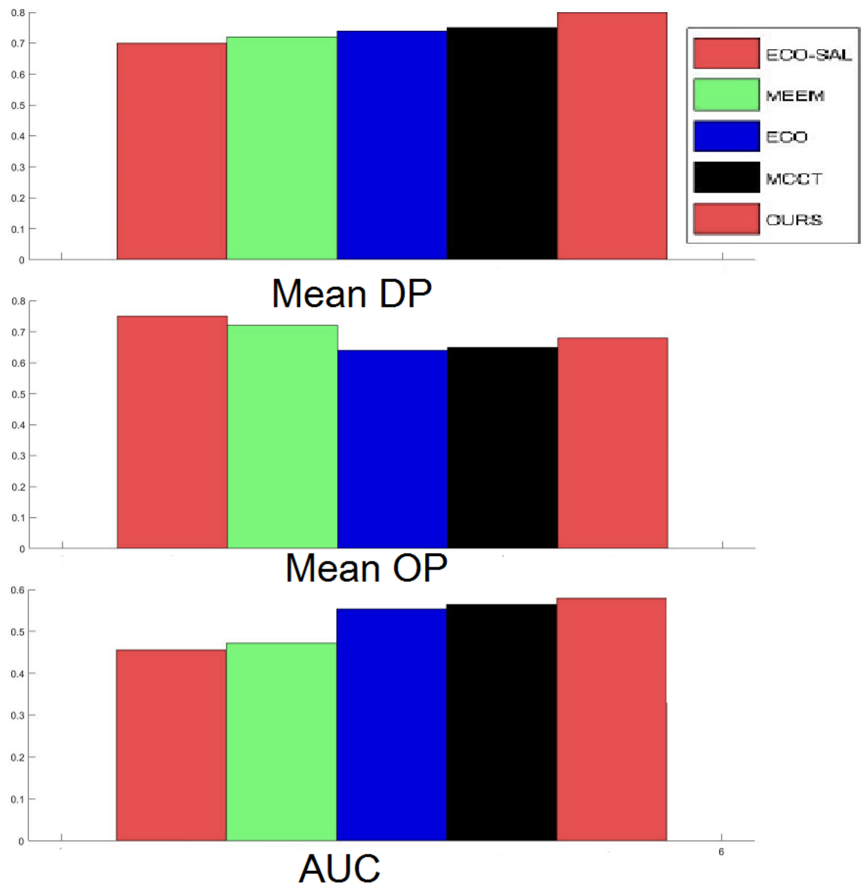


Figure 6.4: Mean DP, OP and AUC results on OTB-2015 sequence

accuracies and failures (tracker robustness). EAO is an estimation of the average overlap; it is expected for a tracker to attain on several smaller sequences [80]. See Fig. 6.6.

Table 6.2 details the quantitative comparison between the EAO scores, accuracy and failure rate for comparative study. Fig. 6.5 provides an insight into trackers numbered in order of EAO scores.

OPE, SRE (spatial robustness evaluation) and TRE (temporal robustness evalu-

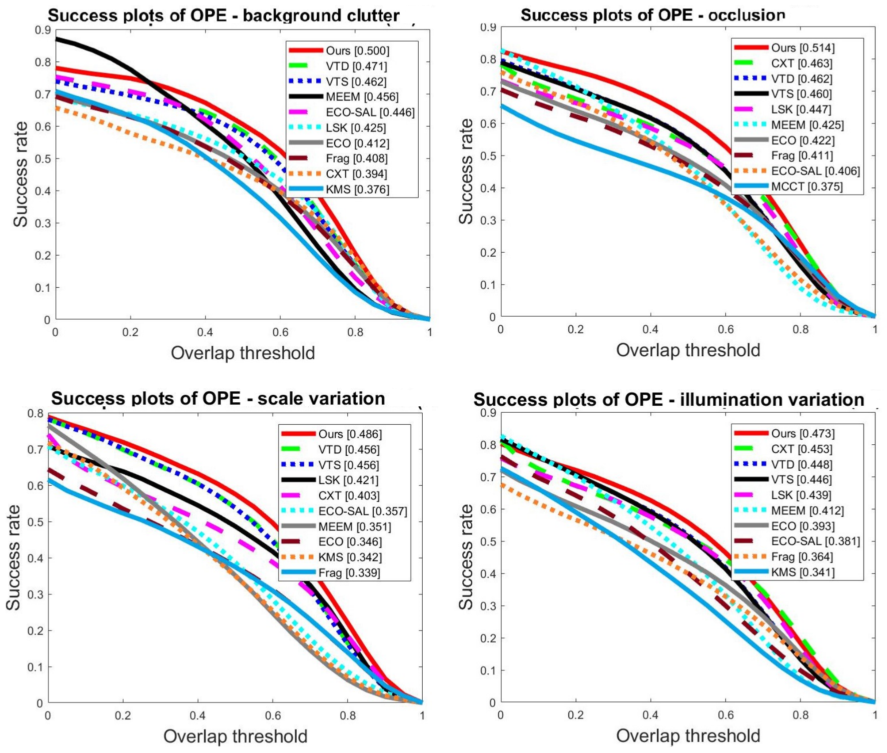


Figure 6.5: One-pass evaluation OPE plot for various attributes.

Table 6.2: Accuracy, robustness and EAO scores for various state-of-the-art trackers

Quantitative evaluation					
Metrics	MCCT [138]	ECO [37]	ECO-SAL [14]	MEEM [158]	ours
Accuracy	0.56	0.51	0.59	0.55	0.58
Failure rate	0.74	0.72	0.91	0.85	0.71
EAO score	0.37	0.35	0.36	0.31	0.39

Table 6.3: Comparison to the baseline: One pass evaluation (OPE)

One-pass Evaluation-OTB2015				
	OC	LI	SV	BKC
MCCT	0.71	0.71	0.76	0.64
Ours	0.76	0.61	0.78	0.71

Table 6.4: Comparison to the baseline: Spatial robustness evaluation (SRE)

Robustness analysis-OTB2015				
	OC	LI	SV	BKC
MCCT	0.61	0.69	0.66	0.64
Ours	0.66	0.61	0.68	0.71

ation) results for OTB2015, under various challenging attributes, are shown in Tables 6.3, 6.4and 6.5, respectively. We denote the occlusion challenge by *OC*, low illumination as *LI*, scale variation challenge as *SV*, and background clutter as *BKC*.

It can be seen that under *low illumination* (LI) conditions, our framework results have not shown much improvement over the MCCT framework. The reason is that saliency prediction is less discriminant in such scenes.

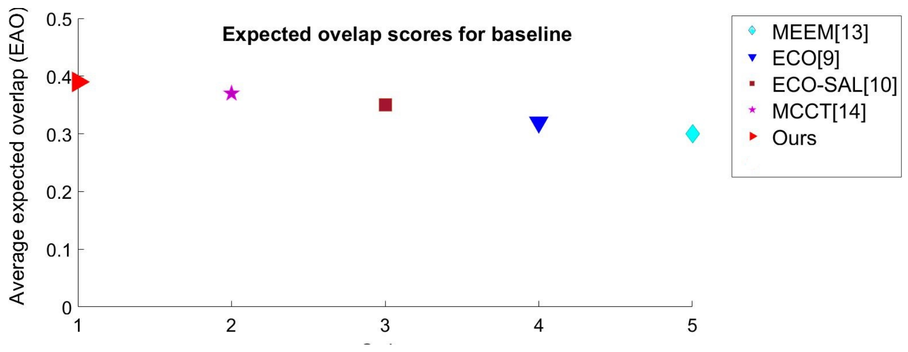


Figure 6.6: EAO curve with trackers labelled in order of EAO values

Table 6.5: Comparison to the baseline: Temporal robustness evaluation (TRE)

Robustness analysis-OTB2015				
	OC	LI	SV	BKC
MCCT	0.74	0.79	0.67	0.67
Ours	0.66	0.71	0.78	0.70

6.5 Summary

We propose novel saliency-guided correlation tracking with 7 candidate trackers. Our experimental results demonstrate that employing saliency features as complementary features in multi-cue-based ensemble tracking helps improve the tracking performance. We suggest that saliency map-based region semantic information has further potentials when used as a higher-level tracking feature.

Moreover, we have shown that ensemble tracking benefited from improved candidate tracker robustness. We assume that this feature can benefit many other baseline tracking paradigms as well. This scheme also helps to reduce the tracking model drift problem caused by occluders. It will prevent trackers corruption over time. Finally, extensive experiments show that our tracker outperformed state-of-the-art methods on popular benchmark datasets.

Chapter 7

MOT Comparative Analysis

We perform an experimental comparative study for model-based GMM Kalman tracker and tracking by detection technique, using two deep learning models (Faster-RCNN and Yolo-v3). For optimum analysis of the strength and the weakness of each of the tracking algorithms under various environmental challenges, we categorised the video sequences used for the experimental comparison into three main attributes. These attributes includes low illumination, occluded and cluttered warehouse areas. These sequences are chosen from warehouse recorded video data. This chapter briefly reports the outcomes for this experimental study.

7.1 Experimental Sequences

Three video sequences are chosen for experimental comparison study, for three main challenges. Refer to Table 7.1 for the sequence characteristics. Columns 3 and 4 list the number of frames and the number of tracks in each sequence.

Table 7.1: Sequences used for comparison

Sequences	Attribute	Frames	Tracks
Seq1	Low illumination areas	150	7
Seq2	Occlusions	614	21
Seq3	Cluttered areas	614	9

We evaluate the results under considered attributes, for each studied tracking technique separately. It helps to obtain an insight into which method is most suitable to which warehouse scene challenge.

We also formulate a ground-truth target location dataset for the corresponding target's ground-truth positions and ground-truth trajectories manually for the considered sequences. This help in robust tracking evaluation results. For example, refer to Figs. 7.1 and 7.2 for the ground truth for the objects. It also shows the corresponding detections from Faster-RCNN detector.

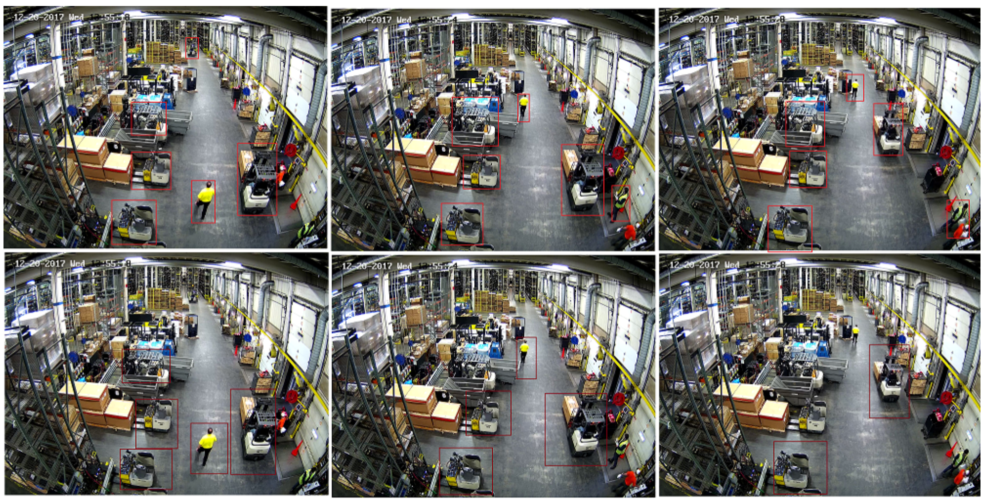


Figure 7.1: *Top*. Ground truth objects are shown in red. *Bottom*. Faster-RCNN detections are shown in bottom frames in brown colour

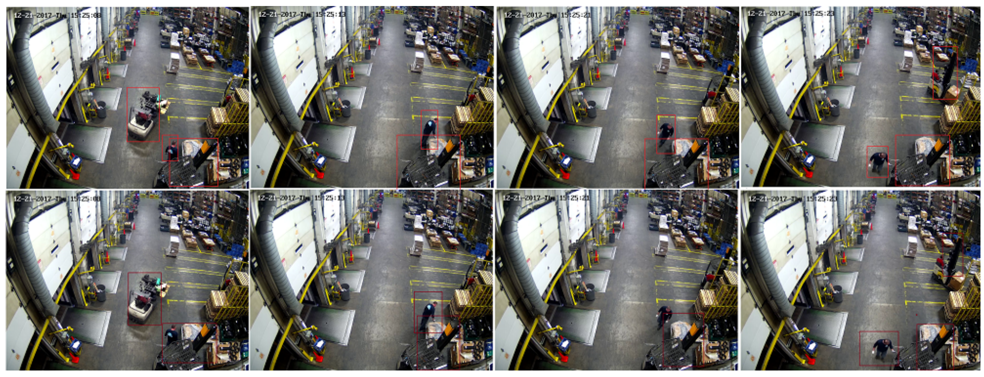


Figure 7.2: *Top*. Ground truth objects are shown in red. *Bottom*. Faster-RCNN detections are shown in bottom frames in brown colour

For fair comparisons, corresponding parameters of each tested method are fixed for all the considered sequences. Refer to Figs. 7.3, 7.4 and 7.5 for qualitative evaluation results for GMM, Faster-RCNN and Yolo-v3 based trackers.

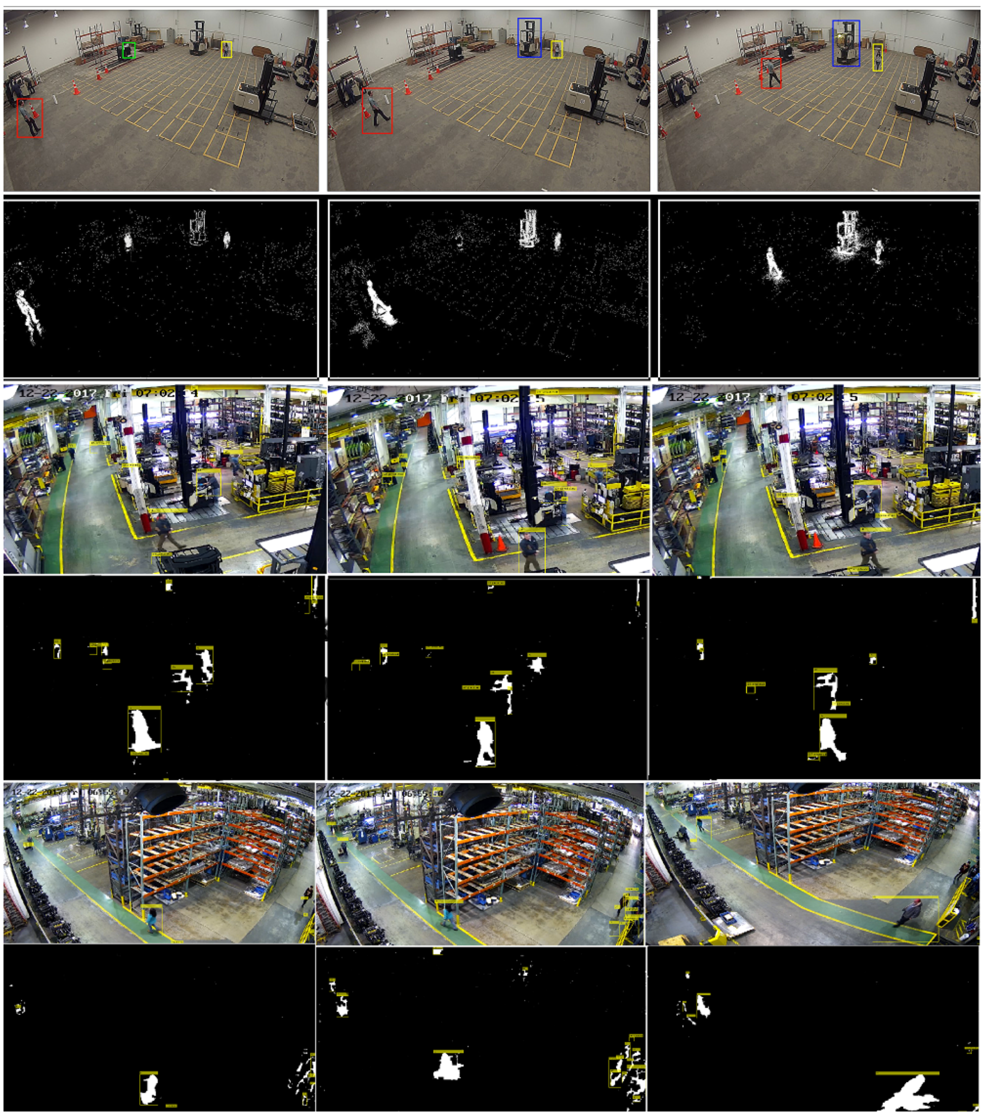


Figure 7.3: For three sequences, resulting tracked bounding boxes are shown. Top row depicts the tracked bounding boxes for each sequence and bottom row shows the corresponding foreground regions.

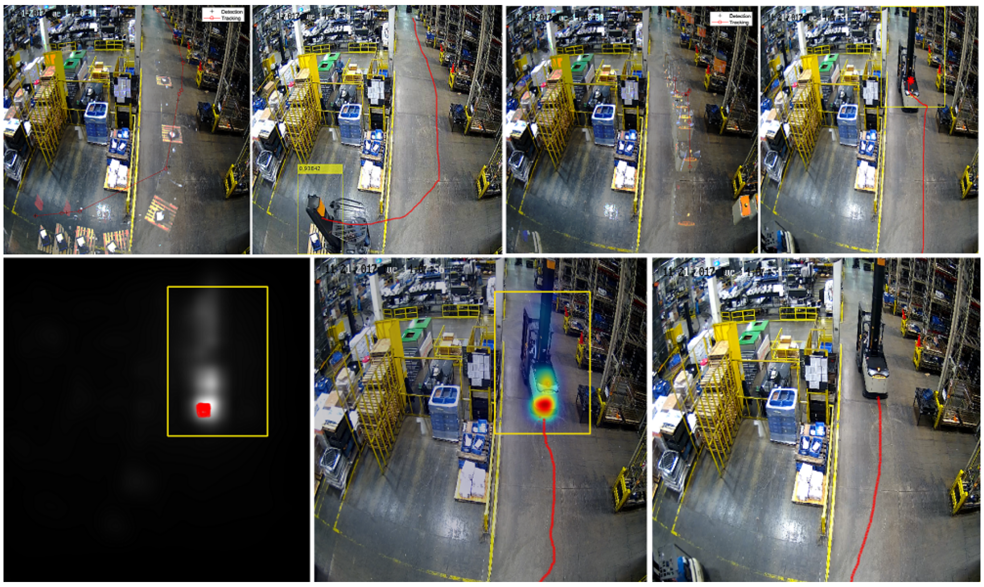


Figure 7.4: Top row depicts the trajectory computed by faster-RCNN based tracker. *Bottom row, left*. Saliency map with detected centroid in red. *Middle*. Trajectory estimated by Kalman. *Right*. Ground truth trajectory

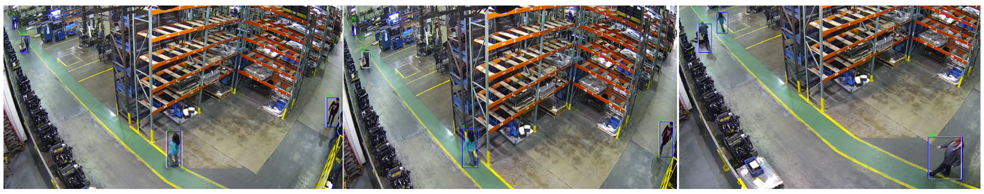
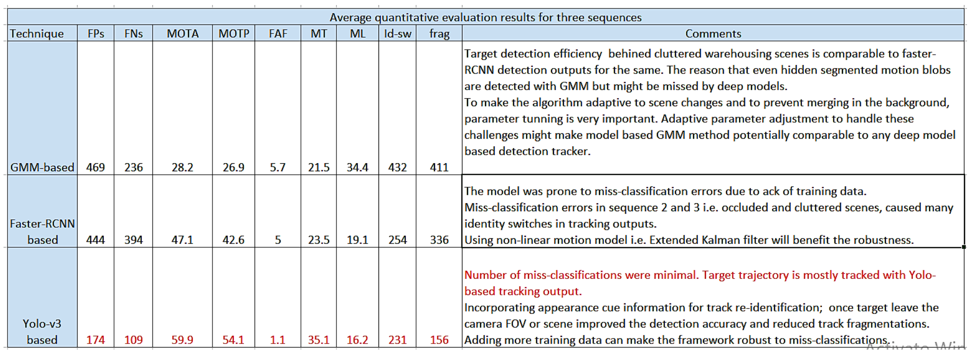


Figure 7.5: Multiple target tracking. *Left to right*. Frames 35, 54 and 88 using Yolo-v3 tracker with bounding box shown for each tracked target

7.2 Evaluation metrics

We observe that there is no single tracking approach that is ideal to handle tracking targets of interest, under all the three mentioned challenging attributes. For fair evaluation, many vital factors such as the target of interest position accuracy (bounding box location), robustness over different appearance or scale changes,

Table 7.2: Average quantitative comparison for model and deep-learning based tracker. Comments are listed in last column.



Average quantitative evaluation results for three sequences										
Technique	FPs	FNs	MOTA	MOTP	FAF	MT	ML	Id-sw	frag	Comments
GMM-based	469	236	28.2	26.9	5.7	21.5	34.4	432	411	Target detection efficiency behined cluttered warehousing scenes is comparable to faster-RCNN detection outputs for the same. The reason that even hidden segmented motion blobs are detected with GMM but might be missed by deep models. To make the algorithm adaptive to scene changes and to prevent merging in the background, parameter tunning is very important. Adaptive parameter adjustment to handle these challenges might make model based GMM method potentially comparable to any deep model based detection tracker.
Faster-RCNN based	444	394	47.1	42.6	5	23.5	19.1	254	336	The model was prone to miss-classification errors due to ack of training data. Miss-classification errors in sequence 2 and 3 i.e. occluded and cluttered scenes, caused many identity switches in tracking outputs. Using non-linear motion model i.e. Extended Kalman filter will benefit the robustness.
Yolo-v3 based	174	109	59.9	54.1	1.1	35.1	16.2	231	156	Number of miss-classifications were minimal. Target trajectory is mostly tracked with Yolo-based tracking output. Incorporating appearance cue information for track re-identification; once target leave the camera FOV or scene improved the detection accuracy and reduced track fragmentations. Adding more training data can make the framework robust to miss-classifications.

tracking time efficiency, memory consumption, and ease of usage for tracking method, are studied. Even in one frame with the tracking output and ground-truth object state, there could be several qualitative and quantitative evaluation metrics to measure accuracy and tracking performance.

Generally, for the multiple object tracking problem, quantitative performance is evaluated using two measures i.e. multiple object tracking accuracy (MOTA) and multiple object tracking precision (MOTP) [101], [27]. MOTA takes considers all configuration errors for tracker results (i.e. false-positives, false negatives), averaged over all the frames:

$$\text{MOTA} = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{7.1}$$

where FN_t is the number of false-negatives or misses, FP_t is the number of false-positives, and IDSW is the number of track identity switches, where t is the frame index and GT is the number of ground truth objects.

The multiple object tracking precision (MOTP) is the average of the dissimilarity between all true-positives and their corresponding ground truth locations:

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{7.2}$$

where c_t denotes the total number of matches in frame t and $d_{t,i}$ is the bounding box overlap of target i with its assigned ground truth object.

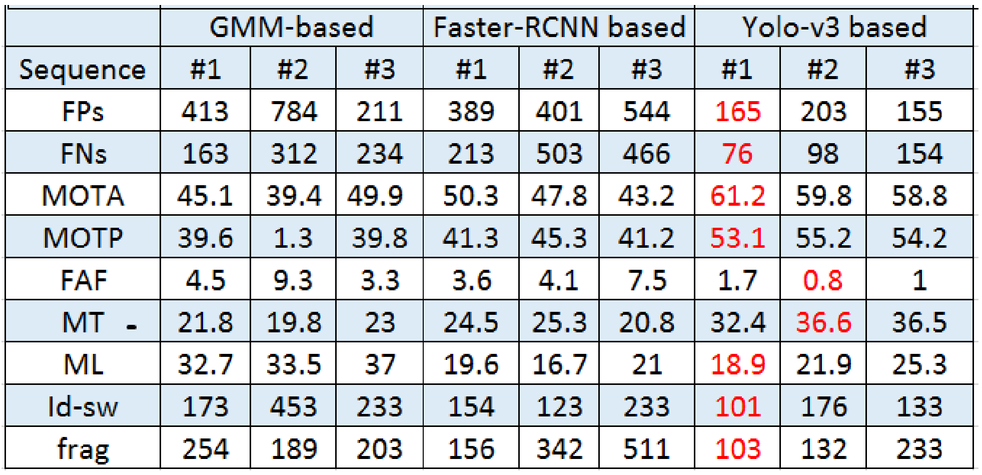
Quantitative comparison results have been summarised in Table 7.3 for each method for the three considered sequences. It is seen that targets are consistently tracked over large parts of Sequences 1 and 3 for the model-based GMM method. Also, targets are tracked over considerable parts of all Sequences 1, 2 and 3 using tracking by deep model-based detection.

Table 7.2 lists the studied specific observations regarding model and deep-learning-based tracking methods. It also shows the quantitative comparison of the two categories of methods compared, based on MOT evaluation metrics. It can be seen that Yolo-v3 based tracking by detection techniques gave the best tracking quality and best quantitative results for the considered warehouse challenging scenes.

7.3 Summary

Evaluation of the model-based GMM technique and deep-learning-based tracking frameworks in an industrial environment of warehouse is carried out. We conclude that when a deep learning model is supplied with enough and variable training data (with targets in every pose, scale, background or position), they are accurate at target localisation and in avoiding tracking errors as compared to model-based methods. Robustness of target detection accuracy is very important for seamless

Table 7.3: Comparison of tracking results on three challenging sequences using MOT evaluation measures [100]



	GMM-based			Faster-RCNN based			Yolo-v3 based		
Sequence	#1	#2	#3	#1	#2	#3	#1	#2	#3
FPs	413	784	211	389	401	544	165	203	155
FNs	163	312	234	213	503	466	76	98	154
MOTA	45.1	39.4	49.9	50.3	47.8	43.2	61.2	59.8	58.8
MOTP	39.6	1.3	39.8	41.3	45.3	41.2	53.1	55.2	54.2
FAF	4.5	9.3	3.3	3.6	4.1	7.5	1.7	0.8	1
MT	21.8	19.8	23	24.5	25.3	20.8	32.4	36.6	36.5
ML	32.7	33.5	37	19.6	16.7	21	18.9	21.9	25.3
Id-sw	173	453	233	154	123	233	101	176	133
frag	254	189	203	156	342	511	103	132	233

target tracking, with minimal track fragmentation and identity switching scenarios.

Any deep model needs to be supplied with training data, that includes targets under a variety of challenging attributes and background complexity. Otherwise, a single trained model might only be good enough for only specific custom tasks it is trained for.

A model trained for one set of attributes and video frames might not be good enough for a set of others. Robustness is a relative measure in this regard and highly variable as per the scene statistics.

Using deep features and models for simplifying vital tracking sub-tasks other than only target detection and localisation is also valid. The tasks that can benefit might include cost affinity computation for assignment of detections to tracks and handling track re-identification problem in multiple target tracking.

We observe that model-based methods are better at finding regions of interest but comparably see fewer variables according to scene attributes. Possibly, a hybrid model and deep feature-learned network can work best in the warehouse's challenging scenarios.

We assume that optimum quality of MOT results will lay the ground for subsequent higher-level computer vision tasks such as people action recognition and process improvement. Furthermore, we suggest that such a detection and tracking framework would be useful for increasing work safety at the warehouses.

Chapter 8

Conclusions and Future Work

This chapter reports the conclusions and possible future research directions for tracking multiple targets in warehouses.

8.1 Detection of Occluded Targets

This work is mainly based on using tracking by detection technique where multiple objects are tracked, based on detection outputs from a detector. Initially, we used one of the model-based motion segmentation techniques (i.e. GMM) to extract moving objects from the scene in cluttered warehouse environments. Refer to Fig. 8.1 for a cluttered warehouse scene. Most of the pedestrian objects are seen occluded behind the warehouse clutter, occupying fewer pixels (small ROIs).

Any algorithm that can segment them, among the cluttered background, will be efficient in localising all the instances of the respective class. Since robustness of most of the tracking algorithms is limited by the quality of localisation results, GMM proved to be an effective technique in the extraction of instances of the pedestrian class; the only limitation is that it has to be moving instances of the corresponding class. Moving (or partially or fully occluded) forklift trucks are also detected, with a lesser number of errors (false positives and false negatives), where it was able to extract the forklift moving blobs.

8.2 Transfer Learning

To assign category labels to the respective segmented moving blobs, a deep architecture is retrained, by transfer learning the Inception v3 pre-trained model. Transfer learning is also employed in re-training faster-RCNN with pre-trained AlexNet model and Yolo-v3 with Darknet53 model for this research.

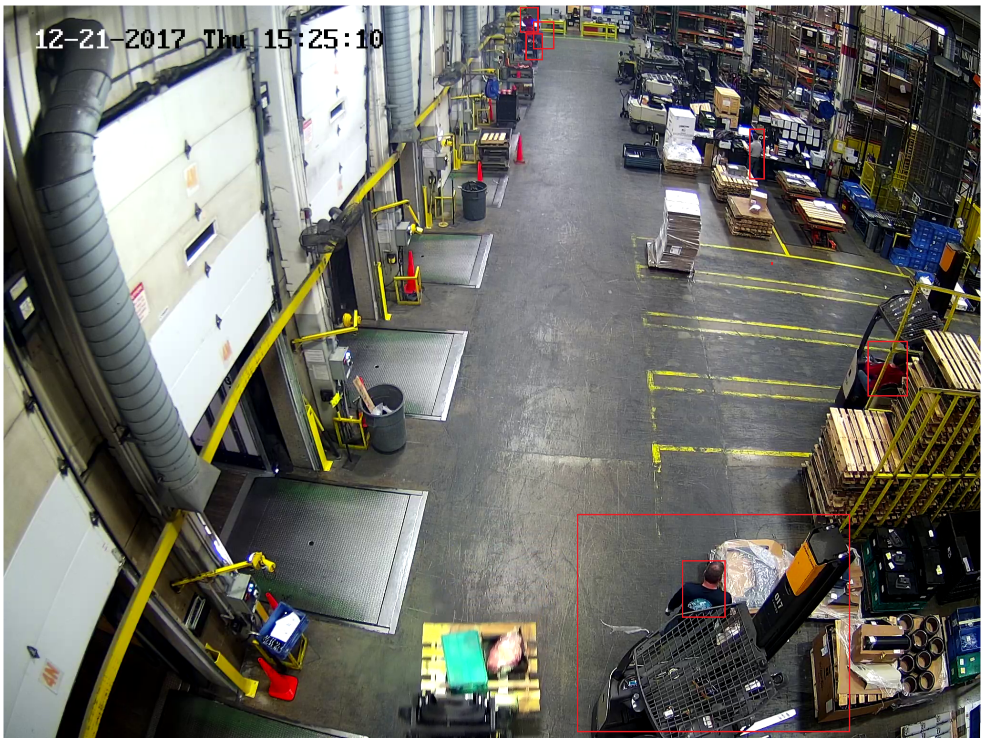


Figure 8.1: Pedestrians are marked in a red bounding box, occluded partially behind the cluttered warehouse background.

Transfer learning is used to customise a pre-trained deep model to a given classification task, by taking advantage of the learned feature maps to classify new classes of objects. A new classifier is added and retrained from scratch, where only feature maps which are learned previously for the corresponding dataset are re-purposed. Since most of the pre-trained models are trained for generic data, re-using them helps to classify new objects, avoiding overfitting.

Alternatively, by fine tuning the higher-order feature maps representation for the model employed, they are made more specifIc to the requisite custom task. By retraining a few top layers of the frozen network and a new classifier for object categories, the model is re-purposed. For this work, with only top-layer re-training, we obtain 99.5 classification accuracy. Transfer learning proved to be an effective technique for obtaining accurate detection outputs when the number of training images are limited. It also took less training time for the model.

In our case, we have acquired our own recorded warehouse dataset. The dataset is very variable with respect to multiple forklift truck and pedestrian poses seen in the videos. The videos have forklift trucks with varying models, make and shape, with every truck is designed for different functions inside warehouses. It includes stock pickers, reach trucks, and others.

Transfer learning worked well, especially when the features of the source and target dataset are quite similar and the overlap is quite significant.

On the other hand, we observe that our retrained model is not able to generalise well for the test images of those forklift models for which we have a relatively small number of training images in our dataset. To obtain robust target classification results and prevent overfitting, a deep learning model needs to be supplied with a variety of truck model images, in all possible poses/positions of forklift forks and shapes.

Thus we conclude that we could have improved the detection/classification accuracy if we were able to get the maximum dataset annotated and train the network from scratch, especially for such miss classification cases. We also assume that warehouses are commercial premises and tracking targets inside such environments needs to be commercially feasible, with little ambiguity or error. We suggest that any MOT framework for such workplaces needs to be robust to detector quality limitations. A model trained from scratch, solely for warehouse custom categories tracking, under various environmental challenges it has to offer, would be the most feasible solution. It also needs better computational resources like graphical processing units (GPUs) and the bulk of well annotated training data for model training from scratch.

Varying background configurations for such forklift trucks also need to be addressed, to be localised in any background statistics, at least for the considered warehouse where the data was acquired. Few motion blobs extracted by the GMM algorithm to be supplied as region proposal in the proposed tracking prototype as in [3], was not accurately classified by retrained Inception v3 model. Due to the bad quality of candidate region hypotheses, tracking performance is somewhat limited with an increased number of false positives and negatives in such cases.

8.3 Kalman Filter Limitations

Kalman filter is used to estimate the state of linear dynamic systems. The Kalman filter's linearity limitation is a bottleneck for estimating variables for non-linear motion.

For non-linear systems, in general, it might not be possible to have optimal state estimation in closed form. There are many variations of Kalman filters that might be worth to be tested in this case. An alternative to a traditional Kalman filter could be an unscented Kalman filter (UKF) for tracking non-linear motions, or also an extended Kalman filter (EKF). We conclude that employing EKF for targets trajectory estimation gave robust trajectory results, close to the ground truth.

Also, a particle filter propagates, by estimating state estimates (known as *particles*) distributed according to a probability density function (pdf) of to the true state [117]. Our experiments with particle filters were comparable to the EKF for tracking forklift trucks.

8.4 Track Re-Identification in MOT

Deep features are being used in simplifying tracking sub-tasks, other than the detection and localisation of targets of interest. Multiple object tracking problems also include computing cost for the assignment of corresponding detections to tracks.

The track re-identification problem in MOT is also a big challenge. It is also one of the main reasons for switches in track identity over time and the track fragmentation problem in MOT. For warehouse cases, when a target leaves the field of view of the camera and appears again after some time in the same view, it has to be associated with the same track identity as before. It could also be the case when targets come very close to each other and merge their respective paths, we need to re-identify the respective targets, against those trajectory fragments.

Though, to lessen the number of track ID switches, further improvement modules need to be incorporated in the tracking framework. We conclude that once the corresponding track has disappeared from the scene, associating the corresponding detection to the same track needs to consider many variables in general.

Incorporating the appearance space information for track assignments is very vital in this regard, in addition to the use of various metric distance measures. These distance measures might include the Euclidean distance, Cosine distance and others. Solving for such distance measure for track assignment, might not be the only validation needed for such assignment. Computing reliable appearance cues fast and easy will aid to solve this track validation robustly. Finding specific track deviations in appearance space is vital and will help to identify and re-identify the targets over longer sequences of videos.

We anticipate various trade-offs when solving such issues. For example, an offline CNN-trained model for pedestrian re-identification has already shown improved results under track-identity switch scenarios [139]. Such an idea for appearance-space distance measures can be extended to other object categories such as trucks in warehouses or cars on roads.

8.5 Pixel Saliency Map Significance

Saliency map computation for the specific region of interest proves to be very effective in various ways in this research. These maps are able to mark vital salient information in the image and have very satisfactory results with occluded objects in warehouse scenes. This method is also computationally fast. We have employed these maps to achieve multiple targeted tasks, as stated below:

1. We tested the computation of static and motion saliency maps by [154] and [65]. In our work [3], it is used for improving quality of GMM-based foreground quality. The obtained foregrounds were improved with few redundant pixels from the background, as part of the foreground and fewer foreground holes.

2. We refined the location of a centroid computed by faster-RCNN bounding box criterion, based on ROI pixel saliency map [65, 149]. We used improved centroid computation, using the most salient region of the ROI as reference. The resulting centroids recomputed for trajectory correction by the linear Kalman filter were more consistent and accurate with respect to ground-truth centroid points [1].

3. We employed these maps in validating the track assignment problem for MOT. We concluded that *appearance-saliency-map guided data association measure* can be used to verify the track identity especially in cases of occluding tracks or multiple bounding boxes on the same target. A saliency distribution dissimilarity measure between a detected ROI and predicted candidate track locations is described by the Bhattacharyya coefficient. It could be used to validate the

track association problem, if used in conjunction with the Mahalanobis distance measure.

4. For deep CNNs, features from the last convolution layers are more closely related to object category level semantic information. These feature maps though are robust to intra-class appearance variations but fail the objective to locate targets precisely. Earlier convolution layer outputs are more appropriate for target object fine-spatial appearance information, but not for the semantics [102]. We propose that *saliency is a discriminant feature* that aids in differentiating the target from the background and can be useful and efficient way to attain robustness against target appearance changes at a semantically higher level.

5. We also used *saliency features* as guiding high-level semantic features in ensemble tracking. Learning correlation filters for saliency along with other features, will aid the tracking quality for each tracker in an ensemble. Any tracker in an ensemble employing this semantic aware higher-level feature will benefit in track accuracy and precision and thus achieve final tracking robustness against appearance and viewpoint changes.

6. We also concluded that the saliency map-based region semantic feature can benefit many other baseline tracking paradigms. This also aids to solve the trackers drift problem in ensemble tracking, caused by occluders. It will also prevent trackers corruption over time by accumulation of the tracking drifts in an ensemble.

8.6 Increasing Warehouse Safety

Various ways to track multiple targets are proposed in this work. We suggest that the need to improve warehouse work safety is vital. The corresponding tracking prototype might be extended further into a safety system, to enhance work safety in such premises.

For example, see Fig. 8.2 depicting the suggested added modules in a proposed tracking prototype [3], that can help to improve situational awareness for the workers. It includes three additional modules other than object detector, classifier and a tracker.

First one outputs targets bird eye's views based on inverse perspective mapping (IPM). Such information will be forwarded to a collision probability prediction

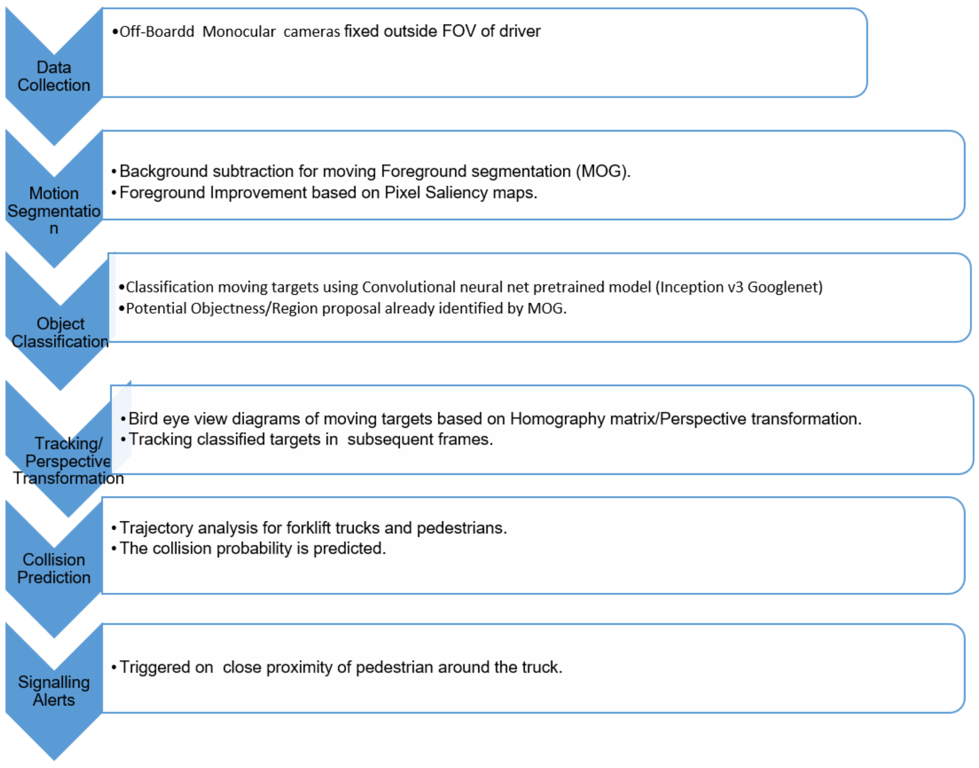

Data Collection
•Off-Boardd Monocular cameras fixed outside FOV of driver
Motion Segmentation
•Background subtraction for moving Foreground segmentation (MOG).
•Foreground Improvement based on Pixel Saliency maps.
Object Classification
•Classification moving targets using Convolutional neural net pretrained model (Inception v3 Googlenet)
•Potential Objectness/Region proposal already identified by MOG.
Tracking/ Perspective Transformation
•Bird eye view diagrams of moving targets based on Homography matrix/Perspective transformation.
•Tracking classified targets in subsequent frames.
Collision Prediction
•Trajectory analysis for forklift trucks and pedestrians.
•The collision probability is predicted.
Signalling Alerts
•Triggered on close proximity of pedestrian around the truck.

Figure 8.2: Suggested prototype for increasing work-related safety in warehouse

module. The third module which is signalling system that can be triggered when it is detected that target lies in close proximity to the predicted trajectory of the other (any hazard could be be a predicted forklift truck path and a pedestrian being in close proximity of that path).

Thus, the corresponding tracked targets can be mapped onto a ground plane (i.e. into a top-down bird's eye view). Refer to Figure 8.3 for a shot depicting the two forklift trucks, marked by red circles to indicate the positions. IPM is a mathematical technique, where a coordinate system is transferred from one perspective to another [127]. Homography matrix H computation is performed for the corresponding mapping from pixel into real-world coordinates. In one of the ways, these points can be obtained by choosing the same four points in the source and destination images and solving for H in a linear system framework. H is a 3×3 matrix such that:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{8.1}$$

where x and y are pixel coordinates for the source image and X, Y are the pixel coordinates for the destination image.This information is communicated to the next module, where an unwanted interaction risk is predicted in advance, for the targets lying close to each other.

The target's distance is recorded in pixel coordinates and distance is converted into metres (might be based on some manual calibration criterion, like the number of pixels equivalent to 1 metre). Unwanted target interaction is defined as an event, for example where the recent forklift path is predicted and it is observed that it lies very close to a recently observed track for a pedestrian. Degree of unwanted interaction [16] is based on the corresponding distance between targets of interest, their speed and direction.

For the task of predicting unwanted interactions between targets (i.e. between forklift trucks and pedestrians), forklift's trajectories need to predicted periodically, a few seconds ahead in up-coming future frames. It is suggested for the corresponding prediction algorithm to keep a record for all the pedestrian positions around a predicted forklift truck trajectory, in pixel coordinates within a window of a few seconds. Around predicted forklift truck positions, it is suggested to perform a fixed radius nearest neighbour search [33]. For instance, they can be the four quadrants around the forklift truck (front-left, front-right, rear-left, rear-right). The range for search distances and prediction time frames should be chosen carefully and will highly depend on scene statistics.

8.7 Approaches To Improve MOT Accuracy

To obtain seamless output tracking results for multiple targets, detection quality is vital. For detection based tracking approaches, robustness of the object detector need to be improved. It is being observed that deep learning-based object detectors; when trained with bulk of training data which is dynamic in terms of scale, viewpoint changes and varying poses, proved to be more robust than model based object detection techniques.

Multiple object tracking accuracy also depend a lot on the capability of the tracking algorithm to handle track identity switches and swaps. Identity of the multiple tracks need to be consistant in various challenging scenes like partial or full occlusions, clutter, multiple scales of same track, targets with similar appearance and illumination changes in scenes. One of the few challenges that affect MOT accuracy is solving track-reidentification problem. Associating the same identity even after the object leaves the scene in camera FOV and reappear after few video frames, is crucial to avoid false identity switches. Track association criterion for the algorithm needs to be improved to cater such challenges i.e. unexpected entry and exit of targets. An improved saliency-colour based dissimilarity measure based track ReID module is proposed in this work to solve such miss-assignment in MOT problems.

For tracking linear motion, Kalman filter proved to be robust for tracking under occlusions and missed targets. Incorporation of saliency-map based localisation information to improve the tracking trajectory is tested and verified in this work.

Quantitative evaluation is stated in earlier chapters based on various MOT variables like type of object detection for detection-based tracking, using saliency-based multiple object tracking, type of detection association criterion and ensemle-based tracking mechanisms. Online MOT challenge dataset is also used for online evaluation and benchmarking for MOT [100].

For warehouse data, all the evaluation is carried based on novel ground truth formulation for pre-captured video sequences. Because, there is no direct evaluation yet on this confidential dataset, a quantifiable comparison(where possible) with other trackers is performed based on mentioned variables under various challenging scenarios.

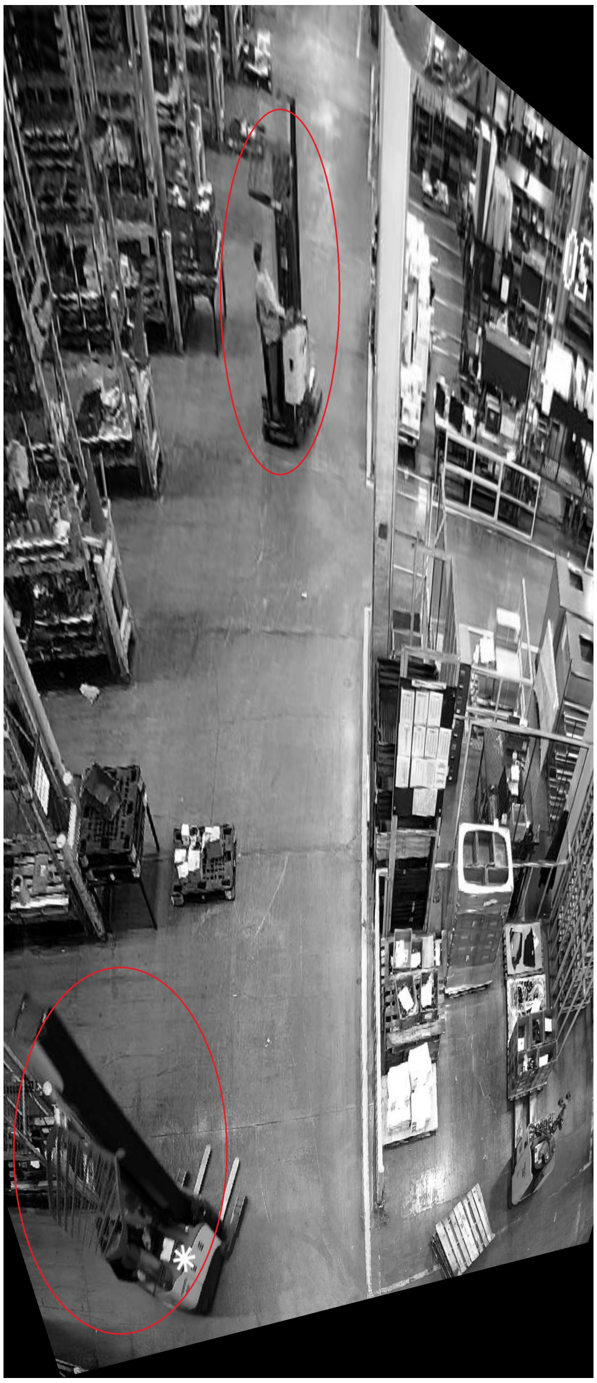


Figure 8.3: A shot showing two forklift truck targets shown with red circles.

Appendix A

Datasets

A.1 Data Recording Phase 1

For Phase 1 of data recording, collection of video data from a warehouse test environment at the Crown facility is done. With the help of Crown engineers, we set up data collection hardware and collect video data of pedestrians and forklifts under different specific test scenarios and configurations.

It includes different camera viewpoints with varying backgrounds, different illumination settings or un-occluded and occluded views of pedestrian and forklift activities. We use GoPro (Hero Edition) cameras for this work.

Refer to Fig. A.1 for the video shots acquired with Camera 1.

Figure A.1: Camera 1 recorded video thumbnails

A.2 Data Recording Phase 2

Four point, tilt and zoom (PTZ) cameras are fixed at different viewpoints in a real production warehouse as in Fig. A.2 and Fig. A.3.

Video data of duration 40 minutes (10 minutes from each of the four cameras) is recorded at the Crown plant. The videos with date stamps between $2017-12-14$ to $2017-12-19$ were captured with cameras facing the directions shown in Fig. A.2. There are 100 recorded videos in total with this camera configuration and settings.

Figures A.4, A.5, A.6, A.7 are shots for recorded videos with details mentioned in the respective captions.

At the next stage, the pan function on the cameras is adjusted to change the view to the approximate directions shown in Fig. A.3. These are the videos with datestamps between $2017-12-20$ to $2017-12-22$. There are 60 videos recorded in this setting.

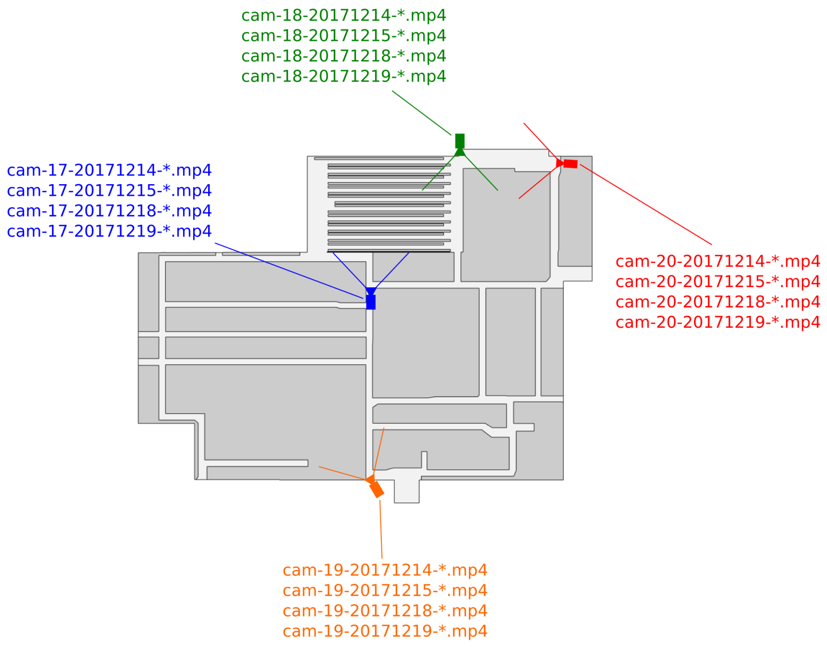


Figure A.2: Camera 17,18,19,20 positions for acquiring 100 videos data

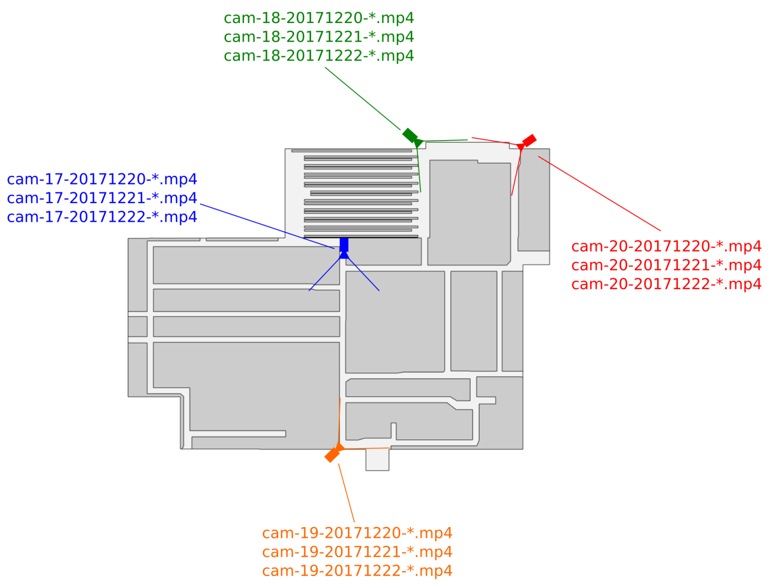


Figure A.3: Camera $17, 18, 19, 20$ positions for acquiring 60 videos data

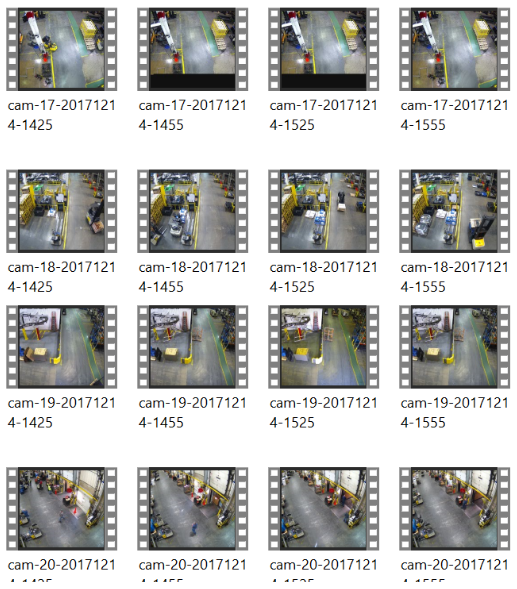


Figure A.4: Video data thumbnails on $2017-12-14$

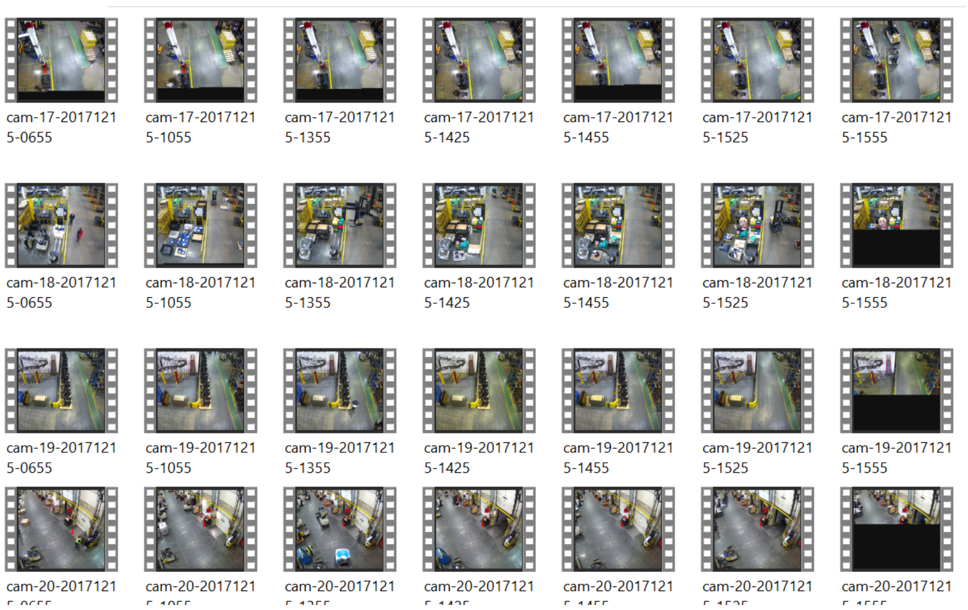


Figure A.5: Video data thumbnails on 2017 − 12 − 15

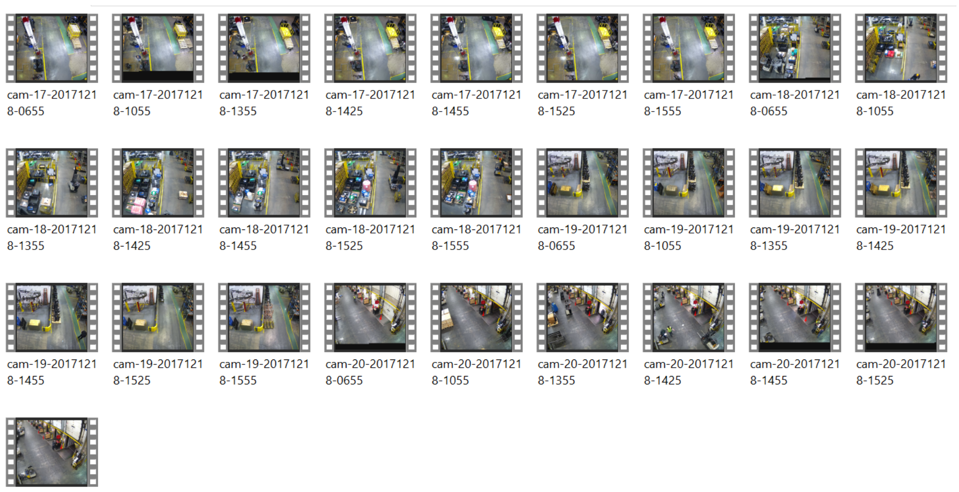


Figure A.6: Video data thumbnails on 2017 − 12 − 18

Note that both the camera 18 and 20 pair and the camera 17 and 19 pair have

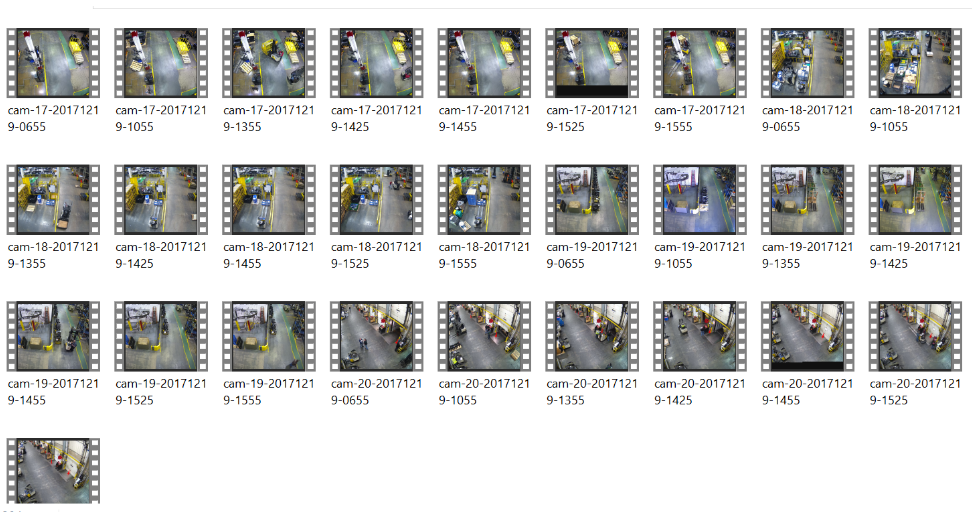


Figure A.7: Video data thumbnails on $2017 - 12 - 19$

overlapping fields of view. Figures A.8, A.9 and A.10 depict recordings on dates mentioned in respective captions.

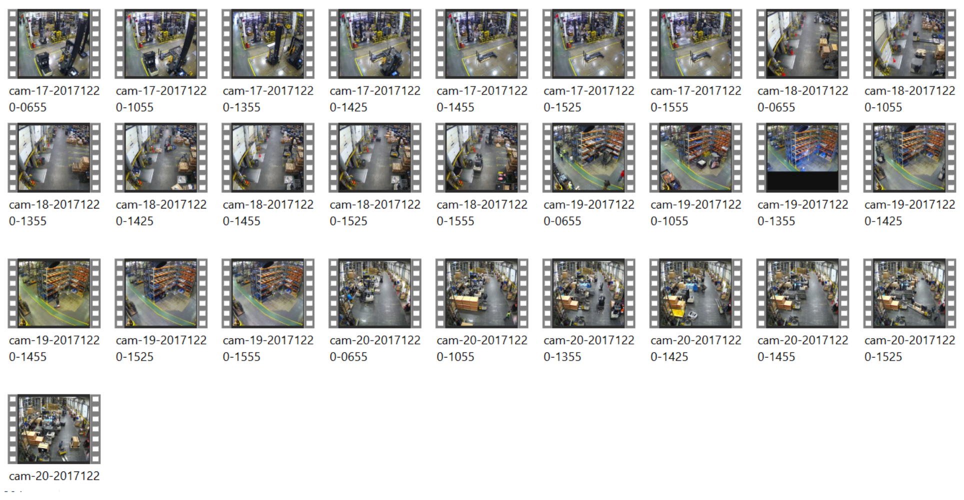


Figure A.8: Video data thumbnails on $2017 - 12 - 20$

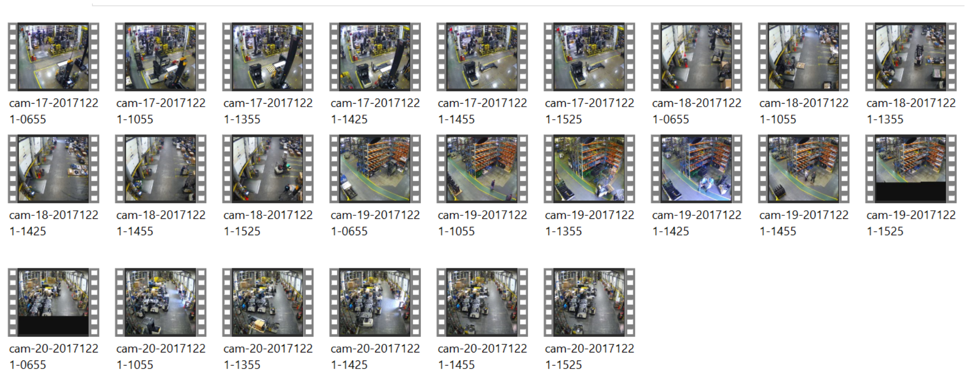


Figure A.9: Video data thumbnails on $2017 - 12 - 21$

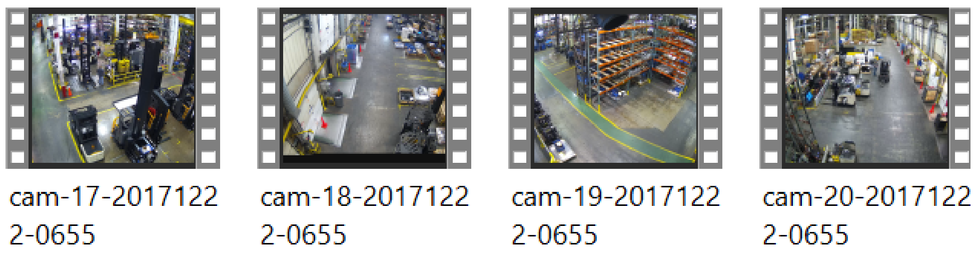


Figure A.10: Video data thumbnails on $2017 - 12 - 22$

A.3 Multiple Object Tracking Benchmark Dataset

This is a centralised benchmark dataset for the task of object detection and tracking, pedestrian tracking, 3D reconstruction, optic flow and stereo estimation. Though there are very limited benchmark datasets available online for standard MOT evaluation, but this one proved to be very helpful for MOT research, especially for tracking pedestrians. It has a large collection of datasets which has challenging sequences with varying attributes [100].

In this centralized benchmark interface for MOT, detections with various kind of state of art detectors for the video sequences are provided. It also contains ground truth dataset for the detections, for a fair comparison with various state of art detectors and trackers. A common evaluation platform is provided to evaluate various performance measures like *MOTA, MOTP, FPs, FNS* and others.

2DMOT2015

This is a collection of video sequences that include sequences with occluded backgrounds, distractors, similar-looking targets, non-linear motions and various tracking challenges. It is recorded in an unconstrained outdoor environment. This dataset is also being used in this work for tracking challenging pedestrian scenes. It also helped to understand various challenging person movements in warehouse premises.

Refer to Fig. A.11 for a walking pedestrian raw scene from a video sequence be-

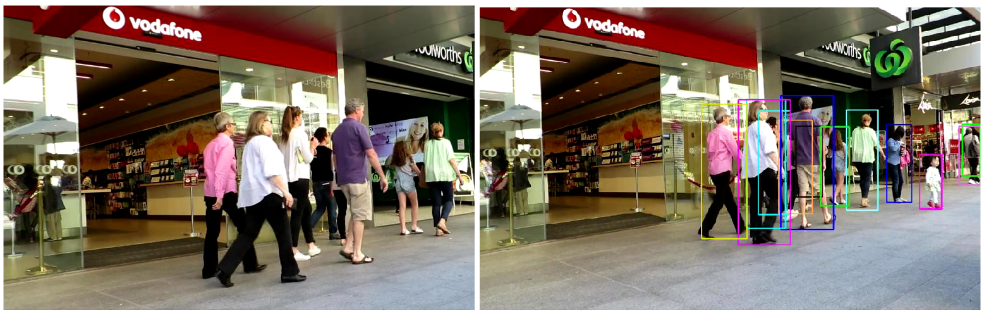


Figure A.11: ADL-Rundle-6 sequence 1. *Left*. A challenging occluded pedestrians scene. *Right*. Ground truth pedestrians

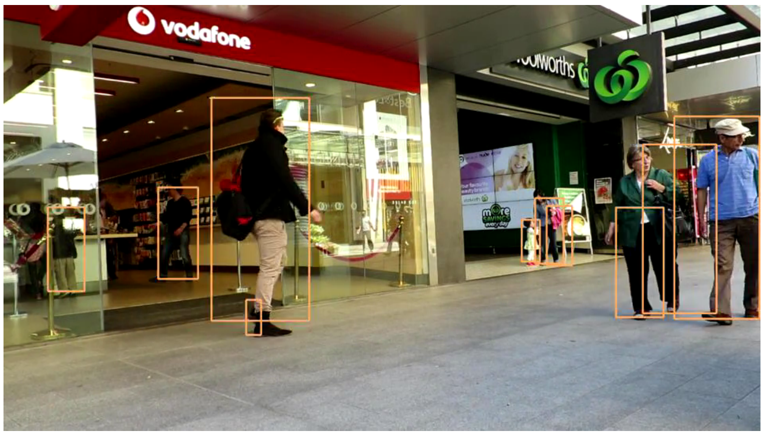


Figure A.12: Detections for the ADL-Rundle-6 sequence

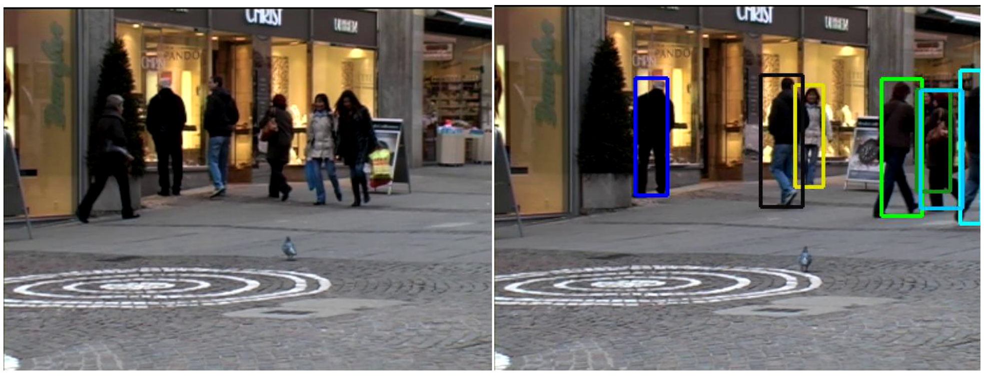


Figure A.13: TUD-Stadtmitte sequence. *Left*. A challenging similar looking occluded pedestrians. *Right*. Ground truth pedestrians

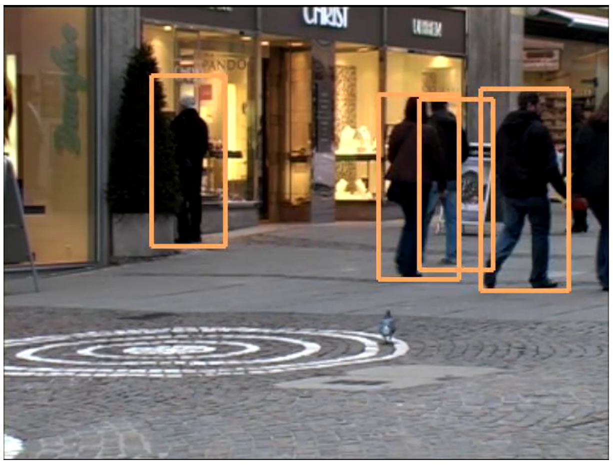


Figure A.14: Detections for the TUD-Stadtmitte sequence

longing to 2MOT2015 collection [101]. Figure A.12 refers to detections for this video sequence.

Refer to Fig. A.13 for a walking pedestrian raw scene from a video sequence belonging to 2DMOT2015 collection. The sequence is used for this work as well. It features multiple persons looking alike and facing backwards. It is a challenging scene to be tracked, with tricky track association problems for all frames. Figure A.14 refers to detections for this video sequence.

A.4 INRIA Person Dataset

For the detection of standing or upright people, this dataset was collected [38]. The images also contain partially occluded static or moving people with various colours of clothing, poses and background. The dataset features positive as well as negative test and train images in sequence collections [67]. Figure A.15 refers to sample examples.

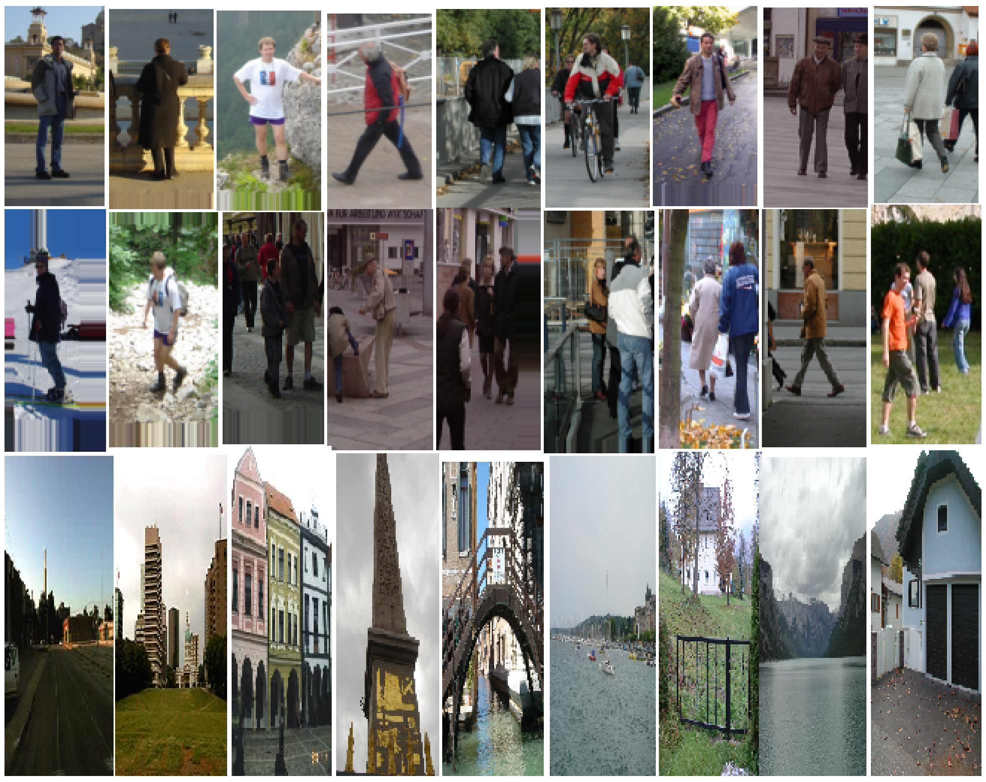


Figure A.15: The first two rows depict the positive pedestrians images in various poses, static or in motion, with various backgrounds and scene contexts. The bottom row shows the negative images from INRIA person dataset

A.5 Caltech Pedestrian Detection Benchmark

Caltech pedestrian dataset is composed of around 10 hours of 640×480 video data acquired from a moving vehicle in an urban environment. It contains $250,000$ frames with a total of $350,000$ bounding boxes and 2300 pedestrians. The corresponding annotation covers temporal correspondence between the bounding boxes and occlusion labels for the targets [30].

Figure A.16 refers to sample examples.

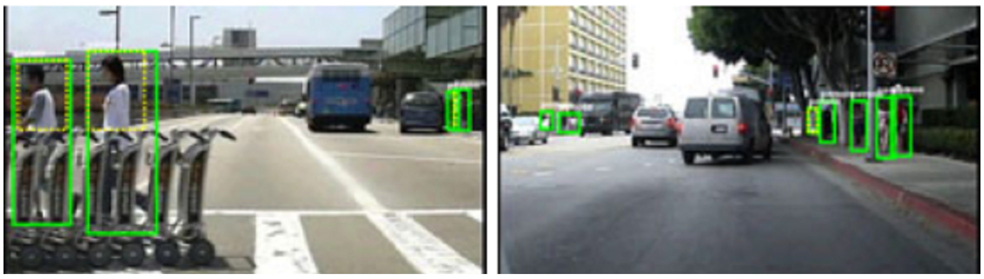


Figure A.16: The sample detection image for pedestrians from Caltech pedestrian detection benchmark

Bibliography

Authored Publications

[1] Fouzia, S., Bell, M. and Klette, R.: Tracking of Load Handling Forklift Trucks and of Pedestrians in Warehouses. Proc. International Computer Symposium, pp. 688–697, 2018. `https://doi.org/10.1007/978-981-13-9190-376`

[2] Fouzia, S., Bell, M. and Klette, R.: Saliency Guided Data Association Measure for Multiple-Object Tracking. International Symposium on Pervasive Systems, Algorithms, and Networks, (2019)

[3] Fouzia, S., Bell, M. and Klette, R.: Deep Learning-Based Improved Object Recognition in Warehouses. Proc. Pacific-Rim Symposium on Image and Video Technology, pp. 350–365, 2017. `https://doi.org/10.1007/978-3-319-75786-5_29`

[4] Fouzia, S., Bell, M. and Klette, R.: Improved Saliency-enhanced Multi-cue Correlation-filter-based Visual Tracking. The 9th Pacific-Rim Symposium on Image and Video Technology, (2019)

[5] Fouzia, S., Bell, M. and Klette, R.: Comparing Trackers for Multiple Targets in Warehouses. International Journal of Fuzzy Logic and Intelligent Systems 19(3): 147–157, 2019 .`https://doi.org/10.5391/IJFIS.2019.19.3.147`

Non-authored References

[6] Alikhani, I., Tavakoli, H.R., Rahtu, E. and Laaksonen, J.: On the contribution of saliency in visual tracking. Proc. VISIGRAPP, vol 4: VISAPP, pp. 17–21 (2016)

[7] Armesto, L., and Tornero, J. ,: A vision-based line tracking application. IEEE Automation in industrial vehicles In Emerging Technologies, pages 1–7 (2009)

[8] Achanta, R., Hemami, S., Estrada, F. and Süsstrunk, S.: Frequency-tuned salient region detection. Proc. IEEE Int. Conf. Computer Vision Pattern Recognition, pp. 1597–1604 (2009)

[9] Andriluka, M., Roth, S., and Schiele, B.: People tracking by detection and people detection by tracking. Proc. IEEE Int. Conf. Computer Vision Pattern Recognition, pp. 1–8.(2008)

[10] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P. and Süsstrunk, S.: SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern Analysis Machine Intelligence, 34(11), 2274–2282 (2012)

[11] Adam, A., Rivlin, E. and Shimshoni, I.: Robust fragments-based tracking using the integral histogram. IEEE Conf. Computer Vision Pattern Recognition, vol. 1, pp. 798–805 (2011)

[12] Özgür, C. , Alias, C. and Noche, B.,: Comparing sensor-based and camera-based approaches to recognizing the occupancy status of the load handling device of forklift trucks. Intl Journal logistics , Vol. 5 pp. 1–9 (2016)

[13] Aytekin, C., Cricri, F., and Aksu, E.: Saliency-enhanced robust visual tracking. arxiv.org/pdf/1802.02783.pdf (2018)

[14] Aytekin, C., Cricri, F., and Aksu, E.: Saliency-enhanced robust visual tracking. European Workshop Visual Information Processing, arXiv:1802.02783 (2018)

[15] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A. and Torr, P.H.: Fully-convolutional siamese networks for object tracking. Proc. European Conf. Computer Vision, pp. 850–865 (2016)

[16] Borges, P.V.K., Zlot, R. and Tews, A.: Integrating off-board cameras and vehicle on-board localization for pedestrian safety. IEEE Transactions on Intelligent Transportation Systems, 14(2), pp.720–730 (2013)

[17] Bochinski, E., Eiselein, V. and Sikora, T.: High-speed tracking-by-detection without using image information. Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance, pp. 1–6, 2017. `https://doi.org/10.1109/AVSS.2017.8078516`

[18] Breiman, L., and Cutler, A.: Random Forests. (2004).

[19] Berclaz, J., Fleuret, F., Turetken, E. and Fua, P.: Multiple object tracking using k-shortest paths optimization. IEEE Trans. Pattern Analysis Machine Intelligence, 33(9), pp. 1806–1819 (2011)

[20] Bae, S.H. and Yoon, K.J.: Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. IEEE Trans. Pattern Analysis Machine Intelligence, 40(3), pp. 595–610 (2017)

[21] Babenko, B., Yang, M.H. and Belongie, S.: Robust object tracking with online multiple instance learning. IEEE Trans. Pattern Analysis Machine Intelligence, 33(8), 1619–1632 (2010)

[22] Bailer, C., Pagani, A. and Stricker, D.: A superior tracking approach: Building a strong tracker through fusion. Proc. European Conf. Computer Vision, pp. 170–185 (2014)

[23] Bolme, D.S., Beveridge, J.R., Draper, B.A. and Lui, Y.M.: Visual object tracking using adaptive correlation filters. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 2544–2550 (2010)

[24] Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B.: Simple online and realtime tracking. Proc. IEEE Int. Conf. Image Processing, pp. 3464–3468 (2016)

[25] Broida, T.J. and Chellappa, R.: Estimation of object motion parameters from noisy images. IEEE Trans. Pattern Analysis Machine Intelligence, 1(1):90–99 (1986)

[26] Cane, T. and Ferryman, J.: Saliency-based detection for maritime object tracking. Proc. IEEE Conference on Computer Vision Pattern Recognition Workshops, pp. 18–25, 2016. `https://doi.org/10.1109/CVPRW.2016.159`

[27] Bernardin, K. and Stiefelhagen, R.: Evaluating multiple object tracking performance: The CLEAR MOT metrics. Journal of Image Video Processing, p. 1, 2008. `https://doi.org/10.1155/2008/246309`

[28] Csurka, G., et al: Visual categorization with bags of keypoints. Workshop on statistical learning in computer vision, ECCV. Vol. 1. No. 1-22. 2004. `https://doi.org/10.1.1.72.604`

[29] Chen, L., Ai, H., Shang, C., Zhuang, Z. and Bai, B.: Online multi-object tracking with convolutional neural networks. IEEE International Conference on Image Processing (ICIP), pp. 645-649, 2017. `https://doi.org/10.1109/ICIP.2017.8296360`

[30] Caltech pedestrian detection benchmark. `www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/` `https://doi.org/10.1109/CVPR.2009.5206631`

[31] Cheng, M. M., Mitra, N. J., Huang, X., Torr, P. H., and Hu, S. M.: Global contrast based salient region detection. IEEE Trans. Pattern Analysis Machine Intelligence, 37(3), 569–582 (2015)

[32] Chang, C.,C., and Lin, C.: LIBSVM: a library for support vector machines. ACM Trans intelligent systems and technology, 2(3), 27(2011)

[33] Cunningham, P. and Delany, S.J.: k-Nearest neighbour classifiers. Multiple Classifier Systems, 34(8), pp.1–17 (2007)

[34] Cucchiara, R. , Piccardi, M., and Prati, A.,: Focus based Feature Extraction for Pallets Recognition. In BMVC pages 1–10 (2000)

[35] Comaniciu, D., Ramesh, V. and Meer, P.: Real-time tracking of non-rigid objects using mean shift. Proc. IEEE Conf. Computer Vision Pattern Recognition, vol. 2, pp. 142–149 (2000)

[36] Dinh, T.B., Vo, N. and Medioni, G.: Context tracker: Exploring supporters and distracters in unconstrained environments. IEEE Conf. Computer Vision Pattern Recognition, pp. 1177–1184 (2011)

[37] Danelljan, M., Bhat, G., Shahbaz Khan, F. and Felsberg, M.: Eco: Efficient convolution operators for tracking. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 6638–6646 (2017)

[38] Dalal, N., and B. Triggs.: Histograms of oriented gradients for human detection. Proc: Computer Vision and Pattern Recognition, 2005. `https://doi.org/10.1109/CVPR.2005.177`

[39] Donahue, J., et al. "Long-term recurrent convolutional networks for visual recognition and description. " In Proc:*Conference on computer vision and pattern recognition* 2015.

[40] Danelljan, M., Häger, G., Khan, F. and Felsberg, M.: Accurate scale estimation for robust visual tracking. British Machine Vision Conference, (2014)

[41] Danielsson, V., and G. Smajli: Improving warehousing operations with video technology. Master thesis, Lund University, (2015).

[42] Drulea, M., I. Szakats, A. Vatavu, and S. Nedevschi: Omnidirectional stereo vision using fisheye lenses. Proc. IEEE International conference Intelligent Computer Communication Processing, pp 251–258, 2014 `https://doi.org/10.1109/ICCP.2014.6937005`

[43] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell,T.: Decaf: A deep convolutional activation feature for generic visual recognition. Proc. Int. Conf. Machine Learning, pages 647655 (2014)

[44] Danelljan, M., Hager, G., Shahbaz Khan, F. and Felsberg, M.: Convolutional features for correlation filter based visual tracking. Proc. IEEE Int. Conf. Computer Vision Workshops, pp. 58–66 (2015)

[45] Danelljan, M., Hager, G., Shahbaz Khan, F. and Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. Proc. IEEE Int. Conf. Computer Vision, pp. 4310–4318 (2015)

[46] Danelljan, M., Robinson, A., Khan, F.S. and Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. Proc. European Conf. Computer Vision, pp. 472–488 (2016)

[47] Felzenszwalb, P. F. , et al: Object detection with discriminatively trained part-based models. Pattern analysis and machine intelligence, 32(9), 1627–1645, 2010. `https://doi.org/10.1109/TPAMI.2009.167`

[48] Fang, K., Xiang, Y., Li, X. and Savarese, S.: Recurrent autoregressive networks for online multi-object tracking. Proc. Winter Conference on Applications of Computer Vision, pp. 466–475, 2018. `https://doi.org/10.1109/WACV.2018.00057`

[49] Feichtenhofer, C., A. Pinz, and A. Zisserman. "Convolutional two-stream network fusion for video action recognition. " Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[50] Friedman, N., and Russell, S.: Image segmentation in video sequences: A probabilistic approach. Proc. Conf. Uncertainty Artificial Intelligence, pages 175–181 (1997)

[51] Franke, U., C. Rabe, H. Badino, and S. Gehrig. 6D-vision: Fusion of stereo and motion for robust environment perception. In Proc. *Pattern Recognition (DAGM)*, pages LNCS 3663, Springer, (2005).

[52] Garibotto, G., S. Masciangelo, P. Bassino, C. Coelho, A. Pavan, and M. Marson. Industrial exploitation of computer vision in logistic automation: autonomous control of an intelligent forklift truck. In Proc. *IEEE Int. Conf. Robotics Automation*, volume 2, pages 1459–1464, (1998)

[53] Girshick, R.: Fast R-CNN. Proc: Int. Conf. Computer Vision, pages 1440–1448 (2015)

[54] Girshick, R., Donahue, J., Darrell, T., and Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. IEEE Conf. Computer Vision Pattern Recognition, pages 580–587 (2014)

[55] Guo, D. J., Zhe-Ming, L., and Hao, L.: Multi-channel adaptive mixture background model for real-time tracking. J. Information Hiding Multimedia Signal Processing, 7, 216–221 (2016)

[56] Hamberg, R., and J. Verriet (eds.). *Automation in Warehouse Development.* Springer, London, (2012).

[57] Hamid Rezatofighi, S., Milan, A., Zhang, Z., Shi, Q., Dick, A. and Reid, I.: Joint probabilistic data association revisited. Proc. IEEE Int. Conf. Computer Vision, pp. 3047–3055 (2015)

[58] Haines, T., and Xiang, T.: Background subtraction with Dirichlet processes. Proc. European Conf. Computer Vision, pages 99–113 (2012)

[59] He, K., Zhang, X., Ren, S. and Sun, J., 2014, September. Spatial pyramid pooling in deep convolutional networks for visual recognition. Proc. European Conference Computer Vision, pages 346–361 (2014)

[60] Hou, X., and Zhang, L.: Saliency detection: A spectral residual approach. Proc. IEEE Conf. Computer Vision Pattern Recognition, pages 1–8 (2007)

[61] Hershey, S., Chaudhuri, S., Ellis, D.P., Gemmeke, J.F., Jansen, A., Moore, R.C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B. and Slaney, M.: CNN architectures for large-scale audio classification. Proc. IEEE Int. Conf. Acoustics Speech Signal Processing, pp. 131–135 (2017)

[62] Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M.M., Hicks, S.L. and Torr, P.H.: Struck: Structured output tracking with kernels. IEEE Trans. Pattern Analysis Machine Intelligence, 38(10), 2096–2109 (2015)

[63] Henriques, J.F., Caseiro, R., Martins, P. and Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Analysis Machine Intelligence, 37(3), 583–596 (2014)

[64] Han, Z., Ye, Q. and Jiao, J.: Online feature evaluation for object tracking using Kalman filter. Proc. IEEE International Conference on Pattern Recognition, pp. 1–4, 2008. `https://doi.org/10.1109/ICPR.2008.4761152`

[65] Harel, J., Koch, C. and Perona, P.: Graph-based visual saliency. Proc. Advances Neural Information Processing, pp. 545–552, (2007).

[66] He, S., R. W. Lau, W. Liu, Z. Huang, and Q. Yang: SuperCNN: A superpixelwise convolutional neural network for salient object detection. Int. J. Comput. Vis., Vol. 115, No. 3, pp. 330–344, (2015).

[67] INRIA person dataset. `http://pascal.inrialpes.fr/data/human/`

[68] Intelligent Video Analytics for: The Warehousing and Distribution Centres Market. See on page `https://www.security.honeywell.com/uk/documents/VideoAnalytics_Warehouses_UK.pdf/.Lastvisit:26July(2017)`

[69] Insafutdinov, E., Andriluka, M., Pishchulin, L.,Tang, S., Levinkov, E., Andres, B. and Schiele, B.: Arttrack: Articulated multi-person tracking in the wild. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 6457–6465 (2017)

[70] Jeon, S., et al. "Localization of pallets based on passive RFID tags. " IEEE intern Conf in Information Technology: New Generations, (2010)

[71] Jiang, H., Fels, S. and Little, J.J.: A linear programming approach for multiple object tracking. IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8, 2007. `https://doi.org/10.1109/CVPR.2007.383180`

[72] Knuth, D. E., T. L. Larrabee, P. M. Roberts: *Mathematical Writing*, Mathematical Association of America, ISBN: 088385063X, 1996.

[73] Knight, W. : Self-taught robot is ready to seize another warehouse job. MIT Technology Review, April 6, (2016).

[74] Krizhevsky, A., I. Sutskever, and G E. Hinton: Imagenet classification with deep convolutional neural networks. International Advances in neural information processing systems, pp. 1097–1105, 2012. `https://doi.org/10.1061/(ASCE)GT.1943-5606.0001284`

[75] KaewTraKulPong, P., and Bowden, R.: An improved adaptive background mixture model for real time tracking with shadow detection. Proc. European Workshop Advanced Video Based Surveillance Systems, pages 135-144 (2002)

[76] Klette, R.: Concise Computer Vision. Springer, London (2014)

[77] Khalid, O., SanMiguel, J.C. and Cavallaro, A.: Multi-tracker partition fusion. IEEE Trans. Circuits Systems Video Technology, 27(7), pp.1527–1539 (2016)

[78] Kim, C., Li, F., Ciptadi, A. and Rehg, J.M.: Multiple hypothesis tracking revisited. Proc. Int. Conference on Computer Vision pp. 4696–4704 (2015)

[79] Kristan, M., et al.: The visual object tracking VOT2016 challenge results. Proc. European Conf. Computer Vision Workshops, pp. 777–823, (2016)

[80] Kristan, M., et al.: The visual object tracking VOT2017 challenge results. Proc. IEEE Int. Conf. Computer Vision Workshops, pp. 1949–1972 (2017)

[81] Kuhn, H.W.: The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1–2), pp. 83-97, 1955 https://doi.org/10.1007/978-3-540-68279-0_2

[82] Kwon, J. and Lee, K.M.: Visual tracking decomposition, I. IEEE Conf. Computer Vision Pattern Recognition, pp. 1269–1276 (2010)

[83] Kwon, J. and Lee, K.M.: Tracking by sampling trackers. IEEE Int. Conf. Computer Vision, pp. 1195–1202 (2010)

[84] Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Zhao, X. and Kim, T.K.: Multiple object tracking: A literature review. arXiv preprint arXiv:1409.7618, (2014)

[85] Lowe, D. G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60 (2):91–110, 2004. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[86] Li, G., and Y. Yu: Deep contrast learning for salient object detection. Conference on Computer Vision and Pattern Recognition, pp. 478–487, (2016)

[87] Lenz, P., Geiger, A. and Urtasun, R.: Followme: Efficient online min-cost flow tracking with bounded memory and computation. Proc. IEEE International Conference on Computer Vision, pp. 4364–4372, 2015. arXiv:1407.6251v2

[88] Leal-Taixé, L., Canton-Ferrer, C. and Schindler, K.: Learning by tracking: Siamese CNN for robust target association. IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 33–40, 2016. https://doi.org/10.1109/CVPRW.2016.59

[89] LeCun, Y., Yoshua, B., and Geoffrey, H.: Deep learning. Nature, 521, 436–444 (2015).

[90] Lee, H. R., Grosse, R., Ranganath, and A., Ng.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations Proc. Int. Conf. Machine Learning, pages 609–616 (2009).

[91] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Computer Vision, 60(2), 91–110 (2004)

[92] Liu, F., Jia, W. and Yang, Z.: A multi-object tracking method based on bounding box and features. Proc. Int. Conf. Computer Science Engineering Education Applications, pp. 217–227 (2019)

[93] Lin, Z., Hua, G. and Davis, L.S.: Multiple instance feature for robust part-based object detection. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 405–412 (2009)

[94] Li, J., Deng, C., Da Xu, R.Y., Tao, D. and Zhao, B.: Robust object tracking with discrete graph-based multiple experts. IEEE Trans. Image Processing. 26(6), pp.2736–2750 (2017)

[95] Li, H., Shen, C. and Shi, Q.: Real-time visual tracking using compressive sensing. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 1305–1312 (2011)

[96] Liu, N., and Han, J.: DhsNet: Deep hierarchical saliency network for salient object detection. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 678–686 (2016)

[97] Liu, B., Huang, J., Yang, L. and Kulikowsk, C.: Robust tracking using local sparse appearance model and k-selection. IEEE Conf. Computer Vision Pattern Recognition, pp. 1313–1320 (2011)

[98] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C.: SSD: Single shot multibox detector. Proc. European Conf. Computer Vision, pp. 21–37 (2016)

[99] Z. Miljković, N. Vuković, M. Mitić, and B. Babić. New hybrid vision-based control approach for automated guided vehicles. *Int. J. Advanced Manufacturing Technology*, 66(1):231–249, (2013).

[100] Multi-object tracking benchmark. `motchallenge.net` arXiv:1603.00831v2, (2016)

[101] Milan, A., Leal-Taix, L., Reid, I., Roth, S. and Schindler, K.: MOT16: A benchmark for multi-object tracking. arXiv:1603.00831 9 (2016)

[102] Ma, C., Huang, J.B., Yang, X. and Yang, M.H.: Hierarchical convolutional features for visual tracking. Proc. IEEE Int. Conf. Computer Vision, pp. 3074–3082 (2015)

[103] Panda, D. K., and Meher, S.: A Gaussian mixture model with Gaussian weight learning rate and foreground detection using neighbourhood correlation. Proc. IEEE Asia Pacific Conf. Postgrad Research Microelectronics, pages 158–163 (2013)

[104] Perronnin, F., Sanchez, J., and Mensink, T.: Improving the Fisher kernel for large-scale image classification. Proc. ECCV, pages 143–156 (2010)

[105] Perazzi, F., Krahenbuhl, P., Pritch, Y., and Hornung, A.: Saliency filters, contrast based filtering for salient region detection. Proc. IEEE Conf. Computing Vision Pattern Recognition, pages 733–740 (2012)

[106] Papageorgiou, C., Oren, M., Poggio T.: A general framework for object detection. Proc. IEEE Int. Conf. Computer Vision, pp. 555–562 (1998)

[107] Pagès, J., Armangué, X., Salvi, J., Freixenet, J.,, and Martí, J.: A computer vision system for autonomous forklift vehicles in industrial environments. In Proc. *Mediterranean Conf. Control Automation*, pages 1–6 (2001)

[108] Prasad, P. and Gupta, A.: Moving object tracking and detection based on Kalman filter and saliency mapping. Proc. Data Engineering Intelligent Computing, pp. 639–646 (2018)

[109] Rennie, C., Shome, R. , Bekris, K. E., and De Souza, A. F. ,: A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and place. IEEE Robotics and Automation Letters, 1(2), pages 11791185 (2016)

[110] Rudinac, M. and Jonker, P. P. ,: Saliency detection and object localization in indoor environments. In Pattern Recognition (ICPR), International Conference In Pattern Recognition, pp 404-407 (2010)

[111] Ren, S., He, K., Girshick, R. and Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. Proc. Advances Neural Information Processing Systems, pp. 91–99, 2015 `https://doi.org/10.1109/TPAMI.2016.2577031`

[112] Redmon, J. and Farhadi, A.: YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)

[113] Sun, S., Akhtar, N., Song, H., Mian, A. and Shah, M.: Deep affinity network for multiple object tracking. 2018. `https://doi.org/10.1109/TPAMI.2019.2929520`

[114] Simonyan, K. and Zisserman, A.: Very deep convolutional networks for large-scale image recognition. 2014. `https://doi.org/10.1109/ACPR.2015.7486599`

[115] Stauffer, C., and Grimson, W. E. L.: Adaptive background mixture models for real-time tracking. Proc. IEEE Computer Vision Pattern Recognition, volume 2, pages 246–252, 1999. `https://doi.org/10.1109/CVPR.1999.784637`

[116] Simonyan, K. and Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[117] Simon, D.: Kalman filtering with state constraints:A survey of linear and nonlinear algorithms. IET Control Theory and Applications, 4(8), pp.1303–1318 (2010)

[118] Sidibé, D., Fofi, D. and Mériaudeau, F.: Using visual saliency for object tracking with particle filters. Proc. European Signal Processing Conf., pp. 1776–1780, 2010 `https://doi.org/10.5281/zenodo.41894`

[119] Sermanet, P., Kavukcuoglu, K., Chintala, S. C., and Le-Cun, Y.: Pedestrian detection with unsupervised multi-stage feature learning. Proc. IEEE Conf. Computer Vision Pattern Recognition, pages 3626–3633 (2013)

[120] Sohn, K., Zhou G., Lee, C., and Lee, H.: Learning and selecting features jointly with point-wise gated Boltzmann machines. Proc. Int. Conf. Machine Learning, pages 217–225, (2013)

[121] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A.: Going deeper with convolutions. Proc. IEEE Conf. Computer Vision Pattern Recognition, pages 1–9 (2015)

[122] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z.: Rethinking the inception architecture for computer vision Proc. IEEE Conf. Computer Vision Pattern Recognition, pages 2818–2826 (2016)

[123] Spada, S.: Vision systems in the warehouse and the middle child syndrome. *Logistics Viewpoints*, `logisticsviewpoints.com/`, posted April 30, (2014).

[124] Schulter, S., Vernaza, P., Choi, W. and Chandraker, M., 2017. Deep network flow for multi-object tracking. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 6951–6960 (2017).

[125] Shi, J. and Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Analysis Machine Intelligence, 22(8):888–905 (2000)

[126] Tavakoli, H.R., Moin, M.S. and Heikkilä, J.: Local similarity number and its application to object tracking. Int. J. Advanced Robotic Systems, 10(3), p. 184 (2013)

[127] Tuohy, S., O'Cualain, D., Jones, E. and Glavin, M.: Distance determination for an automobile environment using inverse perspective mapping in OpenCV. pp. 100–105 (2010)

[128] Tran, D., L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In Pro:" *IEEE International Conference on Computer Vision*, pp. 4489-4497, 2015

[129] TensorFlow: How to retrain inception's final layer for new categories. See on page `www.tensorflow.org/tutorials/`. Last visit: 26 July 2017

[130] Tian, Y. L., Lu, M., and Hampapur, A.: Robust and efficient foreground analysis for real-time video surveillance. Proc. IEEE Conf. Computer Vision Pattern Recognition, volume 1, pages 1182–1187 (2005)

[131] Tesfaye, Y.T., Zemene, E., Prati, A., Pelillo, M. and Shah, M.: Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. arXiv preprint arXiv:1706.06196 (2017).

[132] Tang, S., Andres, B., Andriluka, M. and Schiele, B.: Subgraph decomposition for multi-target tracking. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 5033-5041 (2015)

[133] Utilizing Video Analytics for Warehouse Traffic Optimization. See on page `http://surveillancesecure.com/4223-2/.Lastvisit:26July(2017)`

[134] Vedaldi, A. and Lenc, K.: MatConvNet: Convolutional neural networks for Matlab. ACM Int. Conf. Multimedia, pp. 689–692 (2015)

[135] Varga, R. , Costea, A. and Nedevschi, S. , 2015, September. Improved autonomous load handling with stereo cameras IEEE Intelligent Computer Communication and Processing (ICCP), pp. 251-256 (2015)

[136] Van De Weijer, J., Schmid, C., Verbeek, J. and Larlus, D.: Learning color names for real-world applications. IEEE Trans. Image Processing, 18(7), 1512–1523 (2009)

[137] Wu, Y., Liu, Y., Li, J., Liu, H., and Hu, X.: Traffic sign detection based on convolutional neural networks. Proc. Int. Joint Conf. Neural Networks, pages 1–7, (2013)

[138] Wang, N., Zhou, W., Tian, Q., Hong, R., Wang, M. and Li, H.: Multi-cue correlation filters for robust visual tracking. Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 4844–4853 (2018)

[139] Wojke, N., Bewley, A. and Paulus, D.: Simple online and realtime tracking with a deep association metric. Proc. IEEE Int. Conf. Image Processing, pp. 3645–3649, 2017. `https://doi.org/10.1109/ICIP.2017.8296962`

[140] https://www.linkedin.com/pulse/warehouse-management-system-m-k-bhardwaj

[141] Wu, Y., Lim, J. and Yang, M.H.: Object tracking benchmark. IEEE Trans. Pattern Analysis Machine Intelligence. 37(9), 1834–1848 (2015)

[142] Xia, Y., Hu, R., Wang, Z., and Lu, T.: Moving foreground detection based on spatio-temporal saliency. Int. J. Computer Science Issues, 10(3), 79–84 (2013)

[143] Xia, H., Song, S., and He, L.: A modified Gaussian mixture background model via spatiotemporal distribution with shadow detection. Signal Image Video Processing, 2(10), 343–350 (2016)

[144] Xue, Y. and Liu, H. ,: Intelligent Storage and Retrieval Systems Based on RFID and Vision in Automated Warehouse. JNW, 7(2), pp. 365-369(2012)

[145] Xiang, J., Zhang, G., Hou, J., Sang, N. and Huang, R.: Multiple target tracking by learning feature representation and distance metric jointly. arXiv preprint arXiv:1802.03252. 2018

[146] Xu, Y., Zhou, X., Chen, S. and Li, F.: Deep learning for multiple object tracking: a survey. IET Computer Vision, 13(4), pp.355–368, 2019. https://doi.org/10.1049/iet-cvi.2018.5598

[147] Yoon, K., Kim, D.Y., Yoon, Y.C. and Jeon, M.: Data association for multi-object tracking via deep neural networks. Sensors, 19(3), p. 559, 2019. https://doi.org/10.3390/s19030559

[148] Yu, F., Li, W., Li, Q., Liu, Y., Shi, X. and Yan, J.: POI: Multiple object tracking with high performance detection and appearance feature. Proc. European Conf. Computer Vision, pp. 36–42 (2016)

[149] Yousefhussien, M.A., Browning, N.A. and Kanan, C.: Online tracking using saliency. Proc. IEEE Winter Conf. Applications Computer Vision (WACV) pp. 1–10 (2016)

[150] https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

[151] https://pjreddie.com/darknet/yolo/

[152] Zang, Q., and Klette, R.: Evaluation of an adaptive composite Gaussian model in video surveillance. Proc. Computer Analysis Images Patterns, LNCS 2756, pages 165–172 (2003)

[153] Zhang, P., Wang, D., Lu, H., Wang, H., and Yin, B.: Learning uncertain convolutional features for accurate saliency detection. Proc. Int. Conf. Computer Vision, pp. 212–221 (2017)

[154] Zhou, L., Yang, Z., Yuan, Q., Zhou, Z., and Hu, D.: Salient region detection via integrating diffusion-based compactness and local contrast. Proc. IEEE Trans. Image Processing, 11, 3308–3320 (2015)

[155] Zitnick, C. L., and P. Dollar: Edge boxes: Locating object proposals from edges. Proc. European Conf. Computer Vision, pp. 391–405 (2014)

[156] Zivkovic, Z.: Improved adaptive Gaussian mixture model for background subtraction. Proc. IEEE Int. Conf. Pattern Recognition, volume 2, pages 28–31 (2004)

[157] Zhang, P., Wang, D., Lu, H. and Wang, H.: Non-rigid object tracking via deep multi-scale spatial-temporal discriminative saliency maps. arXiv preprint arXiv:1802.07957 (2018)

[158] Zhang, J., Ma, S. and Sclaroff, S.: MEEM: Robust tracking via multiple experts using entropy minimization. Proc. European Conf. Computer Vision, pp. 188–203 (2014)

[159] Zhang, G., Yuan, Z., Zheng, N., Sheng, X. and Liu, T.: Visual saliency based object tracking. Proc. Asian Conf. Computer Vision, pp. 193–203 (2009)

[160] Zhang, K., Zhang, L., Liu, Q., Zhang, D. and Yang, M.H.: Fast visual tracking via dense spatio-temporal context learning. Proc. European Conf. Computer Vision, pp. 127–141 (2014)

[161] Choi, W.: Near-online multi-target tracking with aggregated local flow descriptor. Proc. IEEE Int. Conf. Computer Vision, pages 3029–3037, (2015)

Index